Evaluating bacterial gene-finding HMM structures as probabilistic logic programs

Søren Mørk¹,²* and Ian Holmes²

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

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 (Pickett and Tindall, 1993; Staden and McLachlan, 1985; 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., 1999; Krogh, 1997; Kulp et al., 2000; Larsen and Krogh, 2003; Konrad et al., 2003; Konrad et al., 1998; Larsen and Krogh, 2003; Konrad et al., 2003; Lukashin and Borodovsky, 1998; Majumder et al., 2000; Munch and Krogh, 2000; Reese et al., 2000; Shumakov et al., 2000). These gene finders have since been elaborated by also using higher ordered emissions, typically fifth ordered (Lukashin and Borodovsky, 1998) or variable ordered emissions (Fitcher et al., 1998; 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 (Huson et al., 2004), in Easygene and Agen (Larsen and Krogh, 1998; Munch and Krogh, 2000).

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 (Boden and Holme, 1995), factorial-HMMs (Ghahramani and Jordan, 1996) and mixed memory HMMs (Saul and Jordan, 1996). 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
Probabilistic logic programs

of algorithms to perform machine learning tasks for all models (Sato and Kameya, 2004). 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 part-HMMs, factorial-HMMs and Stochastic Context Free Grammars (Christiansen et al. 2001]). An alternative approach for using PRISM for evaluating gene finder programs based on artificially generated datasets is given in (Christiansen and Dahmen 2007).

2 APPROACH

To keep things as simple as possible, our preliminary benchmark that encode these three separate classes of genes. 1997), the second extension are three-state versions of the models laterally transferred/phage genes are known to differ (Blattner et al., 2002) and a model based on a mixed memory HMM. In addition to the basic model structures, we have also included two straightforward and a model based on mixed memory HMM. 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 performances of different gene finder model structures, 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 supplemented 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 (ecoparse), the two most common current model structures that capture protein coding potential: a structure with fifth ordered (ecoparse), the two most common current model structures that capture protein coding potential: a structure with fifth ordered

3 METHODS

3.1 PRISM

PRISM is a logic programming language and machine learning system (Sato and Kameya, 2004). The earliest general-purpose engine for bioinformatics automata was implemented in Prolog by (Seth 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 ms/2, that represents discrete random variables. This allows Prolog’s abducibility facts to be assigned probabilistic parameters. Executing a query using a tabulated variant of the prologs SLD resolution produces a tabled search tree—an explanation graph 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 P(Π*) (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. 2004). In addition to the standard EM algorithm and a Deterministic Annealing EM algorithm (Sato and Nakano, 2003) that produces estimates of the maximum likelihood parameter values of a model, PRISM also offers a variational Bayes EM algorithm (Sato et al., 2004) and a Deterministic Annealing Variational Bayes EM algorithm (Katahira et al., 2006).

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, Π) of an observed sequence O and a hidden state path Π. Given an observed sequence O=x1,x2,…,xn and a path Π=π1, π2,…,πn, the transition probabilities αi are defined for position i and states k and l, as:

$$a_{il} = P(π_i = l | π_{i-1} = k),$$ (1)

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

$$e_{il}(b) = P(π_i = b | π_{i-1} = k),$$ (2)

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

$$P(X, Π) = \prod_{i=1}^{n} e_{il}(x_i) \alpha_{il},$$ (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:
% value(transition(state(begin), [state(1)], state(2), end)).
% value(transition(state(1), [state(1)], state(2), end)).
% value(transition(state(2), [state(1)], state(2), end)).
% value(emission(state(1), [a,c,g,t])).
% value(emission(state(2), [a,c,g,t])).

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

% recursion structure:
% recursion(state(G1), [X|Rest]) :-
% new(emission(state(G1), X)),
% new(transition(state(G1), G2)),
% recursion(G2, 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 ‘aagc’, one could use learn(model(a,c,g,t)); to Viterbi-decode that same sequence, vieterbi(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:

\[
\text{values(transition(state([_]),[state([2],state([2],end)])),}
\]

\[
\text{values(emission(state([_]),[a,c,g,t]))}.
\]

where the underscore denotes ‘the anonymous variable’ which is simply a placeholder for any logic variable. Note that except for the \text{values} and \text{new} 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:

\[
\text{values(emission([_]),[a,c,g,t])}.
\]

and the recursive formula to:

\[
\text{recursion(state(Si),P1,[Xi|Rest]): -}
\]

\[
\text{msw(transition(state(Si)),NS),}
\]

\[
\text{msw(emission(state(Si),P1),Xi),}
\]

\[
\text{recursion(NS,Xi,Rest)}.
\]

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

\[
\text{values(transition([_],[state(1),state(2)]))}.
\]

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

- \text{id.psm} is an IID like zeroth order emission HMM with a single state that emits over the alphabet of \{a|c|g|t\}. 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.
- \text{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.
- \text{t3pmc.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 \{a|c|g|t\}.
- \text{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.
- \text{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 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.

