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], Pantelis G. Bagos[1,*]

[1]Department of Computer Science and Biomedical Informatics, University of Thessaly, 35100 Lamia, Greece,

[2]Department of Health Technology, Section for Bioinformatics, Technical University of Denmark, Kgs Lyngby, Denmark,

[3] Genekor Medical SA, Athens,

[4] Institute of Translational Medicine and Therapeutics, University of Pennsylvania, Philadelphia PA, USA

[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.

Supplementary Material

PART A – Usage scenario

A Hidden Markov Model is a probabilistic model defined by a discrete set of "hidden" states, a discrete set of observed symbols (e.g. amino acid residues in the case of proteins), and two set of distribution probabilities; the state transition probability distribution (transitions) and the observed symbol probability distribution (emissions). Let's assume we want to model a protein sequence analysis problem. To do this, we need to specify the state space, the initial probabilities, and the transition probabilities.

Let's consider a specific protein sequence analysis problem, such us the prediction of transmembrane regions of alpha-helical membrane proteins with the ability to incorporate prior topological information. We first need to design the model and consider the number of states and which the permissible transitions are. Figure 1 shows a schematic representation of the model used in HMM-TM, which is the example presented in this scenario.

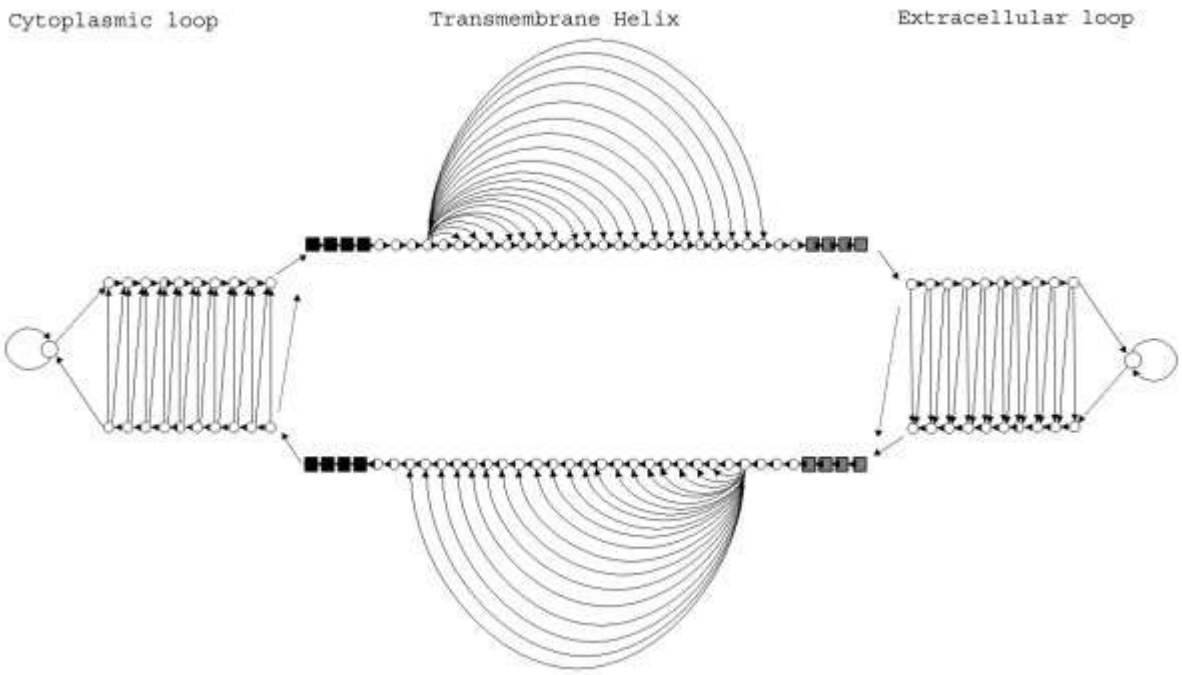


Figure S1. A schematic representation of the HMM-TM model's architecture

Knowing the number of states of the model, the first step is to define the transition probabilities. They are simply the probabilities of staying in the same state or moving to a different state given the current state. Here, the user could use a spreadsheet to create an NxN table where N is the number of states of the model. Each transition probability depends on the biological problem. The transition matrix is perhaps the most important step because non-zero elements define the allowed transitions and thus the model architecture. In the example of alpha-helical transmembrane segments predictions, there are transmembrane states (M and m), extracellular states (O and o), intracellular states (I and i) and the Begin (B) and End (E) states. All non-zero probabilities one can see in Figure 1, show permissible transitions between different states. Each line must sum to 1. For each line, the user can define a function to check the sum. Note that each state can have a name with more than one character; in the particular case for convenience we chose state names that remind us the type of the state, for instance M01, M02 for the membrane states, and so on.

