Vol. 20 no. 18 2004, pages 3613–3619
doi:10.1093/bioinformatics/bth454

Training HMM structure with genetic algorithm for biological sequence analysis

Kyoung-Jae Won[1,*], Adam Prügel-Bennett[1] and Anders Krogh[2]

[1]ISIS Group, ECS, University of Southampton, SO17 1BJ, UK and [2]Bioinformatics Centre, University of Copenhagen, DK-2100 Copenhagen, Denmark

Received on October 23, 2004; revised on July 11, 2004; accepted on July 23, 2004
Advance Access publication August 5, 2004

ABSTRACT

Summary: Hidden Markov models (HMMs) are widely used for biological sequence analysis because of their ability to incorporate biological information in their structure. An automatic means of optimizing the structure of HMMs would be highly desirable. However, this raises two important issues; first, the new HMMs should be biologically interpretable, and second, we need to control the complexity of the HMM so that it has good generalization performance on unseen sequences. In this paper, we explore the possibility of using a genetic algorithm (GA) for optimizing the HMM structure. GAs are sufficiently flexible to allow incorporation of other techniques such as Baum–Welch training within their evolutionary cycle. Furthermore, operators that alter the structure of HMMs can be designed to favour interpretable and simple structures.

In this paper, a training strategy using GAs is proposed, and it is tested on finding HMM structures for the promoter and coding region of the bacterium *Campylobacter jejuni*. The proposed GA for hidden Markov models (GA-HMM) allows, HMMs with different numbers of states to evolve. To prevent over-fitting, a separate dataset is used for comparing the performance of the HMMs to that used for the Baum–Welch training. The GA-HMM was capable of finding an HMM comparable to a hand-coded HMM designed for the same task, which has been published previously.

Contact: j.won@ecs.soton.ac.uk

INTRODUCTION

The vast amount of biological sequence data being produced has been accompanied by the development of machine learning techniques to extract as much information from the data as possible. One of the most successful class of techniques for analysing biological sequences has been hidden Markov models (HMMs). Although these models were originally applied to speech recognition (Rabiner, 1989), they have been proved highly successful for modelling biological sequences (Durbin *et al.*, 1998). Also, they have shown good performance in predicting the structure of biological sequences (Asai *et al.*, 1993). In this paper, the secondary structure of proteins was predicted by the HMMs. The success of HMMs owes much to their ability to encode biological information in their structure while allowing many unknown quantities to be learnt through the optimization of their transition and emission probabilities. The constraints imposed by their structure will often limit excessive overfitting of the training data, although some overfitting is still observed when using Baum–Welch training.

Automatic optimization of the structure of HMMs would potentially be highly beneficial. In many applications the biological mechanism is not well understood. Given sufficient data, it would be possible to learn an HMM architecture that encodes some biological information that we are unaware of. However, in learning the structure of an HMM we do not wish to lose the advantages they currently offer. In particular, we wish to control the complexity of the HMM models and if possible retain their biological interpretability.

In this paper, we investigate the effectiveness of using Genetic Algorithms (GAs) for optimizing the HMM structure. A GA is a robust general purpose optimization technique that evolves a population of solutions (Goldberg, 1989). It is easy to hybridize other algorithms such as Baum–Welch training within a GA. Furthermore, it is possible to design operators that favour biologically plausible changes to the structure of an HMM. That is, to ensure that modules of the states are kept intact. GAs have been widely used to optimize architecture for Neural Networks (Yao, 1999). However, to the best of our knowledge, GAs have only been used once before to optimize the structure of an HMM. Yada *et al.* (1994) used a GA to find a TATA box model. They included a term in their fitness function to penalize overcomplex models, however, their results depended critically on this parameter. In this paper, we explore the use of GAs for evolving HMMs. Our GA differs from Yada *et al.* in that we split the training examples into two. One half is used for training the HMMs using Baum–Welch, the other half is used to evaluate the HMM's fitness. This reduces overfitting of the training data. In addition, to prevent overfitting we performed Baum–Welch only on a proportion of randomly selected individuals from the population of each generation.