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.

![Graphical representation of the conditioning schemes of the underlying structure of the models.](image-url)
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 [Keseler et al. 2009]. 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 [Kusti 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 from the complete set into five subsets. The *E.coli* cross-validation training sets contains ~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[]=t,a,g,a,a,ta,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. 2008] 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 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

<table>
<thead>
<tr>
<th>Model</th>
<th>Free parameters</th>
<th>Variational free energy</th>
<th>Graph nodes</th>
<th>Learn time</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa_s</td>
<td>507</td>
<td>−3.229 × 10^6</td>
<td>2.476 × 10^6</td>
<td>750</td>
</tr>
<tr>
<td>eco_s</td>
<td>67</td>
<td>−3.252 × 10^6</td>
<td>9.061 × 10^6</td>
<td>3024</td>
</tr>
<tr>
<td>ispmc_s</td>
<td>64</td>
<td>−3.06 × 10^6</td>
<td>4.124 × 10^6</td>
<td>201</td>
</tr>
<tr>
<td>mc5_s</td>
<td>313</td>
<td>−3.307 × 10^6</td>
<td>7.448 × 10^6</td>
<td>5</td>
</tr>
<tr>
<td>mm_s</td>
<td>127</td>
<td>−3.244 × 10^6</td>
<td>4.124 × 10^6</td>
<td>190</td>
</tr>
<tr>
<td>aa_3</td>
<td>1523</td>
<td>−3.211 × 10^6</td>
<td>7.428 × 10^6</td>
<td>4451</td>
</tr>
<tr>
<td>eco_3</td>
<td>203</td>
<td>−3.235 × 10^6</td>
<td>27.183 × 10^6</td>
<td>24128</td>
</tr>
<tr>
<td>ispmc_3</td>
<td>74</td>
<td>−3.257 × 10^6</td>
<td>12.371 × 10^6</td>
<td>492</td>
</tr>
<tr>
<td>mc5_3</td>
<td>9419</td>
<td>−3.307 × 10^6</td>
<td>22.341 × 10^6</td>
<td>25</td>
</tr>
<tr>
<td>mm_3</td>
<td>383</td>
<td>−3.228 × 10^6</td>
<td>12.371 × 10^6</td>
<td>528</td>
</tr>
<tr>
<td>aa_a</td>
<td>486</td>
<td>−3.245 × 10^6</td>
<td>16.433 × 10^6</td>
<td>3193</td>
</tr>
<tr>
<td>eco_a</td>
<td>67</td>
<td>−3.250 × 10^6</td>
<td>34.470 × 10^6</td>
<td>12138</td>
</tr>
<tr>
<td>ispmc_a</td>
<td>64</td>
<td>−3.26 × 10^6</td>
<td>24.646 × 10^6</td>
<td>597</td>
</tr>
<tr>
<td>mc5_a</td>
<td>3139</td>
<td>−3.306 × 10^6</td>
<td>49.572 × 10^6</td>
<td>15</td>
</tr>
<tr>
<td>mm_a</td>
<td>64</td>
<td>−3.26 × 10^6</td>
<td>24.646 × 10^6</td>
<td>621</td>
</tr>
</tbody>
</table>

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.
The performance of the models both in terms of statistical fit ranked via cross-validation experiments suggests the following general ranking: \( \text{aa} \gg \text{eco} \gg \text{mm} \gg \text{i3pmc} \gg \text{mc5} \). Interestingly, the model structures that performed the worst were 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 adverse 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 \( B.\text{subtilis} \) experiments, the general ranking of the models are still the same as for the \( E.\text{coli} \) experiments.

### 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 contrast to the approaches presented here, which are 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 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.
Alternative hypothesis is that row entries are greater than column entries.