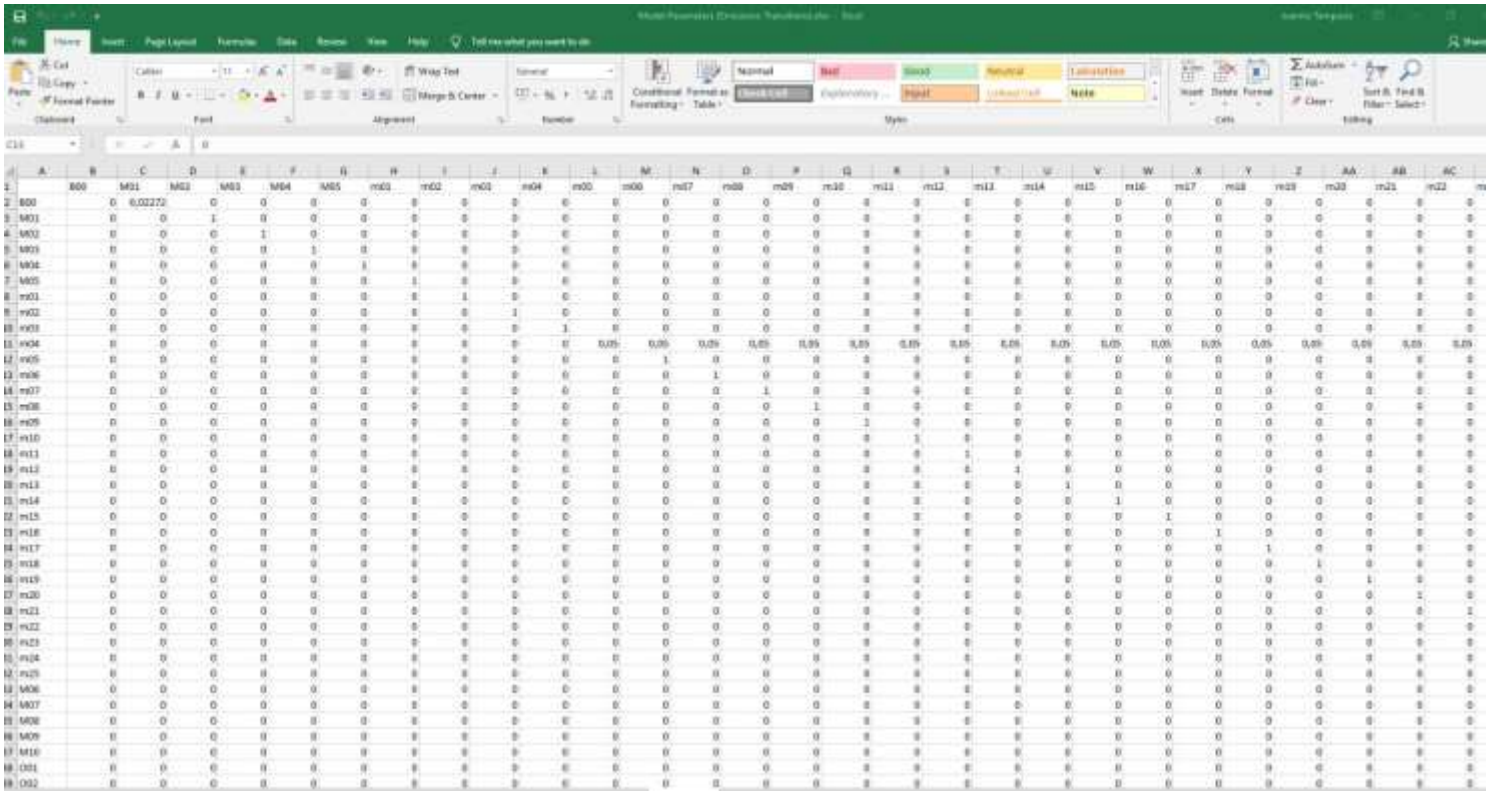
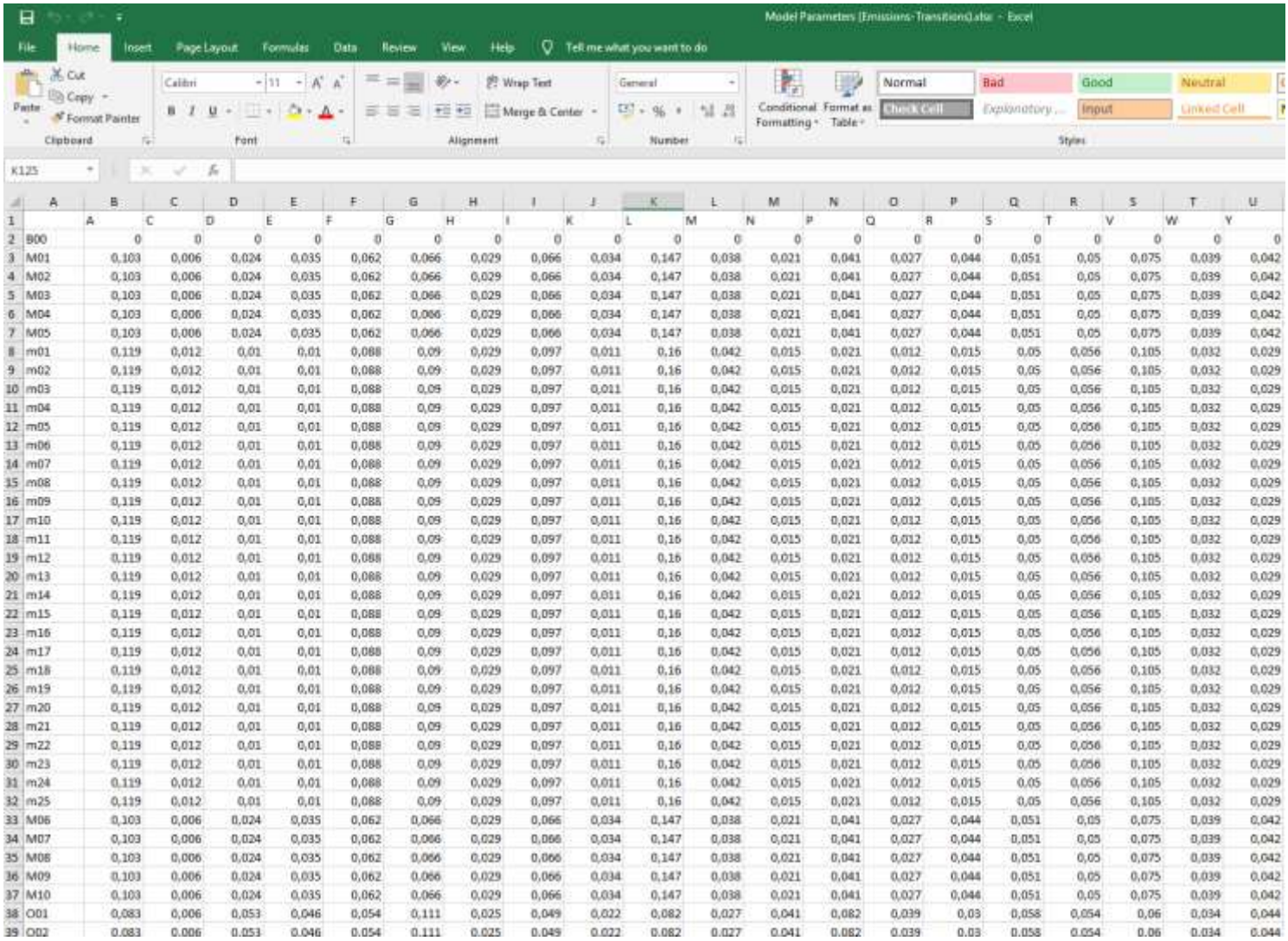


