BIOINFORMATICS ORIGINAL PAPER Vol. 28 no. 5 2012, pages 636–642
doi:10.1093/bioinformatics/btr698

Sequence analysis

Advance Access publication January 3, 2012

Evaluating bacterial gene-finding HMM structures as probabilistic logic programs

Søren Mørk[1,*] and Ian Holmes[2]

[1]Department of Science, Systems and Models, Roskilde University, 4000 Roskilde, Denmark and [2]Department of Bioengineering, University of California, Berkeley, CA 94720, USA

Associate Editor: John Quackenbush

ABSTRACT

Motivation: Probabilistic logic programming offers a powerful way to describe and evaluate structured statistical models. To investigate the practicality of probabilistic logic programming for structure learning in bioinformatics, we undertook a simplified bacterial gene-finding benchmark in PRISM, a probabilistic dialect of Prolog.

Results: We evaluate Hidden Markov Model structures for bacterial protein-coding gene potential, including a simple null model structure, three structures based on existing bacterial gene finders and two novel model structures. We test standard versions as well as ADPH length modeling and three-state versions of the five model structures. The models are all represented as probabilistic logic programs and evaluated using the PRISM machine learning system in terms of statistical information criteria and gene-finding prediction accuracy, in two bacterial genomes. Neither of our implementations of the two currently most used model structures are best performing in terms of statistical information criteria or prediction performances, suggesting that better-fitting models might be achievable.

Availability: The source code of all PRISM models, data and additional scripts are freely available for download at: http://github.com/somork/codonhmm.

Contact: soer@ruc.dk

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

Received on July 7, 2011; revised on December 13, 2011; accepted on December 19, 2011

1 INTRODUCTION

Protein coding potential has long been recognized as the most important signal for automated gene finding (Fickett and Tung, 1992; Staden and McLachian, 1982; Staden, 1984). The introduction of Hidden Markov Models (HMMs) for gene finding by Krogh *et al.* (1994a) sparked the production of a large number of HMM-based, single-sequence gene finders that capture this signal and other signals (Besemer *et al.*, 2001; Burge and Karlin, 1997; Henderson *et al.*, 1997; Korf, 2004; Krogh, 1997; Kulp *et al.*, 1996; Larsen and Krogh, 2003; Lomsadze *et al.*, 2005; Lukashin and Borodovsky, 1998; Majoros *et al.*, 2003, 2004; Munch and Krogh, 2006; Reese *et al.*, 2000; Shmatkov *et al.*, 1999).

*To whom correspondence should be addressed.

Restricting our survey to the simplest case of bacterial gene-finding, the basic codon structure of protein-coding genes has so far been modeled using the following structures:

The Ecoparse gene finder introduced by Krogh *et al.* (1994a) is based on a standard HMM architecture with a silent state governing codon distributions via transitions to 64 separate three state submodels where each state of the codon submodels had fixed emissions of a single character. Stormo and Haussler (1994) introduced the Generalized Hidden Markov Model (GHMM), a type of HMM with duration offering the possibility of emissions of sequences rather than just characters from each state (Rabiner, 1989). Most single sequence *de novo* gene finders have since been based on GHMM's using either emissions of codons according to a three-periodic inhomogeneous Markov Chain (Besemer and Borodovsky, 1999; Borodovsky and McInich, 1993) or using higher ordered emissions, typically fifth ordered (Lukashin and Borodovsky, 1998) or variable ordered emissions (Delcher *et al.*, 1999; Salzberg *et al.*, 1998). The single character emitting HMM based gene finder models has subsequently been elaborated by also using higher ordered emissions (Krogh, 1997), as well as using Acyclic Discrete Phase type length modeling (Bobbio *et al.*, 2003), in Easygene and Agene (Larsen and Krogh, 2003; Munch and Krogh, 2006).

A large number of different HMM architectures have been developed during the recent decades including profile-HMMs (Krogh *et al.*, 1994b), pair-HMMs (Durbin *et al.*, 1998), input–output HMMs or transducers (Bradley and Holmes, 2007), factorial-HMMs (Ghahramani and Jordan, 1996) and mixed memory HMMs (Saul and Jordan, 1999). These models each combine different numbers of emitted sequences, hidden state chains, delete and insert states and different conditioning schemes for emission probabilities and transition probabilities. Employing as efficient model structures as possible for biological sequence analysis is paramount for coping with the vast amounts of sequence data currently being generated. The structure space of possible different combinations is large and exploring it for efficient models for biological sequence analysis is limited by the time-consuming development of dedicated machine learning algorithms and benchmarking procedures. Additionally, efforts to directly compare models are clouded by implementation differences and the heuristic fine-tunings developed through the decades that optimize the models in terms of the sequence analysis task at hand. To overcome these challenges, we use the probabilistic logic programming language and machine learning system PRISM. PRISM offers a generic representation of a large number of diverse model types that subsumes HMMs, SCFGs and Bayesian Networks. The PRISM machine learning system uses a general set

of algorithms to perform machine learning tasks for all models (Sato and Kameya, 2001). Using this approach, we can directly compare the performance of the underlying model structures of various gene finders without differences due to training procedures, the inclusion of different kinds of additional signals, pre/post-processing of data or other implementation differences. The generalized nature of PRISM programs allows the formulation of all types of model structures previously used as well as non-standard conditioning schemes (such as mixed memory HMM's) and the ability to export those conditioning schemes to other types of models [such as pair-HMMs, factorial-HMMs and Stochastic Context Free Grammars (Christiansen *et al.*, 2011)]. An alternative approach for using PRISM for evaluating gene finder programs based on artificially generated datasets is given in Christiansen and Dahmcke (2007).

