Evolving the Structure of Hidden Markov Models

Kyoung-Jae Won, Adam Prügel-Bennett, Anders Krogh

***Abstract*— A Genetic Algorithm (GA) is proposed for finding the structure of Hidden Markov Models (HMMs) used for biological sequence analysis. The GA is designed to preserve biologically meaningful building blocks. The search through the space of HMM structures is combined with optimisation of the emission and transition probabilities using the classic Baum-Welch algorithm. The system is tested on the problem of finding the promoter and coding region of *C. jejuni*. The resulting HMM has a superior discrimination ability to a hand-crafted model that has been published in the literature.**

***Index Terms*— HMM, Machine Learning, GA, biological sequence analysis.**

I. Introduction

HIDDEN Markov Models (HMMs) are probabilistic finite state machines used to find structures in sequential data. An HMM is defined by the set of states, the transition probabilities between states, and a table of emission probabilities associated with each state for all possible symbols that occur in the sequence. They were developed for use in speech applications, where they remain the dominant machine learning technique [1]. Over the past decade they have also become an important tool in bioinformatics [2]. Their great attraction is that they allow domain information to be built into the structure while allowing fine details to be learned from the data through adjusting the transition and emission probabilities. The design of HMM architecture has predominately been the province of the domain expert.

Automatic discovery of the HMM structure has some significant attractions. By eliminating the expert, it allows the data to 'speak for itself'. This opens the possibility of finding completely novel structures, unfettered by theoretical prejudice. In addition, automation of the design process allows many more structures to be tested than is possible if every structure has to be designed by hand. However, automation also comes at a cost. Imposing a structure on the HMM limits the possible outcome (probability distribution for sequences) that can be learned. In the language of machine learning imposes a structure reduces the learning capacity of the HMM. This has the benefit of reducing the estimation errors for the learned parameters given the limited amount of training data. Provided that the structure being imposed faithfully captures some biological constraints, it should not significantly increase the approximation error. As the generalisation error is the sum of the estimation and approximation error, imposing a meaningful structure on the HMM should give good generalisation performance. By automatically optimising the HMM architecture we are potentially throwing away their advantage over other machine learning techniques, namely their amenability to incorporate biological information in their structure. That is, we risk finding a model that 'over-fits' the data.

The aim of the research presented in this paper is to utilise the flexibility provided by Genetic Algorithms (GAs) to gain the advantage of automatic structure discovery while retaining some of the benefits of a hand designed architecture. That is, by choosing the representation and genetic operators, we attempt to bias the search towards biologically plausible HMM architectures. In addition, we can incorporate the Baum-Welch algorithm which is traditionally used to optimise the emission and transition probabilities as part of the GA. GAs appear to be very well suited to this application. The optimisation of HMM architectures is a discrete optimisation problem which is easy to implement in a GA. We can simultaneously optimise the continuous probabilities by hybridising the GA with Baum-Welch. Furthermore, GAs allow us to tailor the search operators so as to bias the search towards biologically plausible structures. This would be far harder to accomplish in a technique such as simulated annealing where the freedom to choose the move set is constrained by the need to maintain detailed-balance.

HMMs have received little attention from the Evolutionary Computing community. We are aware of only two other group that have used GA to optimise HMMs, neither has published in Evolutionary Computing journals [3], [4] (we discuss the relation of this work to our own in the next section). This is perhaps surprising considering the large amount of work on using GAs to optimise the structure of neural networks (see, for example, the review [5]) and more recent work on graphical models [6], [7]. We believe that GAs may be an important tool for evolving HMM architectures. Furthermore, there may be important lessons for GA about guiding search that can be learned from this application.

The rest of this paper is organised as follows. In the next section we give a brief review of Hidden Markov Models. In section III we describe our GA for optimising HMMs. Section IV describes experiments to test our GA. We conclude in section V.

II. Background

A. Hidden Markov Models

An HMM is a learning machine that assigns a probability to an observed sequence of symbols. A sequence, $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$, consist of symbols x_t belonging to some alphabet, $\mathcal{S}$. In biological sequence analysis the alphabet might be, for example, the set of four possible DNA base pairs (bp), 'A', 'C', 'G' and 'T' or the set of 20 amino acids that are the building blocks of nearly all proteins. We denote the set of parameters that define an HMM by Θ. Given a sequence $\boldsymbol{x}$,

School of Electronic and Computer Science, University of Southampton, UK.

an HMM returns a 'probability' $\mathbb{P}(\boldsymbol{x}|\Theta)$, where

$$\sum_{\boldsymbol{x}\in\mathcal{S}^T} \mathbb{P}(\boldsymbol{x}|\Theta) = 1$$

(we use $\mathcal{S}^T$ to denote the set of all sequences of length T). $\mathbb{P}(\boldsymbol{x}|\Theta)$ is known as the *likelihood* of the sequence. To understand the meaning of this probability, we can imagine some process of interest, $\mathcal{P}$, (e.g. molecular evolution) that generates a set of sequences of length T with probability $\mathbb{P}(\boldsymbol{x}|\mathcal{P})$ then our aim is to find a HMM, Θ, such that $\mathbb{P}(\boldsymbol{x}|\Theta)$ is as close as possible to $\mathbb{P}(\boldsymbol{x}|\mathcal{P})$. Of course, we usually do not know $\mathcal{P}$. Rather we have some training examples consisting of a set of sequences. We then choose the HMM so that these training examples have a high likelihood.

Often we are interested in how well a sequence fits the model. To do this we consider the log-odds of a sequence

$$\text{log-odd for sequence } = \log\left(\frac{\mathbb{P}(x|\Theta)}{|\mathcal{S}|^{-T}}\right) \tag{1}$$

where $|\mathcal{S}|$ denotes the cardinality of the set of symbols $\mathcal{S}$. The log-odds is positive if the sequence is more likely than average. To use an HMM for classification we can set a threshold for the log-odds of a new sequence to belong to the same class as the training data.