Figure S2. The transition probabilities table

In the end, the user will copy the column with probabilities and paste at the text editor and save it at the hard disk.

The next step is to define the emission probabilities. The emission probability represents how likely a symbol is to be emissioned on each state. The size of this set depends on the nature of the observed variable. In the case of proteins, an observed sequence is composed of a discrete set of 20 symbols, following the alphabetical order of single-letter codes for the 20 amino acids: A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y. In the example of alpha-helical transmembrane topology prediction, a Nx20 table represents the emission probabilities and should be initialized. Again, the user can use a spreadsheet. The user defines a column for each symbol and a line for each state. Each line must sum to 1. For each line, user can define a function to check the sum. The user can use some plausible values for initializing the emissions or use uniform probabilities and let JUCHMME find the appropriate starting values, using the respective option.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
2	B00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	M01	0,103	0,006	0,024	0,035	0,062	0,066	0,029	0,066	0,034	0,147	0,038	0,021	0,041	0,027	0,044	0,051	0,05	0,075	0,039	0,042
4	M02	0,103	0,006	0,024	0,035	0,062	0,066	0,029	0,066	0,034	0,147	0,038	0,021	0,041	0,027	0,044	0,051	0,05	0,075	0,039	0,042
5	M03	0,103	0,006	0,024	0,035	0,062	0,066	0,029	0,066	0,034	0,147	0,038	0,021	0,041	0,027	0,044	0,051	0,05	0,075	0,039	0,042
6	M04	0,103	0,006	0,024	0,035	0,062	0,066	0,029	0,066	0,034	0,147	0,038	0,021	0,041	0,027	0,044	0,051	0,05	0,075	0,039	0,042
7	M05	0,103	0,006	0,024	0,035	0,062	0,066	0,029	0,066	0,034	0,147	0,038	0,021	0,041	0,027	0,044	0,051	0,05	0,075	0,039	0,042
8	m01	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
9	m02	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
10	m03	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
11	m04	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
12	m05	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
13	m06	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
14	m07	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
15	m08	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
16	m09	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
17	m10	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
18	m11	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
19	m12	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
20	m13	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
21	m14	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
22	m15	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
23	m16	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
24	m17	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
25	m18	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
26	m19	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
27	m20	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
28	m21	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
29	m22	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
30	m23	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
31	m24	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
32	m25	0,119	0,012	0,01	0,01	0,088	0,09	0,029	0,097	0,011	0,16	0,042	0,015	0,021	0,012	0,015	0,05	0,056	0,105	0,032	0,029
33	M06	0,103	0,006	0,024	0,035	0,062	0,066	0,029	0,066	0,034	0,147	0,038	0,021	0,041	0,027	0,044	0,051	0,05	0,075	0,039	0,042
34	M07	0,103	0,006	0,024	0,035	0,062	0,066	0,029	0,066	0,034	0,147	0,038	0,021	0,041	0,027	0,044	0,051	0,05	0,075	0,039	0,042
35	M08	0,103	0,006	0,024	0,035	0,062	0,066	0,029	0,066	0,034	0,147	0,038	0,021	0,041	0,027	0,044	0,051	0,05	0,075	0,039	0,042
36	M09	0,103	0,006	0,024	0,035	0,062	0,066	0,029	0,066	0,034	0,147	0,038	0,021	0,041	0,027	0,044	0,051	0,05	0,075	0,039	0,042
37	M10	0,103	0,006	0,024	0,035	0,062	0,066	0,029	0,066	0,034	0,147	0,038	0,021	0,041	0,027	0,044	0,051	0,05	0,075	0,039	0,042
38	O01	0,083	0,006	0,053	0,046	0,054	0,111	0,025	0,049	0,022	0,082	0,027	0,041	0,082	0,039	0,03	0,058	0,054	0,06	0,034	0,044
39	O02	0,083	0,006	0,053	0,046	0,054	0,111	0,025	0,049	0,022	0,082	0,027	0,041	0,082	0,039	0,03	0,058	0,054	0,06	0,034	0,044

Figure S3. The emission probabilities table

In the end, the user copies the column with probabilities, pastes them at Notepad or any other plain text editor and saves them on the file system.