*To whom correspondence should be addressed.

There have been a number of earlier attempts to learn the structure of an HMM. Stolcke (1994) started from an HMM with a large number of states and merged the states using the log-likelihood and posterior probability as criteria for choosing which states to merge. Stolcke did not allow states that can be useful for dealing with new data to be split. A method involving both state splitting as well as deleting negligible states and transitions has also been used to find a good HMM topologies (Fujiwara *et al*., 1995). They used the transition ambiguity and the expected observation differences as criteria for splitting a state and applied this to find an HMM structure of a leucine zipper motif. These statistical approaches can be used to find a particular pattern-like motif; however, they have not been successfully used to find HMMs that model sequences with long-range structure.

SYSTEMS AND METHODS

GAs for HMM (GA-HMM)

To discover if GAs are potentially useful for evolving HMMs, we implemented a standard GA where a population of HMMs are evolved from one generation to the next. At each generation some proportion of the HMMs are trained with Baum–Welch on a training set. The fitness of the HMMs are measured on a evaluation set and the fitter members are selected. Finally, the members are mutated and crossed-over to form the next generation. This procedure is shown in Figure 1. The state labelled genetic operations includes selection, mutation and crossover.

In the experiments described below, the initial population consists of HMMs with just two states. The number of states will change due to state insertion and deletion mutations and through crossover. Also, as part of the initialization stage the sequence data (O) is divided into a set used for training with Baum–Welch (O_{train}) and a set used for evaluating the fitness (O_{eval}). The algorithm terminates when there is no significant change in the structural model.

Genetic operations for GA-HMM

The genetic operations consist of selection, mutation and crossover in that order. Selection uses proportional selection with stochastic universal sampling (Baker, 1987) to reduce genetic drift. In both stochastic universal sampling and the more traditional roulette wheel selection, the sampling process can be visualized as assigning the pocket sizes of a roulette wheel to be proportional to the probability of selecting an individual. In roulette wheel selection P games are played independently to select P individuals. In stochastic universal sampling, the roulette wheel is spun once and P individuals are selected at equally spaced intervals around the wheel.

For a mutation to be useful it should make changes that cause minimal disruption so that the new HMM has a high probability of having a fitness close to that of the unmutated HMM. We considered mutations that only change either the number of states or the number of transitions by one. This gave us four mutation operators; insert state, delete state, insert transition and delete transition, which are shown in Figure 2. Insertion of a state can happen between any two states or at either end of the chain. When a state is inserted, the states on its right-hand side shift by one as shown in Figure 2a. The emission probabilities of the new state are set to randomly selected values. If a state is deleted, all its transitions are removed. Insertion of a transition can happen between any two states and deletion of a transition happens at any state if the state has more than one outgoing transition. Although these mutations allow highly interconnected HMMs they provide a certain bias towards chain structures because of the state insertion operator.