The HMM is a probabilistic finite state machine which can be represented as a directed graph in which the nodes correspond to states and the edges correspond to possible transitions between states. A transition probability is associated with each edge with the constraint that the sum of the transition probabilities for edges exiting a node must sum to one. In addition, there is an emission probability table associated with each state which encodes the probability of each symbol $a \in \mathcal{S}$ being 'emitted' given that the machine is in that state. We define one state as the start state, which does not emit a symbol, but has transitions to the other states. To compute probabilities from our HMM we consider an 'event' to be a *path* through the graph where we emit a symbol every time we enter a state. The probability of the event is equal to the probability of the path times the probability of emitting that sequence of symbols given that we have taken that path through the finite state machine. The probability of a sequence is then found by averaging over all paths.

To formalise the HMM, we denote the set of states by $\mathcal{Q}$, the transition probabilities from state i to j by $a_{i\,j}$, and the probability of emitting a symbol a given that we are in a state i by $e_i(a)$. Let $\boldsymbol{q} = (q_1, q_2, \ldots, q_T)$ be a sequence of states, then the likelihood of a sequence $\boldsymbol{x}$ is given by

$$\mathbb{P}(x|\Theta) = \sum_{\boldsymbol{q}\in\mathcal{Q}^T} \mathbb{P}(x, q|\Theta) \tag{2}$$

$$\mathbb{P}(x, q|\Theta) = \prod_{t=1}^{T} a_{q_{t-1}\,q_t}\, e_{q_t}(x_t). \tag{3}$$

Naively, the computation of likelihood seems to grow exponentially with the length of the sequence. However, all the computations we need can be computed efficiently using dynamic programming techniques. We can compute the likelihood using the 'forward' algorithm. The forward variable $\alpha_t(i)$ is defined as

$$\alpha_t(i) = \mathbb{P}(x_1, \cdots, x_t, q_t = i|\Theta). \tag{4}$$

Starting from $\alpha_1(i) = a_{01}x_1$, we can find $\alpha_t(i)$ for all states $i \in \mathcal{Q}$ for successive times $1 \le t \le T$ using the recursion

$$\alpha_t(i) = e_i(x_t) \sum_{j\in\mathcal{Q}} a_{ji}\, \alpha_{t-1}(j). \tag{5}$$

This follows from the Markovian nature of the model. When we have found $\alpha_T(i)$ we can compute the likelihood by marginalising out the final state

$$\mathbb{P}(\boldsymbol{x}|\Theta) = \sum_{i\in\mathcal{Q}} \alpha_T(i). \tag{6}$$

There exists an analogous backward algorithm than can also be used to compute the likelihood. We define the backward variable $\beta_t(i)$ to be the probability of matching the sequence $x_{t+1}, \cdots, x_T$ given that we are in state i at time t

$$\beta_t(i) = P(x_{t+1}, \cdots, x_T|q_t = i, \Theta). \tag{7}$$

Again this can be obtained recursively using

$$\beta_t(i) = \sum_{j\in\mathcal{Q}} a_{ij} e_j(x_{t+1}) \beta_{t+1}(j) \tag{8}$$

with initial condition $\beta_T(i) = 1$ for all $i \in \mathcal{Q}$. The likelihood is given by

$$\mathbb{P}(\boldsymbol{x}|\Theta) = \sum_{i\in\mathcal{Q}} a_{0,i}\, e_i(x_1)\, \beta_1(i).$$

More importantly, the backward variable can be used in combination with the forward variable to compute important quantities needed for parameter estimation.

B. HMM parameter estimation

The continuous parameters of the HMM can be estimated so as to maximise the model likelihood $P(x|\Theta)$ on the given sequences. This is achieved using the Baum-Welch algorithm which is an example of a Expectation Maximisation (EM) algorithm [8]. Because the state sequences are not directly observable, statistical information is used to adjust the parameters. The model parameters that maximise the likelihood value are calculated iteratively. The update rule of the transition probabilities are

$$a_{ij} = \frac{n_{ij}}{\sum_{j'\in\mathcal{Q}} n_{ij'}}. \tag{9}$$

where n_{ij} is the number of transitions from state i to state j summed over the sequence, that is,

$$n_{ij} = \sum_{t=1}^{T} n_{ij}(t). \tag{10}$$

The update rule of the emission probabilities can be obtained as

$$e_i(a) = \frac{n_i(a)}{\sum_{a'\in\mathcal{S}} n_i(a')} \tag{11}$$

where $n_i(a)$ is the number of times the symbol a is emitted when in state i. Equations (9) and (11) are self-consistent equations which are satisfied when the likelihood for the training data is local maximal. We satisfy the equation by iteratively updating a_{ij} and $e_i(a)$ according to the observed values for n_{ij} and $n_i(a)$ which are computed using the forward and backward algorithm (see below). If we train on a limited amount of data this estimation of the emission probabilities is liable to over-fit the data. For example, we may not has seen some symbol a emitted at time t in our training data. If we build an HMM with a zero probability of emitting a symbol at this time we would perfectly fit our training data, but we may be reading more into a finite sample of data that we should be. To avoid excessive over-fitting we can add some 'pseudo-counts', α_{ij} to our estimate for n_{ij}, and $\alpha_i(a)$ to our estimate for $n_i(a)$. Our estimated transition and emission probabilities are given by

$$a_{ij} = \frac{n_{ij} + \alpha_{ij}}{\sum_{j' \in \mathcal{Q}(} n_{ij'} + \alpha_{ij'})} \tag{12}$$

$$e_i(a) = \frac{n_i(a) + \alpha_a}{\sum_{a' \in \mathcal{S}} (n_i(a') + \alpha_{a'})}. \tag{13}$$

Although, this appears to be an *ad hoc* fix, it can be motivated from a Bayesian perspective. We can consider the training set to be a sample from a multinomial distribution and assume a Dirichlet distribution for the prior probability, then the pseudo-counts drop out as the coefficients of the prior [9].