2 APPROACH

To keep things as simple as possible, our preliminary benchmark uses the well-studied test case of bacterial gene finding. We use *Escherichia coli* as a test bed for our model structure comparisons, due to the availability of experimentally verified annotations and textbook gene structures. To test the robustness of the performance of the models, we have duplicated the experiments using a distantly related (and less well annotated) *Bacillus subtilis* genome. We evaluate the performance of the models based on learning statistics from learn sessions on the complete experimentally verified *E.coli* dataset as well as on prediction performances based on 5-fold cross-validation experiments for both genomes. In order to compare the performances of different gene finder model structures, we have implemented the first HMM-based gene finder model structure (ecoparse), the two most common current model structures that capture protein coding potential: a structure with fifth ordered emissions and one with inhomogeneous three-periodic Markov chain like emissions, as well as two new types of structures consisting of a model based on a amino acid hidden state sequence and a model based on a mixed memory HMM. In addition to the basic model structures, we have also included two straightforward extensions of the model structures: the first extension involves length modeling, an important feature of contemporary gene finders. Since the codon usage of highly expressed, normally expressed and laterally transferred/phage genes are known to differ (Blattner *et al.*, 1997), the second extension are three-state versions of the models that encode these three separate classes of genes.

3 METHODS

3.1 PRISM

PRISM is a logic programming language and machine learning system (Sato and Kameya, 2001). The earliest general-purpose engine for bioinformatics automata was implemented in Prolog by Searls and Murphy (1995); PRISM is effectively a probabilistic dialect of B-prolog, allowing a pure declarative approach that unifies the description of model and data. The distinguishing feature of PRISM is the build-in predicate *msw/2*, that represents discrete random variables. This allows Prolog's abducible facts to be assigned probabilistic parameters. Executing a query using a tabled variant of the prologs SLD resolution produces a tabled search tree—an explanation graph (corresponds to a dynamic programming matrix), enabling efficient parameter estimation using a generic expectation–maximization algorithm running on explanation graphs as reviewed in Sato (2009).

The usefulness of HMM's is based on a small set of algorithms for 'decoding' a sequence, i.e. calculating the most probable path Π^* and its probability $\mathbf{P}(\Pi^*)$(the Viterbi algorithm), the total probability of a sequence (the Forward algorithm or the Backward Algorithm) and the posterior probability that a specific character at a given position is emitted by a certain state (a combination of the Forward and Backward algorithms) and training models on data (the Baum–Welch algorithm) (Rabiner, 1989).

These algorithms are subsumed by the graphical EM algorithm that PRISM runs on the proof tree-like structures generated by a PRISM model, which means that as soon as one has formulated a PRISM model, one can parameterize it via EM from training data, calculate various probabilities of interest, use the parameterized model to decode data as well as generate simulated data from the parameterized model (Sato *et al.*, 2010). In addition to the standard EM algorithm and a Deterministic Annealing EM algorithm (Ueda and Nakano, 1998) that produces estimates of the maximum likelihood parameter values of a model, PRISM also offers a variational Bayes EM algorithm (Sato *et al.*, 2008) and a Deterministic Annealing Variational Bayes EM algorithm (Katahira *et al.*, 2008).

3.2 HMMs

An HMM is fully characterized by a set of transition probabilities between unobserved states and a set of emission probabilities of observed characters emitted from states. The structure of an HMM can be identified from the factorization scheme used to calculate the joint probability $P(O,\Pi)$ of an observed sequence O and a hidden state path Π. Given an observed sequence $O=x_1,x_2,...,x_n$ and a path $\Pi=\pi_1,\pi_2,...,\pi_n$, the transition probabilities a_{kl} are defined for position i and states k and l, as:

$$a_{kl}=\mathbf{P}(\pi_i=l\,|\,\pi_{i-1}=k), \tag{1}$$

and the emission probabilities $e_k(b)$ are defined for path Π, character x in position i, state k and character b, as:

$$e_k(b)=\mathbf{P}(x_i=b\,|\,\pi_i=k). \tag{2}$$

The joint probability of observing the sequence X and the path Π is:

$$\mathbf{P}(X,\Pi)=a_{0\pi_1}\prod_{i=1}^{n}e_{\pi_i}(x_i)a_{\pi_i\pi_{i+1}}. \tag{3}$$