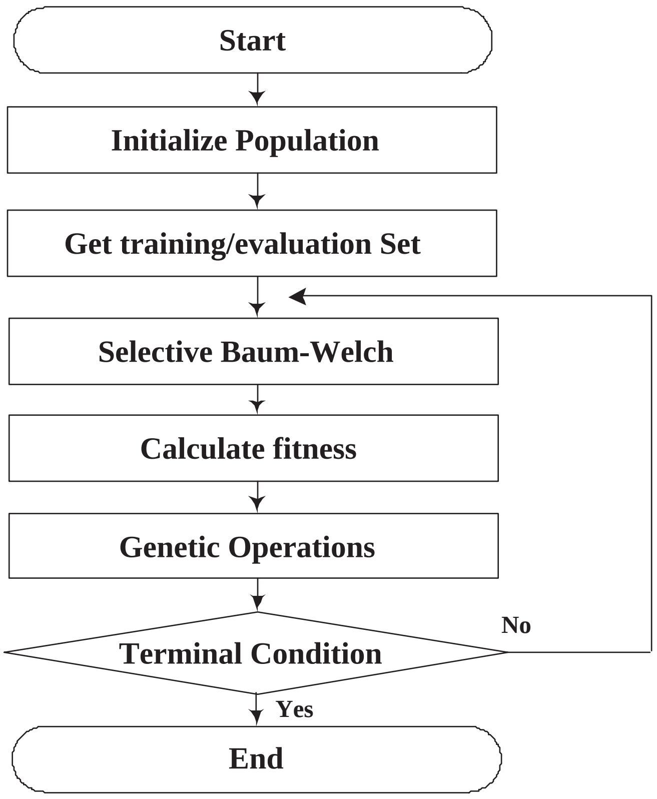


Fig. 1. The GA-HMM algorithm. Baum–Welch training is combined with selection, mutation and crossover to evolve HMMs.

Crossover takes place between two HMMs and exchanges states. A number of successive states can be crossed over in one operation. Only outgoing transitions from a state are exchanged during crossover. An example of crossover is shown in Figure 3.

Selective Baum–Welch

The Baum–Welch algorithm is commonly used to train HMMs. It estimates model parameters from training sequences while maximizing the log-likelihood of the model.

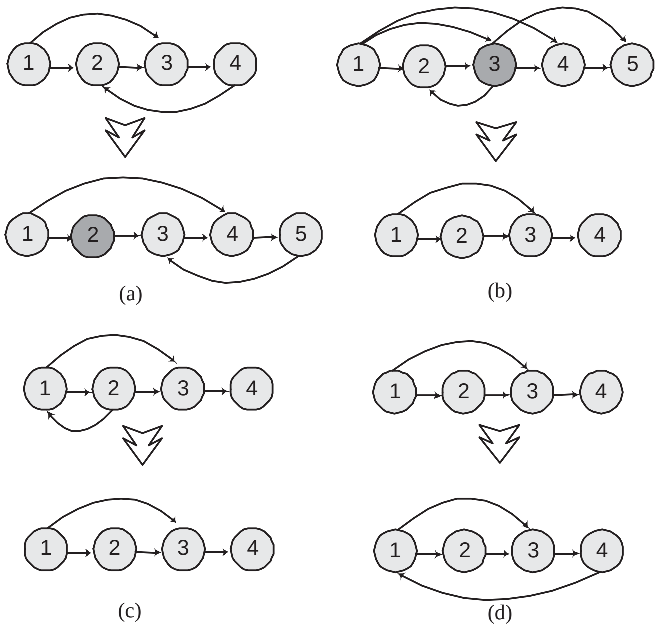


Fig. 2. Four types of mutations. (**a**) insert state (inserting a state in the second position), (**b**) delete state (delete the third state), (**c**) delete transition and (**d**) insert transition.

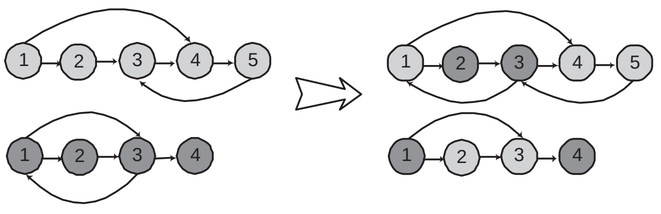


Fig. 3. Crossover. During crossover outgoing transitions move with transition.

The log-likelihood of model k is

$$L_k = \log[P(O|\theta_k)], \qquad (1)$$

where θ_k denotes the parameters of the k-th HMM and O is the given sequences.