In the Baum-Welch algorithm unknown transition and emission frequencies are replaced by their expected values and the parameters re-estimated using the forward and the backward variables. The parameters are updated according to

$$\begin{aligned} n_{ij}(t) &= \mathbb{P}\left(q_{t-1} = i, q_t = j | x, \Theta\right) \\ &= \frac{\alpha_{t-1}(i) a_{ij} e_j(x_t) \beta_t(j)}{\mathbb{P}\left(x|\Theta\right)} \end{aligned} \tag{14}$$

$$\begin{aligned} n_i(t) &= \mathbb{P}\left(q_t = i | x, \Theta\right) \\ &= \frac{\alpha_t(i)\beta_t(i)}{\mathbb{P}\left(x|\Theta\right)}. \end{aligned} \tag{15}$$

The Baum-Welch algorithm acts as a local search algorithm and is liable to become trapped at a local optimum. Furthermore, the HMM can easily over-fit the training data because of the large number of free parameters. For more detail on the training HMMs see *e.g.* [1], [10].

There are several ways of finding a state sequence in a given observation sequence. The posterior decoding method is to choose the states which are individually most likely

$$q_t^* = \arg\max \mathbb{P}\left(q_t = i | x, \Theta\right). \tag{16}$$

From (4) and (7) we can get

$$\arg\max \mathbb{P}\left(q_t = i | x, \Theta\right) = \frac{\mathbb{P}\left(x, q_t = i | \Theta\right)}{\mathbb{P}\left(x|\Theta\right)} \tag{17}$$

This can be evaluated using the forward and backward algorithms.

C. Topologies of HMMs

The effectiveness of an HMM will depend on the number of states and the structure of the graph connecting the states. Traditionally, prior knowledge about a problem is encoded in this structure. In choosing a structure there is a trade-off between simplicity and complexity. A too simple model is unlikely to be able to generate the data set with a high likelihood, while a too complex model will easily learn the data set, but is unlikely to generalise well. The most complex structures are completely connected graphs, while the simplest are linear chains. Fig. 1 shows several types of HMMs. In these examples, we only allow one symbol to be emitted from each state (in general, one state could emit any symbol form the alphabet, $\mathcal{S}$ with a probability given by the table of emission probabilities). The fully connected model, Fig. 1(a), can generate any sequence. The self-loop model of Fig. 1(b) can generate sequences such as AAATTTGCCC, while the left-right model can generate the same sequence as the self-loop, but can also jump states to produces sequences such as AATTCCC.

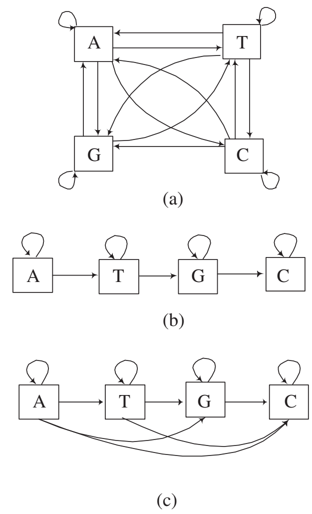

Fig. 1. Several types of HMM topologies; (a) ergodic model (b) self loop model (c) left-right model

The transition probabilities are usually learned from data. Changing the transition probabilities in Fig. 1 would increase the likelihood of being in a particular state. The table of emission probabilities can also be learned from data. It is also possible to encode domain knowledge by restricting the symbols that can be emitted from a particular state, however, in this paper we never make such restrictions. An additional trick commonly used to produce a simpler model is to 'tie' states so they have the same emission or transition probabilities [11]. States are tied if we believe that they model similar parts of a sequence. This reduces the number of free parameters that need to be learned and so reduces the problem of over-fitting.

D. Previous Work

There has been relatively little work in find the HMM structure without any prior knowledge. In speech recognition, state merging methods have been used to find a compact HMM structure [12]. In this work, an HMM architecture is built from a large initial model which is shrunk by merging states. In the bioinformatics field, there has been an attempt to find HMM structures using a local search that inserts or deletes states [13]. They used statistical information about the states to decide the point to insert new states. These attempts have not been very successful for designing the architecture for complex problems. The use of GAs to find an HMM structure has been tried previous in the domain of speech recognition [4]. They considered a 5-state model and evolved a strings which represented the transitions and emissions of an HMM. We are aware of only one other group that has used GAs for optimising HMM structures for biological sequence analysis [3]. They used GAs to insert and delete states and swap transitions. They could deal with complex biological sequences but the model produced is hard to interpret because they allowed unlimited transitions between states. In a previous paper, we have attempted to build a human interpretable HMM structure using a GA [14]. We used a similar block structure to that described here, but the linking between blocks was very limited. The structures we found were too limited for more general applications.

HMMs are, in fact, a special sub-class of Graphical Models [15]. GAs have been used to find the structure for graphical models (see, for example, [6], [7]). However, the structures being learned in graphical models is the dependency graph. The dependency graph for an HMM is shown in Fig. 2. The structure of the HMM determines the probability model between the states rather than the dependency graph. Nevertheless, both HMMs and Graphical Models aim to find a graph topology that accurately represents a probabilistic process. Consequently, there may be crossover of ideas between these two application domains.

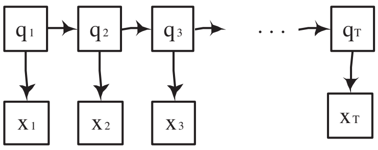

Fig. 2. An HMM represented a Dynamical Bayesian Network.

III. Block-HMM

A. Biological Block Model

To constrain the search of HMM topologies to biologically meaningful structures we represent the HMM structure as a number of blocks. The blocks we consider are one of three basic structures that are frequently used in biological sequence analysis. These are: linear, self-loop and forward blocks. The self-loop and forward block can be either tied (we follow the convention of shading tied blocks) or untied. That is, all the emission and transition probabilities are set equal. A forward block can have a transition from the first block to the n last blocks in the chain. Examples of these block structures are shown in Fig. 3.