The PRISM equivalent of a HMM is a collection of "values" predicates, declaring the values that can be taken by various random variables. These random variables represent the outcome spaces for probabilistic transitions and emissions. A recursive structure, with predicates for initiation and termination, connects these random variables to the state and emission sequences, specifying the partitioning scheme of the joint probability. The following is the complete source code of a 2 state DNA HMM (% marks comments)

```
%  parameters:
values(transition(state(begin)),[state(1),state(2),end]).
values(transition(state(1)),[state(1),state(2),end]).
values(transition(state(2)),[state(1),state(2),end]).
values(emission(state(1)),[a,c,g,t]).
values(emission(state(2)),[a,c,g,t]).

% initiation:
model(Observables) :-
   recursion(state(begin),Observables).

% recursion structure:
recursion(state(Si),[Xi|Rest]) :-
   msw(emission(state(Si)),Xi),
   msw(transition(state(Si)),NS),
   recursion(NS,Rest).

% termination:
recursion(end,[]).
```

This fully functional PRISM program can be parameterized, decoded to or sampled from using built-in PRISM functions. For example, to fit the model to a sequence 'aaagt', one could use learn([model([a,a,a,g,t])]).; to Viterbi-decode that same sequence, viterbif(model([a,c,g,t])).; and to sample

a sequence into samples once, get_samples(1,model(X),Samples). Since the outcome spaces are shared for transitions and emissions a more compact program would be the replacement of the parameters section with:

```
values(transition(state(_)),[state(1),state(2),end]).
values(emission(state(_)),[a,c,g,t]).
```

where the underscore denotes 'the anonymous variable' which is simply a placeholder for any logic variable. Note that except for the *values* and *msw* predicates that are special PRISM predicates, the names of the remaining predicates and variables are arbitrary and replacing them with single letters would result in a program with the exact same properties (consisting of a single line of code with 138 characters including 5 white spaces).

The elegant brevity of the source code for a HMM given as a PRISM program and its close structural resemblance with the model architecture makes it very easy to produce novel models via small changes, e.g. changing the emission probabilities to:

```
values(emission(_,_),[a,c,g,t]).
```

and the recursive formula to:

```
recursion(state(Si),P1,[Xi|Rest]): -
   msw(emission(state(Si),P1),Xi),
   msw(transition(state(Si)),NS),
   recursion(NS,Xi,Rest).
```

transforms a standard HMM into a second-order HMM with emissions conditioned on the present state and the previous emission, whereas changing to:

```
values(transition(_,_),[state(1),state(2)]).
```

with

```
recursion(state(Si),P1,[Xi|Rest]):-
   msw(emission(state(Si)),Xi),
   msw(transition(state(Si)),P1),NS),
   recursion(NS,Xi,Rest).
```

is a second order like mixed memory HMM with transitions conditioned on present state and previous emission. In the following, we will use such model extensions to develop novel model structures suitable for modeling protein coding potential.

3.3 Models of protein coding potential

- iid.psm is an IID like zeroth order emission HMM with a single state that emits over the alphabet of {acgt}. The model captures base frequencies and has a geometric length distribution. This type of model has traditionally been used as null model or as a model of intergenic sequences.
- mc5.psm is a fifth order Markov chain like HMM, with a single state and emissions conditioned on state and five previous emissions. The model captures di-codon preferences of coding regions.
- i3pmc.psm is a inhomogeneous three-periodic Markov chain with three sequential states with the emission of the first state conditioned on state, the emission of the second state conditioned on state and the previous emission, and the emission of the third state conditioned on state and the two previous emissions. All states emits from {acgt}.
- eco.psm the ecoparse architecture with three consecutive states with single symbol emissions for each codon combination and a single silent state to control the codon distribution.
- aa.psm has 20 hidden states corresponding to the 20 amino acids that emits synonymous codons. The hidden state path corresponds to the translated amino acid sequence of the encoded protein. aa models codon bias and amino acid sequence composition of the encoded protein simultaneously. The model assigns higher probability to synonymous than non-synonymous sequences, hence it is capable of attaining higher likelihood if the genes have more similar amino acid sequences than nucleotide sequences. The amino acid frequencies are governed by transitions from a silent state.
- mm.psm is a three state mixed memory HMM (Saul and Jordan, 1999) with higher ordered transitions, i.e. with transitions conditioned on the previous emissions. The model recognizes start and stop codons via making the transition probabilities conditional on the previous two and the present emission. Triplet emissions with first position conditioned on state, second position conditioned on state and previous emission and third emission conditioned on state and previous two emissions.