Although Baum–Welch maximizes the log-likelihood for the sequences, it can overfit the given sequences and produce inferior results compared with HMMs that have experienced fewer training cycles. We tested this by training sequences with Baum–Welch without applying the GA algorithm. Figure 4 shows the negative log-likelihood versus the number of Baum–Welch iterations in an experiment with one of the models described later. The negative log-likelihood measured on the sequence data decreases monotonically, whereas on unseen data it initially decreases, but then increases again due to overfitting. Initially, we tested a GA where Baum–Welch was performed at each generation. However, this caused overfitting as the HMMs were trained too frequently. A selective

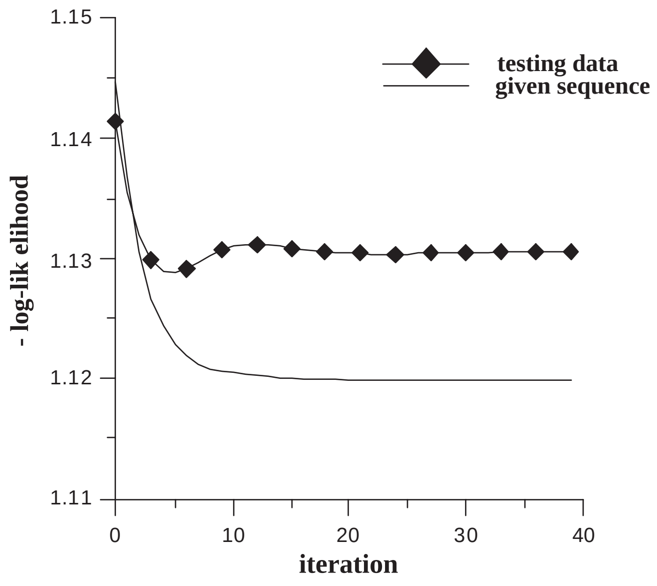


Fig. 4. The graph shows the negative log-likelihood for an HMM plotted against the number of Baum–Welch iterations. The negative log-likelihood for the given data is monotonically decreased by the Baum–Welch algorithm. However, the negative log-likelihood measured on independent testing data will typically decrease, reach a minimum and then increase again. In this instance, the best generalization performance was found after five iterations. The training data used were from *Campylobacter jejuni* and is described in experiment I below.

Baum–Welch scheme was adopted to reduce overfitting. The generalization performance was found to improve by changing the algorithm so that Baum–Welch training only occurred with a fixed probability. In this scheme, 20% of the population are randomly selected in each generation for the Baum–Welch training. In addition, we made sure that the fittest member of the population was never subjected to Baum–Welch training as this could cause it a loss of fitness. The overfitted HMM structures receive lower fitness value when the HMM is applied to the test set.

Fitness value

The performance of the GA was found to depend strongly on the fitness value used. The fitness value used by Yada *et al.* (1994) incorporated a balance factor between complexity and likelihood

$$w_k = \frac{1}{-2\log[L(O;\theta_k)] + 2\lambda p_k}. \qquad (2)$$

Here, $L(O;\theta_k)$ is the maximum-logarithmic-likelihood estimate of the k-th HMM, p_k is the number of free parameters in the k-th HMM (its complexity) and λ is the balance factor. By adjusting the balance factor, the complexity can be controlled. However, the complexity was found to be very

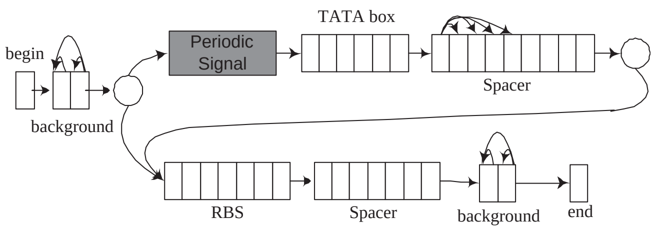


Fig. 5. Model for predicting the promoter region of *C.jejuni* from Petersen *et al.* (2003). In Experiment I we try to learn the periodic region, starting from HMMs with two states.

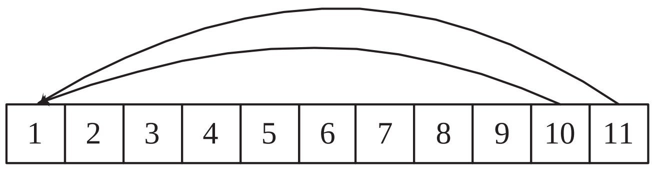


Fig. 6. The hand constructed HMM model for the periodic promoter region of *C.jejuni* used by Petersen *et al.* (2003).

sensitive to the value of λ and it was difficult to choose a value for λ a priori.