The linear block can model a particular pattern of a sequence, the, so called, conserved regions. The TATA box and the Ribosomal Binding Site which show patterns of TAtAAT

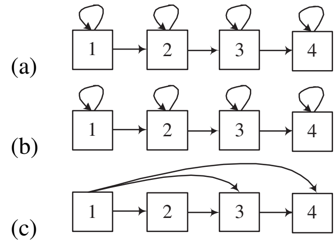

Fig. 3. HMM blocks that compose the whole HMM structure. (a) linear block (b) self-loop model (tying is optional) (c) forward-jump block.

and AAGGA, respectively, can be expressed with the linear block. Tying can be switched on and off in the self loop and forward blocks. The self-loop can be used to model a sequence of any length while the forward jump block can be used to represent varying length subsequences up to some fixed length.

The blocks are fully linked together to form the whole HMM architecture. This is illustrated in Fig. 4. The last state of a block has transitions to the first states of all the blocks. The resulting HMM can be thought of as a fully connected graph consisting of 'super-vertices' made up of blocks whose internal states are not fully connected. We call this structure a block-HMM. Special patterns like periodic signals can be generated with the path between blocks.

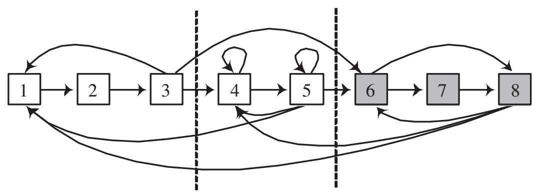

Fig. 4. An example of HMM composed of blocks (block-HMM). Three blocks are used in this model and all the blocks are fully connected to each other.

B. Genetic Operators for the block-HMM

Each block is represented by a pair. The first element defines the length of the block while the second element gives the type (a, b or c corresponding to linear, self-loop or forward-jump block respectively). The type also specifies whether the nodes are tied or untied (t or u) and in the case of forward blocks the number of forward connections. The full HMM is represented as a string on pairs as shown in Fig. 5. For example, the HMM in Fig. 4 is represented as '((3,a),(2,bu),(3,ct1))'. As the blocks are equivalent in their connectivity their is no information in the ordering of the blocks.

length of block 1	type of block 1	length of block 2	type of block 2	. . .	. . .	length of block n	type of block n

Fig. 5. The string representation of a block-HMM. The information on the lengths and the types of the blocks are stored.

In crossover, two parent strings are chosen at random. Some number of randomly chosen blocks are then swapped to create two children. The children then replace the parents. When we swap blocks the transition probability leaving the block are kept unchanged. Since the position of the blocks does not carry any meaning, we do not impose any constraint on which blocks are swapped. The number of blocks is kept fixed, however, as the blocks can have variable lengths, the number of states is not fixed. We also allow block consisting of no states (zero blocks), which effectively allows us to have a variable number of blocks up to some maximum. The evolution of a variable size structure is similar to the situation common in many applications of Genetic Programming.

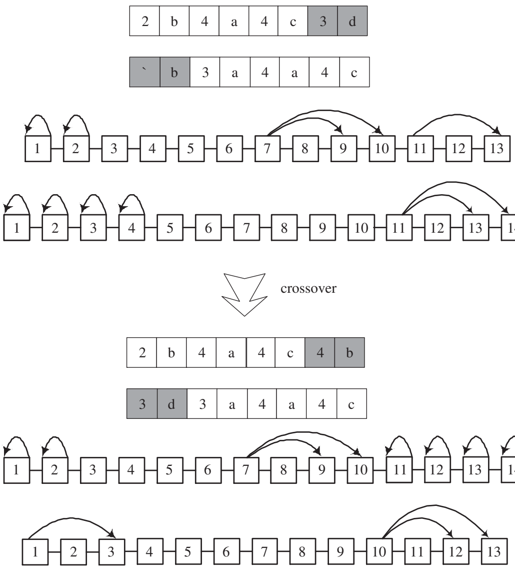

Fig. 6. Crossover in block-HMM. The crossover swaps the HMM states without breaking the property of HMM blocks

Fig. 6 shows an example of the crossover scheme. The last block of the first child crosses with the second block of the second child. To simplify the diagram, transitions between blocks are not shown here. Under the crossover scheme the properties of the interpretable blocks are not broken. This allows us to exchanging meaningful blocks without causing too much disruption.

Mutation can take place in any block of the HMM. There are a variety of different mutations that we allow. Mutation can change the length of a block. For forward-jump block mutations can change the number of transitions. For example, in the case of a 4-state forward-jump block, there are 6 different type of mutations possible. These are illustrated in Fig. 7. The mutation can add (Fig. 7(a)) or delete (Fig. 7(b)) a transition inside a block. To prevent losing the property of the block, deletion of a transition is not allowed when there is only two transition (to the second state and the last state). In the same way adding transition does not take place when the first state of the block has transition to all the other transition. In Fig. 7(c) and Fig. 7(d) a state is deleted from a block. The outcome depends on which block is deleted. In Fig. 7(e) and Fig. 7(f) a state is added to the block. Again the outcome depends on which block is added. In the cases of linear and self loop blocks, there is only one way to add and delete a state. These six different types of mutation supply the block-HMM with sufficient variation without changing the properties of the block.

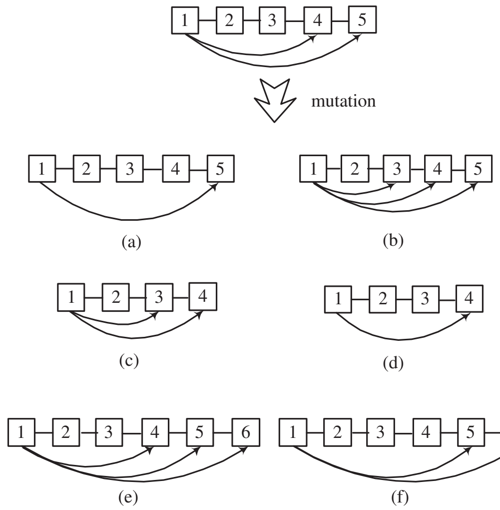

Fig. 7. Six possible types of mutations from a 4-state jump forward block.