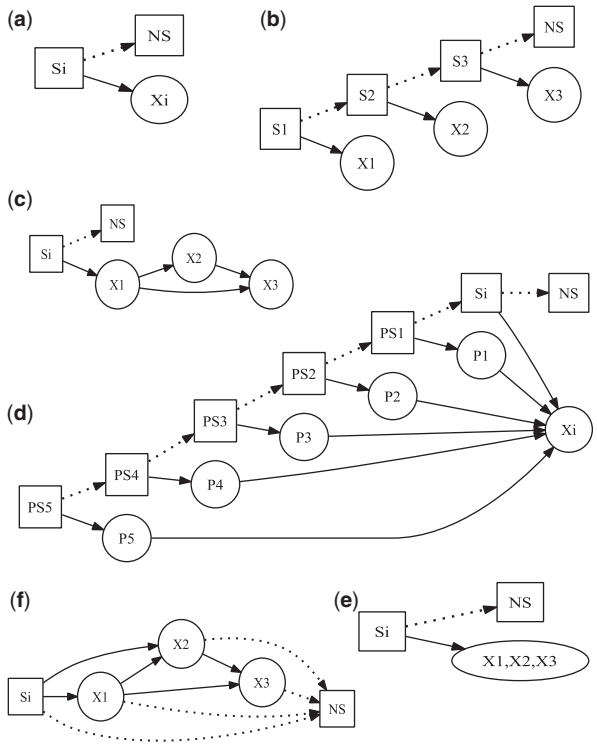

Fig. 1. Graphical representation of the conditioning schemes of the underlying structure of the models. (**a**) iid.psm; (**b**) eco.psm; (**c**) i3pmc.psm; (**d**) mc5.psm; (**e**) aa.psm; and (**f**) mm.psm. Squares represent the hidden State(S), Previous State(PS) or Next State(S); circles represent emissions (X) or past emissions (P). The dotted arrows are conditional transition probabilities and the full arrows are conditional emission probabilities.

All models (except iid.psm that is only used as null model) comes in three variants: a standard version as outlined above, an Acyclic Discrete Phase Type (ADPH) length modeling version, and a three-state version with three separate states that cannot transition to each other (e.g. once there is a transition to one of the states, the state path stays in that state until the end state) that corresponds to the three classes of bacterial genes: highly expressed genes, normally expressed genes and laterally transferred/phage genes. ADPH versions are created via adding six lines of code to the models and the three-state versions by adding two lines of code to the models. A graphical presentation of the standard version model structures are given in Figure 1.

4 EXPERIMENTS AND RESULTS

4.1 Training data

The *E.coli* genome used is the Refseq NC_000913.2 wild-type K12 strain MG1665 genome (Blattner *et al.*, 1997). The *E.coli* training set comprise the 2413 experimentally verified *E.coli* MG1665 genes from the ecocyc annotation (Keseler *et al.*, 2009) with canonical start and stop codons and no frameshift mutations. The complete verified ecocyc training set comprises a total of 2.487.654 nucleotides. The *Bacillus* training set used is the Refseq NC_000964 *B.subtilis* subsp. subtilis str. 168 (Kunst *et al.*, 1997). The complete *Bacillus* training set comprise 4155 genes with canonical start and stop codons and no frameshift mutations from the .gbk annotation (3.695.139 nt in total). The *E.coli* test set comprises all 68.826 potential ORFs from canonical start codon (ATG, GTG, TTG) to canonical stop codon (TAA, TAG, TGA) with a length over 60 nt (13.554.717 nt in total). The *Bacillus* test set comprises the equivalent 63.198 potential ORFs (11.438.079 nt in total). Training sets for all cross-validation studies were produced by randomly assigning the genes of the full training set into five subsets. The *E.coli* cross-validation training sets contains $\sim$ 500 genes each (see Supplementary Material for exact dataset sizes). Each cross-validation training sets were removed from the test sets and golden standards for the prediction performance evaluations. All sequence sets were converted to a format compatible with the PRISM models, i.e. a collection of e.g.: 'model([a,t,g,a,a,a,t,a,a]).'. Prediction performances were evaluated using the complete training sets as golden standards.

4.2 Training algorithms

All models are written as .psm files (available in for download). An additional prolog file *default_setting.pl* contains code for batch execution and settings of learning modes. The models were trained on the training datasets using *learn* with standard settings of VB-EM [see default_setting.pl and Sato *et al.* (2010) for instructions and other available options] with learning statistics and parameter values stored in separate files. Viterbi probabilities were calculated using *viterbi*.

4.3 Learning statistics

PRISM reports the following information relevant for model selection: the size of the explanation graph, the size of the table space used, the number of EM iterations, the total time of learning, the number of parameters, the number of parameter instances and the variational free energy values obtained after VB-EM learning. The variational free energy score is an approximation of log of the marginal likelihood (like the Bayesian Information Criterion), and is explained in detail in Sato *et al.* (2008). Table 1 shows a summary of the statistics from the learning sessions.

4.4 Prediction accuracy

Prediction accuracy was obtained using log-odds values obtained from the Viterbi probabilities of all potential open reading frames with a length over 60 nt for a given model and its null model (iid.psm). The log odds scores of the potential ORFs sequence were divided into a positive set (according to the golden standard) and a negative set. Confusion matrices and prediction performance metrics were calculated using all observed log odds scores of the positive set as thresholds. In order to ensure that a choice of log odds threshold is not biased in favor of any particular model, the prediction metrics example reported is the one that for each model maximizes the difference between true positive rate (TPR) and false positive rate (FPR), corresponding to the intercept of the ROC curve with the parallel to the no-discrimination line. Performance measures are all based on accurate prediction of stop codon position only, since discovery of novel genes is more interesting than correctly annotating the starting position of an already known gene. Prediction performance was determined by TPR (sensitivity/precision), FPR (Recall) and ROC curve area under the curve (AUC) value. AUC values were calculated from the pairs of TPRs and FPRs using the trapezoidal rule, which given the large number of points should be a relatively close approximation (DeLong *et al.*, 1988). Subsequently, the significance of the ranking of the models based on AUC were calculated through pairwise paired *t*-tests in R.