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

<table>
<thead>
<tr>
<th>Model</th>
<th>TPR</th>
<th>FPR</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>0.955 ± 0.002</td>
<td>0.056 ± 0.002</td>
<td>0.973 ± 0.001</td>
</tr>
<tr>
<td>eco</td>
<td>0.951 ± 0.006</td>
<td>0.053 ± 0.006</td>
<td>0.973 ± 0.001</td>
</tr>
<tr>
<td>i3pmc</td>
<td>0.953 ± 0.002</td>
<td>0.055 ± 0.002</td>
<td>0.972 ± 0.001</td>
</tr>
<tr>
<td>mc5</td>
<td>0.804 ± 0.006</td>
<td>0.087 ± 0.006</td>
<td>0.884 ± 0.004</td>
</tr>
<tr>
<td>mm</td>
<td>0.955 ± 0.004</td>
<td>0.055 ± 0.004</td>
<td>0.973 ± 0.001</td>
</tr>
<tr>
<td>aa_3</td>
<td>0.963 ± 0.004</td>
<td>0.052 ± 0.004</td>
<td>0.976 ± 0.001</td>
</tr>
<tr>
<td>eco_3</td>
<td>0.959 ± 0.002</td>
<td>0.051 ± 0.002</td>
<td>0.975 ± 0.001</td>
</tr>
<tr>
<td>i3pmc_3</td>
<td>0.953 ± 0.005</td>
<td>0.063 ± 0.005</td>
<td>0.971 ± 0.001</td>
</tr>
<tr>
<td>mc5_3</td>
<td>1.000 ± 0.005</td>
<td>0.087 ± 0.005</td>
<td>0.884 ± 0.003</td>
</tr>
<tr>
<td>mm_3</td>
<td>0.957 ± 0.001</td>
<td>0.051 ± 0.001</td>
<td>0.975 ± 0.001</td>
</tr>
<tr>
<td>aa_a</td>
<td>0.951 ± 0.003</td>
<td>0.054 ± 0.003</td>
<td>0.972 ± 0.001</td>
</tr>
<tr>
<td>eco_a</td>
<td>0.944 ± 0.004</td>
<td>0.048 ± 0.004</td>
<td>0.972 ± 0.001</td>
</tr>
<tr>
<td>i3pmc_a</td>
<td>0.949 ± 0.003</td>
<td>0.052 ± 0.003</td>
<td>0.972 ± 0.001</td>
</tr>
<tr>
<td>mc5_a</td>
<td>0.8 ± 0.009</td>
<td>0.091 ± 0.009</td>
<td>0.877 ± 0.004</td>
</tr>
<tr>
<td>mm_a</td>
<td>0.946 ± 0.004</td>
<td>0.049 ± 0.004</td>
<td>0.972 ± 0.001</td>
</tr>
</tbody>
</table>

Mean values ± 1 SD.

Table 6. P-values from pairwise signed r-tests of the ranking of the AUC values of the standard models from the Bacillus cross-validation experiments

<table>
<thead>
<tr>
<th>Model</th>
<th>aa/s3a_i3pmc/s3a/mc5/s3a/mm/s3a</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>0.00853</td>
</tr>
<tr>
<td>eco</td>
<td>0.01477</td>
</tr>
<tr>
<td>i3pmc</td>
<td>0.9938</td>
</tr>
<tr>
<td>mc5</td>
<td>1</td>
</tr>
<tr>
<td>mm</td>
<td>0.9759</td>
</tr>
<tr>
<td>aa_3</td>
<td>0.05268</td>
</tr>
<tr>
<td>eco_3</td>
<td>0.09473</td>
</tr>
<tr>
<td>i3pmc_3</td>
<td>1</td>
</tr>
<tr>
<td>mc5_3</td>
<td>1</td>
</tr>
<tr>
<td>mm_3</td>
<td>1</td>
</tr>
<tr>
<td>aa_a</td>
<td>0.2583</td>
</tr>
<tr>
<td>eco_a</td>
<td>0.7417</td>
</tr>
<tr>
<td>i3pmc_a</td>
<td>0.9882</td>
</tr>
<tr>
<td>mc5_a</td>
<td>1</td>
</tr>
<tr>
<td>mm_a</td>
<td>0.04564</td>
</tr>
</tbody>
</table>

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 of the size of small (bacterial) genomes.

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

Conflict of Interest: none declared.

REFERENCES