In addition to changing the length of the block and the transitions, we also allow mutations that change the type of the block. For self-loop and forward jump blocks, we can mutate between tied and untied versions. We can also mutate the type altogether. We also allow type mutation to a block of zero length. Zero blocks allow us to explore structures with few number of blocks.

C. Fitness Evaluation

Over complex models are likely to fit the data well, but to generalise badly. We therefore need a way to gauge the generalisation ability. To achieve this we split the training data into two; one half is used as a Baum-Welch training set and the other half as a fitness evaluation set. We use the training set to find the transition and emission probabilities using the Baum-Welch algorithm and the evaluation set to measure the fitnesses used in selecting members of the population. Although we could potentially over-fit on the evaluation set, this does not seem to be a problem in practice. One explanation for this is that overly complex models will over-fit the training data when we run the Baum-Welch algorithm. They then perform badly on the fitness evaluation set. Because we use Baum-Welch at every generation, overly complex models will always be disadvantaged.

We take as fitness values the reciprocal of the negative log-likelihood

$$E_\mu = \frac{1}{-\log\left(\mathbb{P}\left(x|\Theta_\mu\right)\right)} \tag{18}$$

where μ labels the different members of the population. We select member of the population with a Boltzmann probability

$$F_\mu = \frac{w_\mu}{\sum_{\nu=1}^{N} w_\nu}, \qquad w_\mu = \mathrm{e}^{s\, E_\mu/\sigma} \tag{19}$$

where σ is the standard deviation in the distribution of fitnesses in the population. The parameter s controls the selection strength. We used stochastic universal sampling method to reduce the genetic drift when we perform selection [16].

D. Training Procedure

To test our HMM we split our data into a training and test set. The training data was further split into a Baum-Welch training set and a evaluation set, as describe above. The number of blocks is chosen at the beginning of the training and kept fixed. The length and type of blocks was randomly chosen so that there were on average the same number of linear, self-loop, forward-jump and zero blocks.

At each generation we applied one iteration of the Baum-Welch training algorithm. We then evaluated the fitness of the population and performed selection (with selection strength $s = 1$), mutation and crossover. Mutation and crossover are applied with a small probability. We continued until there was no significant change in the structure of the HMM.

IV. Experiment on Block-HMM

A. Experiment with Artificial Data

To test the performance of the block-HMM, we conducted three experiments with artificial data. The first two experiments were to find an HMM to represent data generated from the languages (ATG)$^+$ and (AAGATGAGGACG)$^+$ where '+' means any number of repetitions. We used a population composed of HMMs with 2 blocks. Table I shows parameters used in the toy experiment. That is, at each generation we choose two individuals and perform crossover to create two children which replace the two parent strings, one individual where we mutate either the length or the number of transitions in a randomly chosen block and one individual where we mutate the type of a randomly chosen block. For this simple problem we did not allow tying. The best solutions to these two problems found by the GA are illustrated illustrated in Fig. 8. The resulting HMMs are reasonable solutions to the problem although probably not those that a human would have come up with.

TABLE I

Block-HMM parameters used in the experiment.

Parameter	value
Population size	20
Iteration	300
Number of crossover per iteration	1
Number of mutation per iteration	1
Number of type mutation per iteration	1

The third test we carried out was to find an HMM to recognise the language (AAGATGAGGACG)$^+(ATGC)^+$. The first half of the sequence is the same as that used in the previous experiment, while the second half is a repetition of four symbols. The number of iteration used in this experiment is 600. Fig. 9 shows a graph of the maximum log-likelihood value versus the generation for four different runs of the GA. In case 3 of the three block model, the GA has still not found a particularly good HMM after 600 iterations.

The best solutions found by the GA in the four runs are shown in Fig. 10. Note that we have not shown transitions

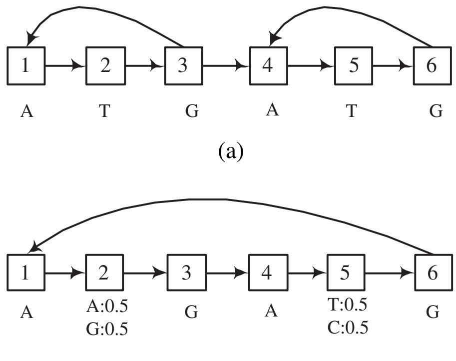

Fig. 8. The result of block-HMM with 2 blocks. (a) (ATG)$^+$ (b) (AAGATGAGGACG)$^+$

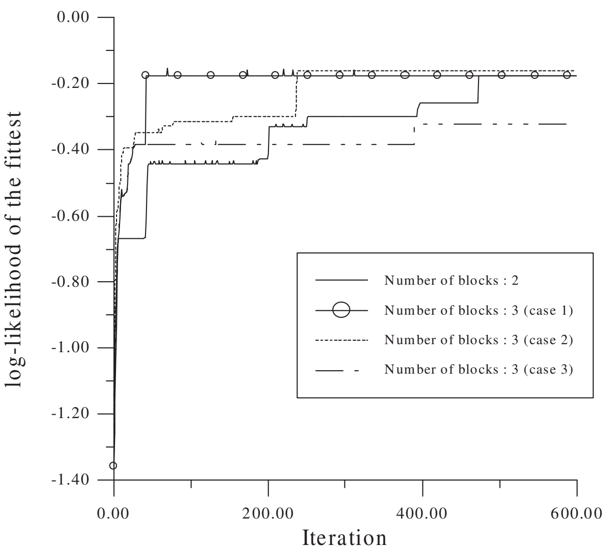

Fig. 9. The behaviour of the block-HMM GA shown as a function of the number of generations for 4 runs.

which the Baum-Welch algorithm has driven to zero. In the first run (Fig. 10(a)) we initiated the GA with two blocks, while in the next three runs (Fig. 10 (b),(c) and (d)) we used a three block model.

These toy examples illustrates that a GA is capable of finding reasonable structures that can be readily interpreted by humans. However, the models are not the simplest that could solve the problem nor do they always have an optimal structure. Nevertheless, they demonstrate that the GA can find reasonably parsimonious solutions which give a good approximation of the true likelihood. To test the performance of the block-HMM on more complex sequences, we used biological sequences in the following experiment.