The receiver operator characteristics (ROC) curves from the *E.coli* cross-validation experiments is given for the standard versions of the models in Figure 2. (ROC curves for all the cross-validation experiments are available in Supplementary Materials.)

Table 2 gives the average ROC optimized prediction performances of the *E.coli* cross-validation experiments ± 1 SD as well as the average AUC values ± 1 SD. ROC optimized confusion matrices and prediction performances including match coefficient and Mathews correlation coefficient are available in Supplementary Materials. *P*-values of pairwise paired *t*-tests of the ranking of the *E.coli* AUC values are given in Table 3. *P*-values from pairwise paired *t*-tests of the prediction performance ranking: three-state version > standard version > adph version, using the *E.coli* AUC values are given in Table 4. The corresponding tables from the *Bacillus* cross-validation experiments are given as Tables 5–7. (*P*-values of pairwise paired *t*-tests of the ranking based on variational free energy scores are available in Supplementary Materials.)

Table 1. Learn Statistics of the different models trained on the entire *E.coli* verified ecocyc dataset

Model	Free parameters	Variational free energy	Graph nodes	Learn time
aa_s	507	-3.229×10^6	2.476×10^6	750
eco_s	67	-3.252×10^6	9.061×10^6	3024
i3pmc_s	64	-3.26×10^6	4.124×10^6	201
mc5_s	3139	-3.307×10^6	7.448×10^6	5
mm_s	127	-3.244×10^6	4.124×10^6	190
aa_3	1523	-3.211×10^6	7.428×10^6	4451
eco_3	203	-3.235×10^6	27.183×10^6	24 128
i3pmc_3	74	-3.257×10^6	12.371×10^6	492
mc5_3	9419	-3.307×10^6	22.343×10^6	25
mm_3	383	-3.228×10^6	12.371×10^6	528
aa_a	486	-3.245×10^6	16.433×10^6	3193
eco_a	67	-3.250×10^6	34.470×10^6	12 138
i3pmc_a	64	-3.26×10^6	24.646×10^6	597
mc5_a	3139	-3.306×10^6	49.572×10^6	15
mm_a	64	-3.26×10^6	24.646×10^6	621

Values reported includes the number of free parameters, the Variational Free Energy score after learning, the number of nodes in the explanation graph, and the learn time in minutes.

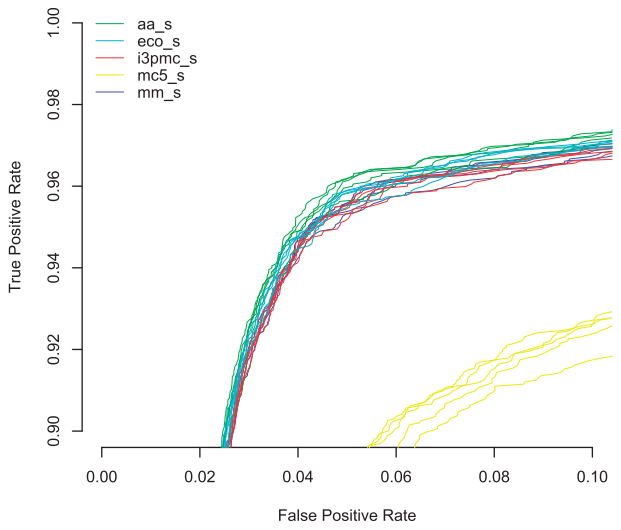

Fig. 2. *E.coli* cross-validation ROC curves for the standard models using thresholds over all log odds values in the positive set. Notice that only the area from 0.0–0.1 FPR and 0.9–1.0 TPR is shown.

Table 2. Prediction performances of the *E.coli* verified ecocyc cross-validation experiments

Model	TPR	FPR	AUC
aa	0.957 ± 0.004	0.045 ± 0.0	0.978 ± 0.001
eco	0.956 ± 0.002	0.047 ± 0.0	0.976 ± 0.001
i3pmc	0.95 ± 0.002	0.043 ± 0.0	0.975 ± 0.001
mc5	0.902 ± 0.002	0.062 ± 0.0	0.951 ± 0.003
mm	0.951 ± 0.001	0.044 ± 0.0	0.976 ± 0.001
aa_3	0.972 ± 0.002	0.042 ± 0.0	0.987 ± 0.0
eco_3	0.971 ± 0.007	0.046 ± 0.0	0.985 ± 0.002
i3pmc_3	0.959 ± 0.004	0.048 ± 0.0	0.979 ± 0.001
mc5_3	0.9 ± 0.004	0.064 ± 0.0	0.952 ± 0.002
mm_3	0.965 ± 0.003	0.046 ± 0.0	0.983 ± 0.002
aa_a	0.954 ± 0.003	0.044 ± 0.0	0.977 ± 0.001
eco_a	0.951 ± 0.005	0.044 ± 0.0	0.975 ± 0.001
i3pmc_a	0.947 ± 0.003	0.042 ± 0.0	0.974 ± 0.001
mc5_a	0.896 ± 0.005	0.064 ± 0.0	0.947 ± 0.002
mm_a	0.954 ± 0.005	0.045 ± 0.0	0.976 ± 0.001

Mean values ±1 SD.