Now that we have the initial transition and emission probabilities set up, we can create a Markov diagram using an appropriate software package.

The next step is to set up the model file using a plain text editor. The user creates the file according to the final state names, observed symbols and label names. Specifically, the user can name the model, define the symbol alphabet (Figure S3, see ESYM), the label alphabet (Figure S3, see PSYM), as well as, the specific states of the model (Figure S3, see STATE) and their corresponding labels (Figure S3, see PSYM). OSYM is a special “intermediate” condition between states and labels that defines the tied states, which are the states that share the same emission probabilities (in order to reduce the model parameters). Note that in the current configuration a state can correspond only to one label, but states with different labels can be tied together. The user may also define prior probabilities for each symbol or label if he/she wishes to (Figure S3, see PRIOR). The model configuration file is quite self-explanatory as one can see in Figure 3.

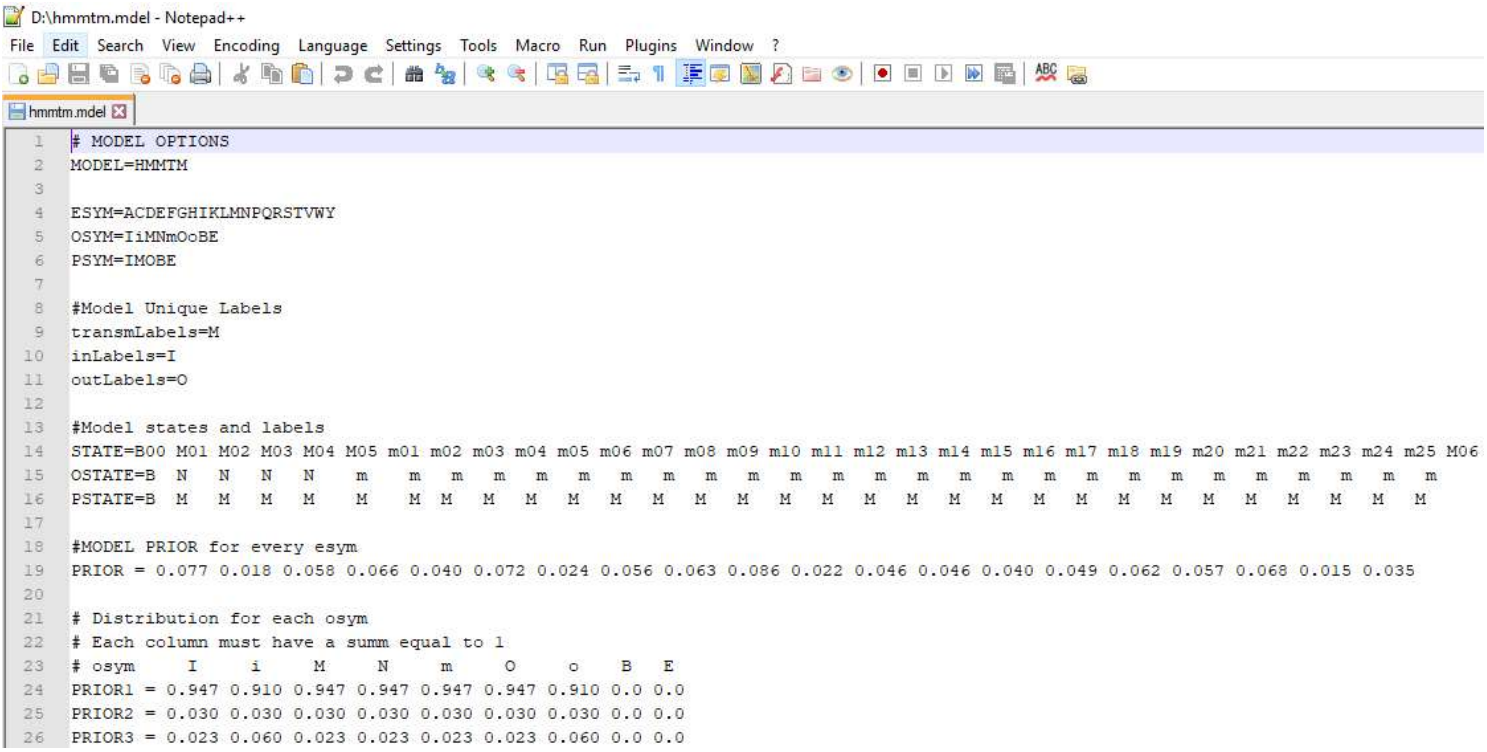


Figure S4. The model configuration file

The next and final step is to set up the configuration file using a plain text editor like Notepad. If the user defines only a few parameters and not all of them, the system will run using the default values as described in the manual. The user should be careful in this stage since all states defined in this file should correspond to the ones used in the transition and emission tables.

Training

While setting up the training parameters, the user may choose between the available training methods by using the respective configuration settings as shown below.

ML Configuration Settings

```
# TRAINING OPTIONS
RUN_CML=false
RUN_GRADIENT=false
HNN=false
ALLOW_BEGIN=true
ALLOW_END=true
RUN_ViterbiTraining=false
threshold=0.02
maxIter=200
```

CML Configuration Settings

```
# TRAINING OPTIONS
RUN_CML=true  (!important to enable Conditional Maximum Likelihood Learning)
RUN_GRADIENT=true
HNN=false
ALLOW_BEGIN=true
ALLOW_END=true
RUN_ViterbiTraining=false
threshold=0.02
```

```
maxIter=200
```

HNN Configuration Settings

```
# TRAINING OPTIONS
RUN_CML= true  (!important to enable Conditional Maximum Likelihood Learning)
RUN_GRADIENT=true
HNN= true  (!important to enable HNN extension)

#HNN OPTIONS
windowLeft=3
windowRigth=3
nhidden=3
ADD_GRAD=0.0
DECAY=0.001
#1: Sigmoid, 2: Sigmoid Modified, 3: Tanh
hiddenLayerFunction=2
```

The user may test different configurations before concluding to the most efficient one.