B. Coding region model of *C. jejuni*

To investigate the block-HMM's ability to find an HMM structure for biological sequences, we performed an experiment with 200 sequences from the coding regions of *C. jejuni*.

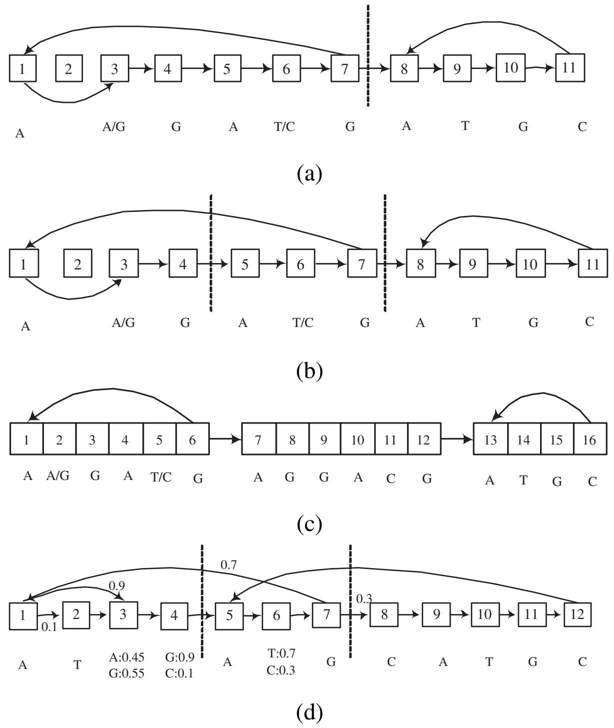

Fig. 10. The result of block-HMM for the (AAGATGAGGACG)$^+$$(ATGC)^+$ (a) with 2 blocks (b) with 3 blocks (case 1) (c) with 3 blocks (case 2) (d) with 3 blocks (case 3).

The sequence data comprises a start codon[1] (ATG) at the beginning of the sequence some number of codons and a stop codon (TAA, TAG or TGA) at the end. The length of these sequences range from around 200 to 2000. Figure 11 shows a typical HMM that could be used to representing the coding region [17]. Of the 200 sequences available, 150 sequences are used for training and 50 sequences for evaluation. From looking at the data alone it is almost impossible for non-specialists to see that the data consists of codons.

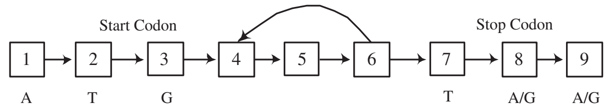

Fig. 11. A possible HMM candidate for the coding region of *C. jejuni*.

We conducted the experiment twice, once using four blocks and once with three blocks. The initial lengths of the blocks range between 3 to 7. The GA parameters used in the simulations are shown in table II.

Fig. 12 shows the resulting structures of the block-HMM found using GAs. In Fig. 12(a) the second state of the first block is not used. In Fig. 12(b) the emission probabilities of the state between 7 and 15 are tied. Detailed transition and emission probabilities are available from the web page mentioned at the end of this paper. Although these are not the most parsimonious solutions they do seem to have identified

TABLE II

BLOCK-HMM PARAMETERS USED IN THE EXPERIMENT WITH BIOLOGICAL SEQUENCES.

Parameter	value
Population size	30
Iteration	600
Number of crossovers per iteration	4
Number of mutations per iteration	4
Number of type mutations per iteration	4

the triplet nature of the sequences.

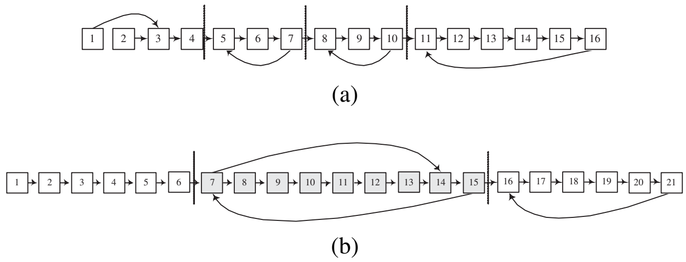

Fig. 12. The result of block-HMM. It searched the 3 state loops with GAs.

C. Promoter model of C. jejuni

A promoter is a region of DNA located upstream from the transcription start site. It contains binding sites for RNA polymerase and proteins that regulate the rate of transcription. There are conserved sequence of TTGACA at around the -35 region[2] and TAtAAT (where 't' indicates either T or A) at around the -10 region, called TATA box. In the promoter region of *C. jejuni*, the TTGACA region is weakly conserved, but a T-rich domain is common upstream of the TATA box [18]. Petersen *et al.* suggested an HMM architecture for this promoter region [19]. Their model includes the TATA box (TATAAT) and a Ribosomal binding site (AAGGA) with other parts of the structure constructed from knowledge of the promoter sequence. This model is shown in Fig. 13.

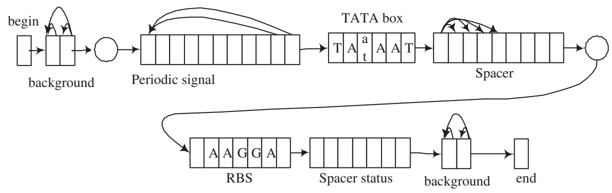

Fig. 13. Model for predicting the promoter region of *C. jejuni*, adopted from [19] and modified. They modelled this structure with biological knowledge.

The sequence contains more Ts and As then Gs and Cs. Although the Ribosomal binding sites can be easily checked, the sequence is not always in the same position. In some cases, the sequence AAGGA doesn't appear or is highly mutated. It is difficult for non-experts to figure out which part of the sequence is the TATA box because most part of the promoter region is composed of Ts and As.

[1]Codon is a group of three nucleotides that specifies one of 20 amino acids during translation of an mRNA into a polypeptide.

[2]'-35' means it is located 35bp before the transcription start site.