5 RESULTS

The performance of the models both in terms of statistical fit ranked as per the variational free energy value after training on the complete training set and in terms of prediction performance evaluated via cross-validation experiments suggests the following general ranking: aa >= eco >= mm > i3pmc > mc5. Interestingly, the model structures that performed the worst were the the two currently most popular model structures: the fifth order Markov chain model (mc5) and the three-periodic inhomogeneous Markov chain model (i3pmc). With a single exception (mm), ADPH length modeling has an averse effect on the prediction performance, whereas using the three-state versions dramatically improved the performance of all models. Even though the difference in prediction performance for the standard models is less pronounced for the *B.subtilis* experiments, the general ranking of the models are still the same as for the *E.coli* experiments.

Table 3. *P*-values from pairwise paired *t*-tests of the ranking of the standard models of the AUC values of the ROC curves from the *E.coli* verified ecocyc cross-validation experiments

>	aa_ s/3/a	eco_ s/3/a	i3pmc_ s/3/a	mc5_ s/3/a	mm_ s/3/a
aa		2.048e-05	9.768e-06	2.095e-05	1.676e-05
eco	1		1.848e-06	3.612e-05	1.393e-05
i3pmc	1	1		4.526e-05	1
mc5	1	1	1		1
mm	1	1	1.586e-07	4.322e-05	
aa_3		0.0727	0.0001077	1.857e-06	0.005065
eco_3	0.9273		0.005022	2.35e-05	0.2019
i3pmc_3	0.9999	0.995		1.1e-05	0.9887
mc5_3	1	1	1		1
mm_3	0.995	0.7981	0.01135	2.120e-06	
aa_a		0.0001479	1.495e-05	3.621e-06	0.0001170
eco_a	0.9999		2.171e-06	3.826e-06	0.9988
i3pmc_a	1	1		4.879e-06	1
mc5_a	1	1	1		1
mm_a	0.9999	0.001164	6.347e-06	4.17e-06	

For each of the model versions, values are reported for comparisons within class only. Alternative hypothesis is that row entries are greater than column entries.

Table 4. *P*-values from pairwise paired *t*-tests of the ranking of the three-state, standard and ADPH models of the AUC values from the *E.coli* verified ecocyc cross-validation experiments

	aa	eco	i3pmc	mc5	mm
3> *s*	1.276e-05	0.001599	0.0004739	0.02713	0.0007062
s> *a*	0.01031	0.04478	0.03815	0.005672	0.5935

6 DISCUSSION

The findings of this study have been based on very architecturally simple models of protein coding potential only. We have used bacterial gene finding as a test environment favoring a thorough comparative test of the most prominent current single sequence bacterial gene finder model structures. Our results are not directly comparable to prediction performances of current state of the art gene finders in absolute terms, since we do not employ the heuristics that have been carefully developed through the last decades. However, as more and more data amasses we need as powerful models as possible. We propose that systematic testing of underlying model structures in a language where the model is foregrounded is a valuable approach to constructing reliable models for real-world applications.

7 CONCLUSION

We have developed and tested a number of both new and existing gene finder architectures for modeling protein coding potential in

Table 5. Prediction performances of the *Bacillus* cross-validation experiments

Model	TPR	FPR	AUC
aa	0.955 ± 0.002	0.056 ± 0.0	0.973 ± 0.001
eco	0.951 ± 0.006	0.053 ± 0.0	0.973 ± 0.001
i3pmc	0.953 ± 0.002	0.055 ± 0.0	0.972 ± 0.001
mc5	0.804 ± 0.006	0.087 ± 0.0	0.884 ± 0.004
mm	0.955 ± 0.004	0.055 ± 0.0	0.973 ± 0.001
aa_3	0.963 ± 0.004	0.052 ± 0.0	0.976 ± 0.001
eco_3	0.959 ± 0.002	0.051 ± 0.0	0.975 ± 0.001
i3pmc_3	0.953 ± 0.005	0.063 ± 0.0	0.971 ± 0.001
mc5_3	0.804 ± 0.005	0.087 ± 0.0	0.884 ± 0.003
mm_3	0.957 ± 0.001	0.051 ± 0.0	0.975 ± 0.001
aa_a	0.951 ± 0.003	0.054 ± 0.0	0.972 ± 0.001
eco_a	0.944 ± 0.004	0.048 ± 0.0	0.972 ± 0.001
i3pmc_a	0.949 ± 0.003	0.052 ± 0.0	0.972 ± 0.001
mc5_a	0.8 ± 0.009	0.091 ± 0.0	0.877 ± 0.004
mm_a	0.946 ± 0.004	0.049 ± 0.0	0.972 ± 0.001

Mean values ±1 SD.