Decoding

Using the available algorithms, we can identify the most likely sequence of hidden states or the most probable labeling given the sequence of observations. Again, the user can choose which decoding algorithm will be used by editing the respective section of the configuration file as shown below.

Configuration Settings

```
# DECODING OPTIONS
VITERBI=true
NBEST=false
DYNAMIC=false
POSVIT=false
PLP=true
```

Since JUCHMME was originally developed to train and test transmembrane prediction models, some commonly used measures of accuracy are calculated and offered to the users. Specifically, the number of correctly predicted residues (Q), the segment overlap (SOV) measure and the number of correctly predicted topologies are calculated by JUCHMME and may be used to evaluate the new models.

```
VITERBI:

Q2:0.845   Qa:0.737   Qna:0.916 Pa:0.855   Pna:0.839  Qfas:0.827
Ca:0.673   SM:0.885   TP:596    FP:37      FN:122     Correct Top:10      Correct Ori:10      Avg SOV:0.787

PLP:

Q2:0.852   Qa:0.772   Qna:0.906 Pa:0.846   Pna:0.856  Qfas:0.839
Ca:0.690   SM:0.904   TP:627    FP:47      FN:91      Correct Top:14      Correct Ori:14      Avg SOV:0.807
```

To further highlight how simple it is to build a new model using JUCHMME, let's consider a follow-up scenario. In an attempt to further improve the reliability of alpha-helical transmembrane topology prediction, we chose to incorporate the post-translational modifications, phosphorylation and glycosylation, which are known to be compartment-specific and therefore the presence of a phosphorylation or glycosylation site in a transmembrane protein provides topological information. With

this in mind the model of HMM-TM was re-designed to include phosphorylation and glycosylation specific states as part of the intracellular and the extracellular sub-models respectively (Figure S5).

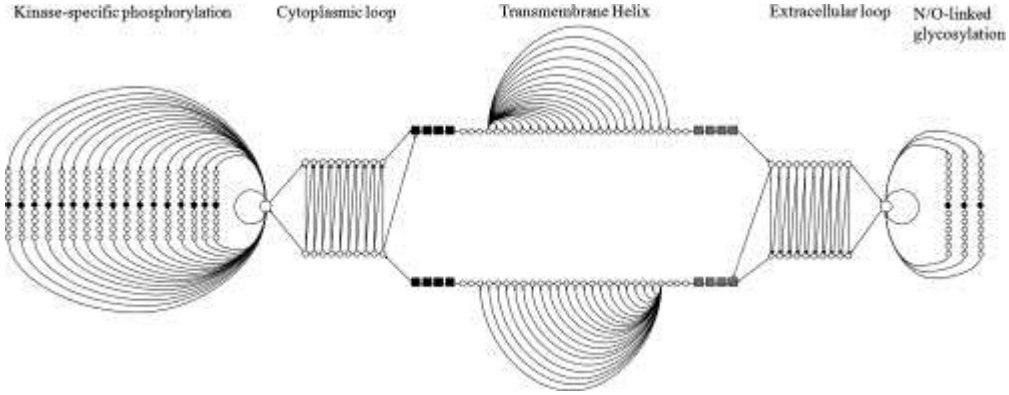


Figure S5. A schematic representation of the HMMpTM model's architecture

To build the new model, we modified the architecture of the existing HMM-TM model. As one can see in Figure S6, using JUCHMME it is easy to modify the architecture of existing models by editing the model configuration file. The OSYM, PSYM, STATE, OSTATE, PSTATE fields are changed accordingly. Also, both the transition and emission probabilities tables need to be modified using a spreadsheet based on the new model.