We tested the performance of the block-HMM to investigate if it can find the biologically meaningful region without any prior knowledge. The GA parameter are the same as those used in the previous experiment. Of 175 sequences available, 132 sequences are used for Baum-Welch training and 43 sequences are used for the fitness evaluation. We conducted simulations using 9, 8 and 7 blocks.

The best structures found by the GA are shown in Fig. 14. The block-HMM could find the 'AAGGA' and 'TAtAAT' regions with 9 blocks (Fig. 14 (a)) and 8 blocks (Fig. 14 (b)). In addition, it found the presence of semi-conserved TGx upstream of TATA box which is characteristic of the *C. jejuni* promoter region [19]. However, when the number of blocks is 7 (Fig. 14 (c)), the block-HMM could find only AAGGA sequence. The TATA box was buried inside other states. In Fig. 14, the emission probabilities of the states in the shaded cells are tied.

To perform a quantitative test of the generalisation performance of structures found in the previous experiment we conducted a discrimination test. In order to collect a sufficient amount of data we did a five-fold cross-validation experiment. This replicates a test performed by Petersen *et al.* [19]. As the structures generated in previous test and shown in Fig. 14 were found using all the training data, we retrained all the emission and transition probabilities starting from random values. However, to retain the main structure we removed all transition probabilities less than 0.002 and to maintain the positions of TATA box and Ribosomal binding site we introduced pseudo-count in these conserved regions. For example, if the emission probabilities of one state is A:0.9 T:0.1 G:0.05 C:0.05 than the pseudo-count becomes 90, 10, 5, 5 for the A,T,G,C respectively. This guarantees that these conserved regions occurs in the same place when we retrain the model in the cross-validation procedure. We used Baum-Welch to retrain the weights using 140 out of the 175 sequences as training data. The remaining 35 sequences were used as test data.

To perform a discrimination test we require a background sequence. To obtain this, we use a 500 000bp sequence generated by a third-order Markov chain that had been trained on the *C. jejuni* genome. To test the discrimination we set a log-odds threshold so that there were 10 or less false positives and then measured the number of true positives. Table III shows the total number of false positives (FP) and true positives (TP) summed over all of the five-fold cross-validation tests.

TABLE III
THE RESULT OF THE BLOCK-HMM

Simulation	Number of FP	Number of TP	TP rate
block-HMM (9 blocks)	7	132	75%
block-HMM (8 blocks)	9	126	72%
block-HMM (7 blocks)	10	120	69%
Petersen's HMM	10 (best case)	119	68%

We repeated this test on ten different other HMM structures found by using a GA. In all but one case, we could get 7∼9 false positives and a true positives rate ranging from 70% to 76%. Only on one simulation out of the ten, did the GA fail to find a TATA box. These results are superior to those published for a hand-crafted HMM [19], although we must be slightly cautious in interpreting our results as the structures were trained using the same data that it was tested on. We were forced to do this because the dataset was small. We did take the precaution of retraining all the weights to reduce the risk of biasing our results. Although, the results are not conclusive, they are very promising.

V. CONCLUSION

In this paper, we have described a Genetic Algorithm which evolves the structure of HMMs in a biologically constrained way. We have performed a number of tests to illustrate that the resulting HMMs are relatively easy to interpret. Furthermore on the problem of the promoter model of *C. jejuni* the results are competitive with an expert designed HMM. This is quite remarkable given that our GA had no prior knowledge of conserved regions such as the TATA box and Ribosome binding site as well as other structures which have been acquired by experts over many years. These are promising results from early work. There is a lot more that can be investigated. For example, in order to reduce over-fitting we have split our training data into a Baum-Welch training set and a evaluation set used in selecting members of the population. However, we could still over-fit the evaluation set. One method to control this is to include a regularisation term in that fitness term that punishes more complex models. Such a scheme has been used by Yada *et al.* [3]. However, in early experiments we found that this involved setting a parameter which was difficult to do with the limited amount of data. Nevertheless, this may be worth revisiting, particularly as we try to evolve more complex models.

Evolving HMM structures is a time consuming process since training and evaluating the likelihoods for long sequences takes many operations. At present, this is a restriction on automatic structure finding. However, as computer get faster and the number of applications grow with the exponential rise in the amount of sequencing data we expect that automatic HMM structure search will become increasingly important. Genetic Algorithms seem ideally placed to play an important part in this development.

Data Availability

The biological data used in the tests discussed in this paper can be obtained from http://www.ecs.soton.ac.uk/∼kjw02r/data.htm. In addition, this site contains all the details of the HMM structures found by the Genetic Algorithm.

June 22, 2004

REFERENCES

[1] L. R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[2] R. M. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological sequence analysis*. Cambridge: Cambridge University Press, 1998.

[3] T. Yada, M. Ishikawa, H. Tanaka, and K. Asai, "DNA sequence analysis using hidden markov model and genetic algorithm," in *Genome Informatics Vol.5, pp.178-179*, 1994.