Table 6. *P*-values from pairwise paired *t*-tests of the ranking of the AUC values of the standard models from the *Bacillus* cross-validation experiments

>	aa_ s/3/a	eco_ s/3/a	i3pmc_ s/3/a	mc5_ s/3/a	mm_ s/3/a
aa		0.0853	0.006177	2.187e-07	0.02411
eco	0.9147		0.0001165	1.691e-07	0.0003565
i3pmc	0.9938	0.9999		1.588e-07	1
mc5	1	1	1		1
mm	0.9759	0.9996	5.357e-05	1.641e-07	
aa_3		0.05268	5.516e-06	2.975e-07	2.137e-05
eco_3	0.9473		0.001716	1.603e-07	0.6412
i3pmc_3	1	0.9983		2.427e-07	1
mc5_3	1	1	1		1
mm_3	1	0.3588	1.72e-05	3.23e-07	
aa_a		0.2583	0.01180	2.349e-07	0.9564
eco_a	0.7417		0.0001139	1.833e-07	1
i3pmc_a	0.9882	0.9999		1.78e-07	1
mc5_a	1	1	1		1
mm_a	0.04364	6.121e-05	7.598e-05	1.905e-07	

For each of the model versions, values are reported for comparisons within class only. Alternative hypothesis is that row entries are greater than column entries.

a generalized framework that permits direct comparisons of the performance of the underlying model structures. Our approach demonstrates that there are very efficient model structures hidden in the vast structure space of non-standard HMMs. Specifically, the novel structures presented here seems promising candidates for advancing gene finding and additional biological sequence analysis tasks that rely on protein coding potential. Additionally, the general approach that we have used for exploring HMM structures for capturing protein coding potential is a promising route to exploring and discovering efficient models used for more complex biological sequence analysis tasks (such as RNA structure prediction or modeling viral genomes). Lastly, we have shown that the probabilistic logic programming language, PRISM, is a capable framework for the rapid prototyping and benchmarking of statistical models in bioinformatics, up to datasets the size of small (bacterial) genomes.

Table 7. *P*-values from pairwise paired *t*-tests of the ranking of the three-state, standard and ADPH models of the AUC values from the *Bacillus* cross-validation experiments

	aa	eco	i3pmc	mc5	mm
3> *s*	0.001636	0.01845	0.9704	0.8063	0.007597
s> *a*	2.219e-05	5.076e-05	3.881e-05	1.553e-05	0.0001157

Funding: 'Logic-statistic modeling and analysis of biological sequence data' funded by the NABIIT program under the Danish Strategic Research Council (to S.M.); NIH/NIGMS grant (R01-GM076705 to I.H.).

Conflict of Interest: none declared.

REFERENCES

Besemer,J. and Borodovsky,M. (1999) Heuristic approach to deriving models for gene finding. *Nucleic Acids Res.*, **27**, 3911–3920.

Besemer,J. *et al.* (2001) GeneMarkS: a self-training method for prediction of gene starts in microbial genomes. Implications for finding sequence motifs in regulatory regions. *Nucleic Acids Res.*, **29**, 2607–2618.

Blattner,F.R. *et al.* (1997) The complete genome sequence of Escherichia coli K-12. *Science*, **277**, 1453–1462.

Bobbio,A. *et al.* (2003) Acyclic discrete phase type distributions: properties and a parameter estimation algorithm. *Perform. Eval.*, **54**, 1–32.

Borodovsky,M. and McInich,J. (1993) GENMARK: parallel gene recognition for both DNA strands. *Comput. Chem.*, **17**, 123.

Bradley,R.K. and Holmes,I. (2007) Transducers: an emerging probabilistic framework for modeling indels on trees. *Bioinformatics*, **23**, 3258–3262.

Burge,C. and Karlin,S. (1997) Prediction of complete gene structures in human genomic dna. *J. Mol. Biol.*, **268**, 78–94.

Christiansen,H. and Dahmcke,C.M. (2007) A machine learning approach to test data generation: a case study in evaluation of gene finders. In Perner,P. (ed.) *Machine Learning and Data Mining in Pattern Recognition*. Springer Verlag, Berlin, Germany. pp. 741–755.

Christiansen,H. *et al.* (2011) Taming the zoo of discrete HMM subspecies & some of their relatives. In Bel-Enquix,G. (eds) *Biology, Computation and Linguistics, New Interdisciplinary Paradigms*, vol. 228 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands. pp. 28–42.

Delcher,A.L. *et al.* (1999) Improved microbial gene identification with GLIMMER. *Nucleic Acids Res.*, **27**, 4636–4641.