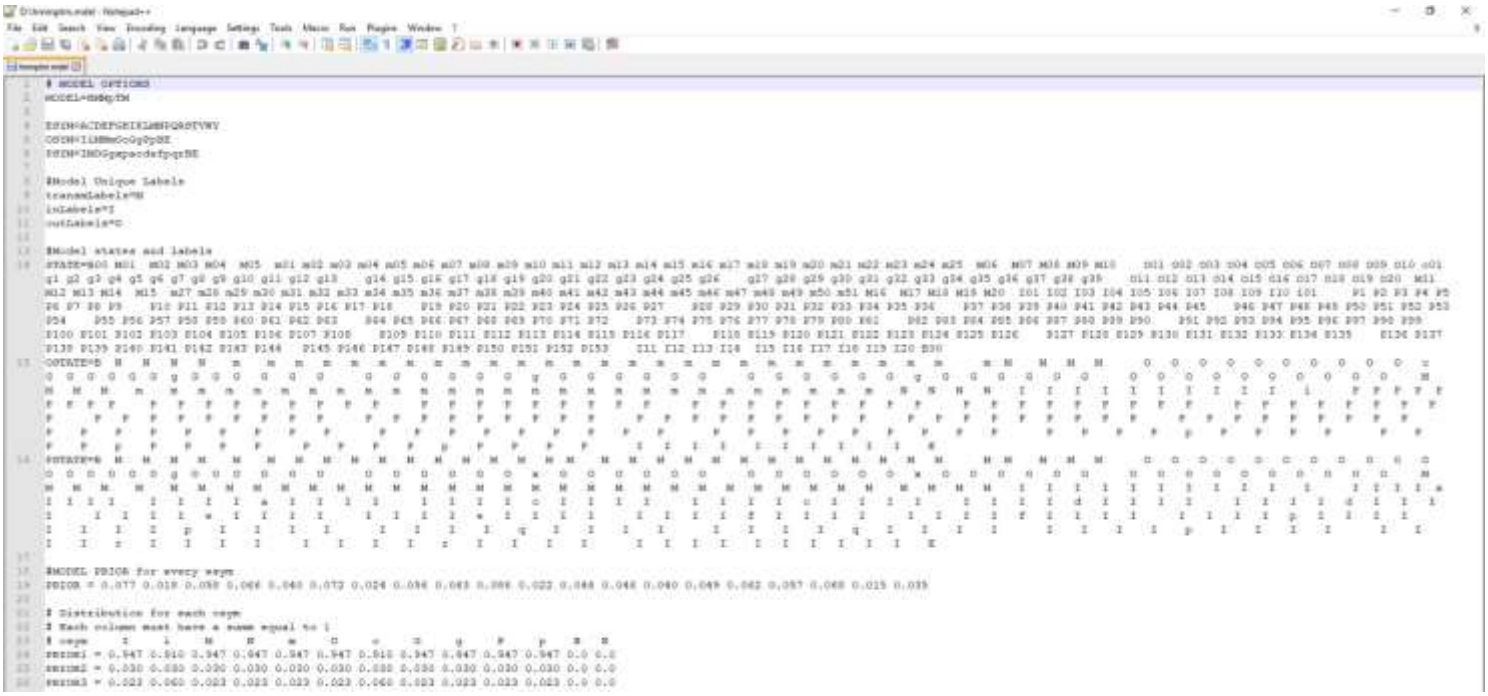


Figure S6. The HMMpTM model configuration file

PART B – Comparison of Speed

The following table and figures present the testing of traditional HMM Dishonest Casino model, CpG Island model [1], PRED-TMBB model [2] and PRED-TAT model [3]. The programs were compiled and run in Ubuntu Linux on an Intel Xeon E5-2660 10-core 2.00 GHz server with 32 GB RAM. Time was evaluated using the Unix time function and all values reported as the average of 5 runs. We used a varying number of input sequences (500, 1000, 3000, 5000 with 500 symbols per sequence) and we evaluated both Viterbi and Posterior decoding algorithms.

Table S1. Evaluation of JUCHMME and StochHMM in terms of speed using different models and varying number of input sequences

Model	Algorithm	Number of Sequences	JUCHMME (serial mode)	JUCHMME (Parallel mode)	StochHMM
Casino 2 states	**Viterbi**	**500**	2.150	1.941	0.181
		1000	3.536	3.373	0.277
		3000	10.303	10.148	3.918
		5000	15.864	15.749	5.328
	Posterior	**500**	2.499	1.951	0.255
		1000	4.417	3.223	0.493
		3000	11.799	10.073	3.169
		5000	17.882	16.072	5.527
CpG 8 states	**Viterbi**	**500**	3.905	2.073	0.239
		1000	7.003	3.526	0.365
		3000	18.546	10.365	3.798
		5000	29.584	16.891	6.357
	Posterior	**500**	5.508	2.464	0.506
		1000	10.992	3.782	0.854
		3000	29.165	10.456	5.294
		5000	47.550	16.126	8.134
PREDTMBB 62 states	**Viterbi**	**500**	10.675	2.990	30.236
		1000	20.171	5.256	61.383
		3000	61.666	14.155	174.204
		5000	95.548	21.080	295.080
	Posterior	**500**	13.254	3.238	50.437
		1000	25.920	5.900	100.950
		3000	75.943	17.261	299.394
		5000	120.879	29.387	511.065
PREDTAT 142 states	**Viterbi**	**500**	50.197	7.695	121.757
		1000	96.830	13.343	243.175

		3000	285.813	37.386	723.434
		5000	463.032	60.217	1198.107
	Posterior	**500**	67.891	8.838	214.173
		1000	131.387	16.439	427.095
		3000	370.474	45.670	1293.602
		5000	643.665	79.328	2199.807