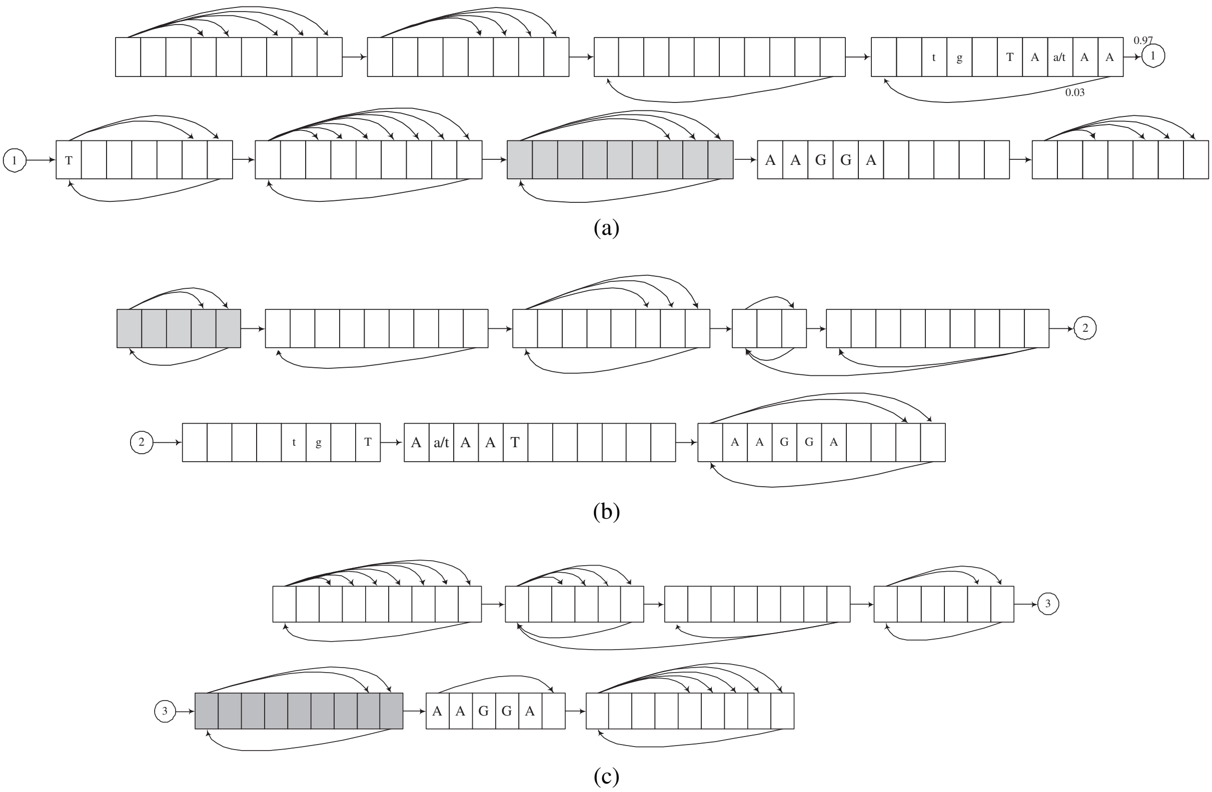

Fig. 14. The best structures found by the GA for (a) 9, (b) 8 and (c) 7 blocks. The 'AAGGA' sequence is found on every simulation and 'TAtAAT' sequence is found in (a) and (b). Each cell represents a state. Emissions of shaded cells are tied.

[4] C. W. S. Chau, C.K.Diu, and W.R.Fahrner, "Optimization of HMM by a Genetic Algorithm," in *International Conference on Aucoustics, Speech and Signal Processing*, 1997, pp. 1727–1730.

[5] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[6] I. Poli and A. Roverato, "A genetic algorithm for graphical model selection," *Journal of the Italian Statistical Society*, vol. 2, pp. 197–208, 1998.

[7] I. Inza, P. L. naga, and B. Sierra., "Feature subset selection by bayesian networks: a comparison with genetic and sequential algorithms," *International Journal of Approximate Reasoning*, vol. 27, no. 2, pp. 143–164, 2001.

[8] A. Dempster, N. M. Laird, and D. B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*. Royal statistical Society B, 1977, vol. 39, pp. 1–38.

[9] K. Sjolander, K. Karplus, M. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler, "Dirichlet mixture: A method for improved detection of weak but significant proten sequence homology," in *CABIOS, vol.12, pp.327-345*, 1996.

[10] A. Krogh and S. K. Riis, "Hidden Neural Networks," vol. 11, 1999, pp. 541–563.

[11] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Transactions on Pattern Anal. Machine Intell.*, vol. 5, pp. 179–190, 1983.

[12] A. Stolcke, "Bayesian Learning of Probabilistic Language Models," Ph.D. dissertation, University of California at Berkeley, 1994.

[13] Y. Fujiwara, M. Asogawa, and A. Konagaya, "Motif Extraction using an Improved Iterative Duplication Method for HMM Topology Learning," in *In Pacific Symposium on Biocumputing '96*, 1995, pp. 713–714.

[14] K.-J. Won, A. Prügell-Bennett, and A. Krogh, "The Block Hidden Markov Model for Biological Sequence Ananlysis," in *KES2004, submitted*, 2004.

[15] M. I. Jordan, *Learning in Graphical Models*. MIT Press, 1999.

[16] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proceedings of the Second International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates (Hillsdale), 1987.

[17] A. Krogh, *An introduction to hidden Markov models for biological sequences*, ser. Computational Methods in Molecular Biology. Amsterdam: Elsevier, 1998, ch. 4, pp. 45–63.

[18] M. M. Wosten, M. Boeve, M. G. Koot, A. C. van Nuene, and B. A. van der Zeijst, "Identification of *Campylobacter jejuni* promoter sequence," *J. Bacteriaol.*, vol. 180, pp. 594–599, 1998.

[19] P. L., T. S. Larsen, D. W. Ussery, S. L. W. On, and A. Krogh, "RpoD promoters in Campylobacter jejuni exhibit a strong periodic signal instead of a -35 box," *Journal of Molecular Biology*, vol. 326, no. 5, pp. 1361–1372, 2003.

Kyoung-Jae Won received the BS and MS degree in electronics from Chung-Ang University, Korea, in 1996, 1998, respectively. Under the volunteer program of Korean government, he worked at the College of Science, National University of Hanoi, Vietnam, 1999-2001. Currently, he is a PhD candidate at School of Electronics and Computer Science, Southampton University, U.K. His interest are in the areas of bioinformatics, Hidden Markov Models and evolutionary algorithms.

Adam Prügel-Bennett received B.Sc. degree in physics from Southampton University, Southampton, U.K., and the Ph.D. degree in theoretical physics from Edinburgh University, Edinburgh, U.K. He worked in research jobs in Oxford, Paris, Manchester, Copenhagen, and Dresden before finally returning to Southampton University as a Senior Lecturer in the School of Electronics and Computer Science. His main areas of research are in evolutionary algorithms, machine learning, and computer vision.

Anders Krogh is a Professor of Bioinformatics at University of Copenhagen.