DeLong,E.R. *et al.* (1988) Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, **44**, 837–845.

Durbin,R. *et al.* (1998) *Biological Sequence Analysis*. Cambridge University Press, Cambridge, UK.

Fickett,J.W. and Tung,C.-S. (1992) Assessment of protein coding measures. *Nucleic Acids Res.*, **20**, 6441–6450.

Ghahramani,Z. and Jordan,M.I. (1996) Factorial hidden Markov models. *Mach. Learn.*, **29**, 245–273.

Henderson,J. *et al.* (1997) Finding genes in DNA with a Hidden Markov Model. *J. Comp. Biol.*, **4**, 127–141.

Katahira,K. *et al.* (2008) Deterministic annealing variant of variational Bayes method. *J. Phys. Conf.*, **95**, 012015.

Keseler,I.M. *et al.* (2009) EcoCyc: a comprehensive view of Escherichia coli biology. *Nucleic Acids Res.*, **37** (Suppl. 1), D464–D470.

Korf,I. (2004) Gene finding in novel genomes. *BMC Bioinformatics*, **5**, 59.

Krogh,A. (1997) Two methods for improving performance of an hmm and their application for gene finding. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **5**, 179–186.

Krogh,A. *et al.* (1994a) A hidden Markov model that finds genes in *E.coli* DNA. *Nucleic Acids Res.*, **22**, 4768–4778.

Krogh,A. *et al.* (1994b) Hidden Markov Models in computational biology : applications to protein modeling. *J. Mol. Biol.*, **235**, 1501–1531.

Kulp,D. *et al.* (1996) A generalized hidden markov model for the recognition of human genes in dna. In *Proceedings of the Fourth International Conference on Intelligent System for Molecular Biology*. AAAI Press, Menlo Park, CA, USA.

Kunst,F. *et al.* (1997) The complete genome sequence of the gram-positive bacterium bacillus subtilis. *Nature*, **390**, 249–256.

Larsen,T. and Krogh,A. (2003) Easygene - a prokaryotic gene finder that ranks orfs by statistical significance. *BMC Bioinformatics*, **4**, 21.

Lomsadze,A. *et al.* (2005) Gene identification in novel eukaryotic genomes by self-training algorithm. *Nucleic Acids Res.*, **33**, 6494–6506.

Lukashin,A. and Borodovsky,M. (1998) GeneMark.hmm: new solutions for gene finding. *Nucleic Acids Res.*, **26**, 1107–1115.

Majoros,W.H. *et al.* (2003) GlimmerM, Exonomy and Unveil: three ab initio eukaryotic genefinders. *Nucleic Acids Res.*, **31**, 3601–3604.

Majoros,W.H. *et al.* (2004) TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders. *Bioinformatics*, **20**, 2878–2879.

Munch,K. and Krogh,A. (2006) Automatic generation of gene finders for eukaryotic species. *BMC Bioinformatics*, **7**, 263.

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

Reese,M.G. *et al.* (2000) Genie, gene finding in Drosophila melanogaster. *Genome Res.*, **10**, 529–538.

Salzberg,S.L. *et al.* (1998) Microbial gene identification using interpolated Markov models. *Nucleic Acids Res.*, **26**, 544–548.

Sato,T. and Kameya,Y. (2001) Parameter learning of logic programs for symbolic-statistical modeling. *J. Artif. Intell. Res.*, **15**, 391–454.

Sato,T. *et al.* (2008) Variational Bayes via propositionalized probability computation in PRISM. *Ann. Math. Artif. Intell.*, **54**, 135–158.

Sato,T. *et al.* (2010). *PRISM User Manual (Version 2.0)*. at: http://sato-www.cs.titech.ac.jp/prism/download/prism20.pdf.

Sato,T. (2009) Generative modeling by PRISM. In Hill,P.M. and Warren,D.S. (eds) *ICLP*, vol. 5649 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, Germany. pp. 24–35.

Saul,L.K. and Jordan,M.I. (1999) Mixed memory markov models: Decomposing complex stochastic processes as mixtures of simpler ones. *Mach. Learn.*, **37**, 75–87.

Searls,D.B. and Murphy,K.P. (1995) Automata-theoretic models of mutation and alignment. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **3**, 341–349.

Shmatkov,A.M. *et al.* (1999) Finding prokaryotic genes by the frame-by-frame' algorithm: targeting gene starts and overlapping genes. *Bioinformatics*, **15**, 874–886.

Staden,R. and McLachian,A. (1982) Codon preference and its use in identifying protein coding regions in long DNA sequences. *Nucleic Acids Res.*, **10**, 141–156.

Staden,R. (1984) Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Res.*, **12**, 505–519.

Stormo,G.D. and Haussler,D. (1994) Optimally parsing a sequence into different classes based on multiple types of information. In *Proceedings of Second International Conference on Intelligent Systems for Molecular Biology*. AAAI/MIT Press, Menlo Park, CA, pp. 369–375.

Ueda,N. and Nakano,R. (1998) Deterministic annealing em algorithm. *Neural Netw.*, **11**, 271–282.