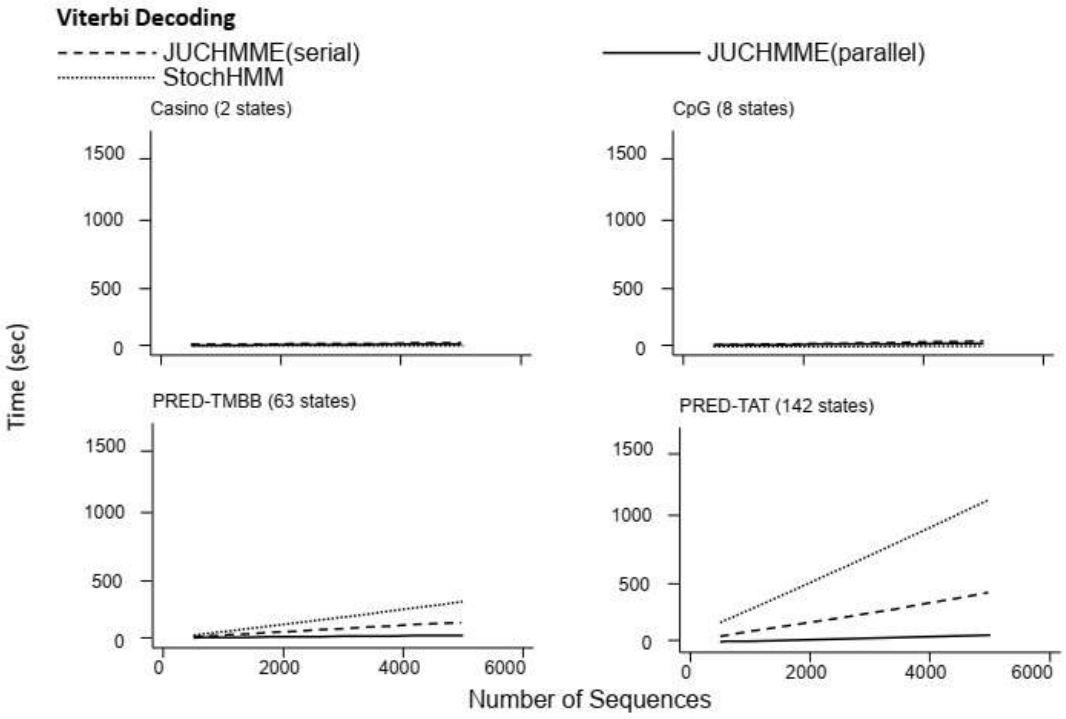


Figure S7. Run time comparison of JUCHMME and StochHMM using Viterbi Decoding

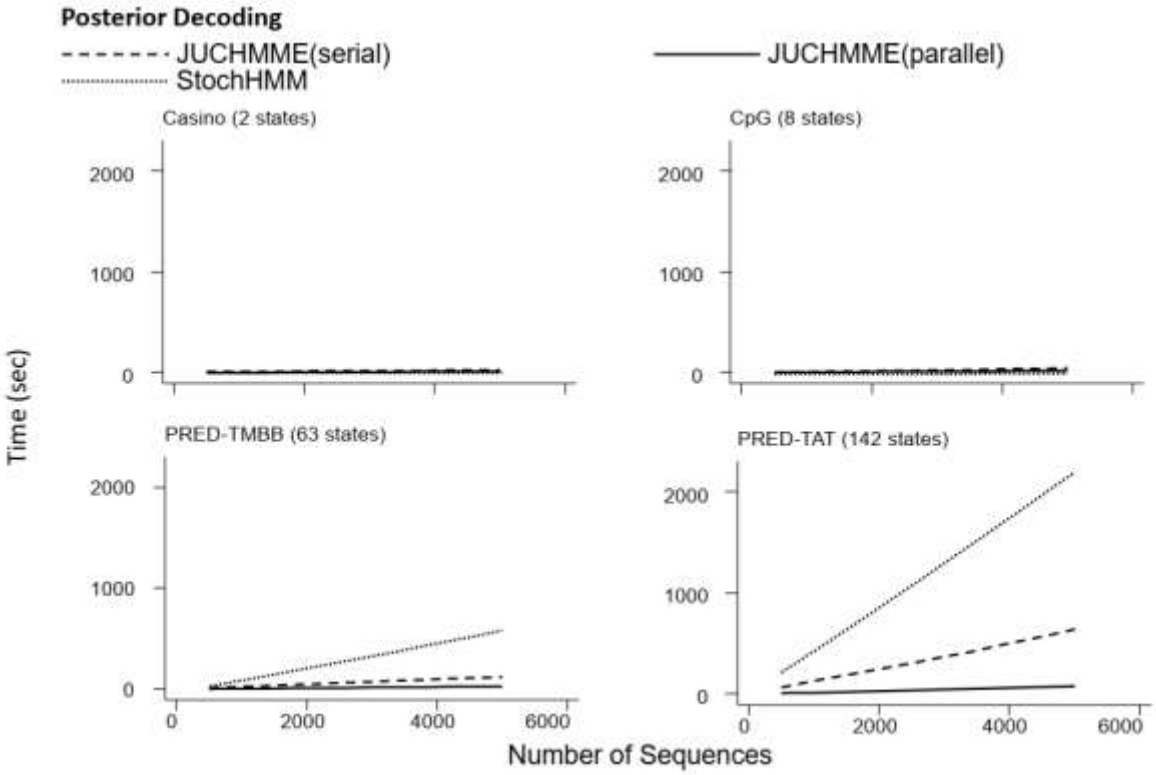


Figure S8. Run time comparison of JUCHMME and StochHMM using Posterior Decoding

PART C – Comparison of Features

Table S2. Comparison of features available in different HMM packages. All packages provide functionalities for standard HMMs trained with Baum-Welch algorithm and decoded with Posterior/Viterbi algorithms, so these are not listed.

Features	JUCHMME	PHMM [4]	MAMOT [5]	StochHMM [6]	HMMoC [7]	HMMConverter [8]	Modhmm [9]	Jahmm [10]	JaCHMM [11]	UMDHMM [12]
Types of models										
Class HMM for Labeled Sequences (CHMM) [13]	✓	✓					✓		✓	
Hidden Neural Networks (HNN) [14]	✓									
Higher order emissions or transitions (HOHMM, PHMM, HMMSDO) [15, 16] [17]	✓			✓	✓					
pair-HMMs [18]					✓	✓				
Generalized HMMs (GHMM) [19]					✓	✓				
Inhomogeneous chains					✓					
Continuous emissions				✓						
Silent states				✓			✓			
Multiple emissions, Joint emissions, Lexical transitions				✓						
Training Algorithms										
Conditional Maximum Likelihood (CML) training [20]	✓						✓			
Gradient-descent training (including RPROP) [21]	✓									
Viterbi Training [22]	✓	✓								
Semi-supervised learning (SSL) [23]	✓									
Linear-memory Baum-Welch training [24]						✓				
Linear-memory Viterbi training [25]						✓				
Decoding Algorithms										
Optimal Accuracy Posterior Decoding [26]	✓									
N-best decoding [27]	✓	✓		✓						
Posterior-Viterbi decoding [28]	✓	✓	✓							
Constrained decoding [29]	✓									
Stochastic decoding				✓						
Linear-memory posterior sampling						✓				
Hirschberg decoding algorithm [30]						✓				
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	✓			✓	✓	✓	✓			

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

References

1. Durbin, R., *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. 1998: Cambridge University Press.
2. Bagos, P.G., et al., *A Hidden Markov Model method, capable of predicting and discriminating beta-barrel outer membrane proteins.* BMC Bioinformatics, 2004. **5**: p. 29.
3. Bagos, P.G., et al., *Combined prediction of Tat and Sec signal peptides with hidden Markov models.* Bioinformatics, 2010. **26**(22): p. 2811-7.
4. Fariselli, P. *PHMM*. 2007; Available from: http://www.biocomp.unibo.it/piero/PHMM/.
5. Schütz, F. and M. Delorenzi, *MAMOT: hidden Markov modeling tool.* Bioinformatics, 2008. **24**(11): p. 1399-1400.
6. Lott, P.C. and I. Korf, *StochHMM: a flexible hidden Markov model tool and C++ library.* Bioinformatics, 2014. **30**(11): p. 1625-1626.
7. Lunter, G., *HMMoC—a compiler for hidden Markov models.* Bioinformatics, 2007. **23**(18): p. 2485-2487.
8. Lam, T.Y. and I.M. Meyer, *HMMCONVERTER 1.0: a toolbox for hidden Markov models.* Nucleic acids research, 2009. **37**(21): p. e139-e139.
9. Viklund, H. and A. Elofsson. *modhmm: Development of a modular HMM package for biological sequence analysis*. 2003 [cited 2019 4 Feb]; Available from: http://modhmm.sourceforge.net/.
10. Francois, J.-M., *Jahmm-An implementation of HMM in Java.* URL http://code. google. com/p/jahmm, 2006.
11. Ultes, S., et al. *Jachmm: a java-based conditioned hidden markov model library*. in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. 2013. IEEE.
12. Kanungo, T., *UMDHMMQ Hidden Markov Model Toolkit.*, in *Extended Finite State Models of Language*, K. A., Editor. 1999, Cambridge University Press.
13. Krogh, A. *Hidden Markov models for labeled sequences*. in *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*. 1994. IEEE.
14. Krogh, A. and S.K. Riis, *Hidden neural networks.* Neural Computation, 1999. **11**(2): p. 541-563.
15. Tamposis, I.A., et al., *Extending Hidden Markov Models to Allow Conditioning on Previous Observations.* Journal of Bioinformatics and Computational Biology, 2018.
16. Li, Y., *Hidden Markov models with states depending on observations.* Pattern Recogn. Lett., 2005. **26**(7): p. 977-984.
17. Forchhammer, S. and J. Rissanen, *Partially hidden Markov models.* IEEE Transactions on Information Theory, 1996. **42**(4): p. 1253-1256.
18. Arribas-Gil, A., G. Elisabeth, and M. Catherine, *Parameter Estimation in Pair-Hidden Markov Models.* Scandinavian Journal of Statistics, 2006. **33**(4): p. 651-671.
19. Majoros, W.H., et al., *Efficient decoding algorithms for generalized hidden Markov model gene finders.* BMC Bioinformatics, 2005. **6**: p. 16.
20. Krogh, A., *Two methods for improving performance of an HMM and their application for gene finding.* Proc Int Conf Intell Syst Mol Biol, 1997. **5**: p. 179-86.
21. Baldi, P. and Y. Chauvin, *Smooth on-line learning algorithms for hidden Markov models.* Neural Computation, 1994. **6**(2): p. 307-318.
22. Juang, B.-H. and L.R. Rabiner, *The segmental K-means algorithm for estimating parameters of hidden Markov models.* IEEE Transactions on acoustics, speech, and signal Processing, 1990. **38**(9): p. 1639-1641.
23. Tamposis, I.A., et al., *Semi-supervised learning of Hidden Markov Models for biological sequence analysis.* Bioinformatics, 2018: p. bty910-bty910.
24. Miklós, I. and I.M. Meyer, *A linear memory algorithm for Baum-Welch training.* BMC bioinformatics, 2005. **6**(1): p. 231.

25. Lam, T.Y. and I.M. Meyer, *Efficient parameter training for hidden Markov models using posterior sampling training and Viterbi training.* arXiv preprint arXiv:0909.0737, 2009.

26. Käll, L., A. Krogh, and E.L. Sonnhammer, *An HMM posterior decoder for sequence feature prediction that includes homology information.* Bioinformatics, 2005. **21**(suppl_1): p. i251-i257.

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

28. Fariselli, P., P.L. Martelli, and R. Casadio, *A new decoding algorithm for hidden Markov models improves the prediction of the topology of all-beta membrane proteins.* BMC bioinformatics, 2005. **6**(4): p. S12.

29. Bagos, P.G., T.D. Liakopoulos, and S.J. Hamodrakas, *Algorithms for incorporating prior topological information in HMMs: application to transmembrane proteins.* BMC bioinformatics, 2006. **7**(1): p. 189.

30. Hirschberg, D.S., *A linear space algorithm for computing maximal common subsequences.* Communications of the ACM, 1975. **18**(6): p. 341-343.