In this paper, we avoided the problem of setting the balance factor by separating the given data (O) into a set used for Baum–Welch training (O_{train}) and a set used for evaluation of the generalization ability (O_{eval}). The fitness function used in our GA-HMM algorithm is

$$w_j = \frac{1}{-\log[L(O_{\text{eval}};\theta_j)]}. \tag{3}$$

This method reduces overfitting. As the training and evaluation sets are separated, overfitted models get lower fitness values in the fitness evaluation stage and will be discarded through the selection operation.

The probability of choosing member k during the selection is then given by

$$F_k = \frac{w_k}{\sum_{j=1}^{P} w_k}. \tag{4}$$

RESULTS AND DISCUSSION

Experiment I: promoter model of *C.jejuni*

Our first experiment was to find an HMM for the RpoD promoter region upstream of the TATA-box (Petersen *et al.*, 2003). The full promoter model developed by Petersen *et al.* is shown in Figure 5. The RpoD promoter consists of approximately repeated blocks with an observed period of 10.6 nucleotides. A hand-crafted model for this region used in Petersen *et al.* (2003) is shown in Figure 6.

In our experiment, we evolved a population of HMMs for this region using a GA starting from two states models. The

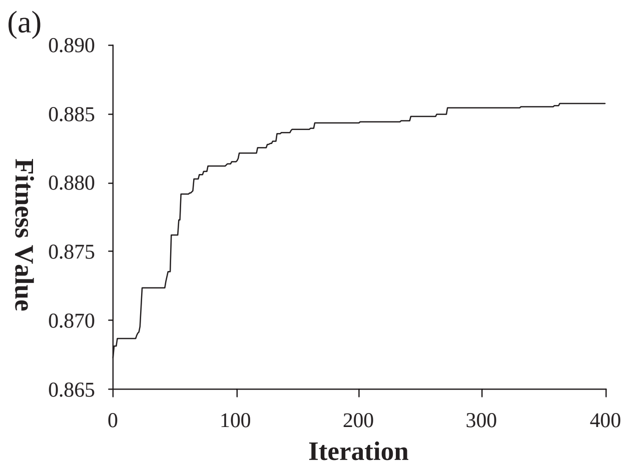
Table 1. GA-HMM parameters used in Experiment I

Parameter	Value
Population size	30
Offspring size	4
Iteration	300
Crossover rate	0.06
Insert/delete transition	0.06
Insert/delete state	0.12

parameters controlling the GA are shown in Table 1. The dataset consisting of 175 sequences and 135 sequences was used for cross-validation (O) and 40 for testing (S_{test}). Of the 135 sequences, 95 were used for Baum–Welch training (O_{train})and 40 used for measuring the fitness (O_{eval}) . The experiment was repeated five times (5-fold cross-validation) to obtain better statistics. The data used in this experiment can be obtained from http://www.ecs.soton.ac.uk/~ kjw02r/data.html.

Figure 7a shows the fitness of the best member of the population versus the generation for the first data partition. Only evaluation sequences that are not involved in the training are used in calculating the fitness value [Equation (3)]. Because the GA-HMM reduces overfitting, the fitness value of the fittest member of the population always increases. The graphs for the other cross-validation sets look similar. In Figure 7b, the average number of states in the HMM is shown. We observe that the number of states grows until it reaches around 10, which appears to support the observed periodicity found in this region of the promoter. At this stage the number of states remains roughly fixed while the fitness value continues to rise, indicating that the fine structure of the model is still evolving. Occasionally, the number of states found by the GA-HMM was less than 10 states. We believe this is because the GA-HMM found a local optimum of around 5–6 states.

An example of one of the HMM structures found by the GA is shown in Figure 8. The model is considerably more complex than that proposed by Petersen *et al.* shown in Figure 6. Table 2 shows a comparison of the fitness values of the test set $[-1/L(S_{\text{test}};\theta)]$ and their variances obtained from the five cross-validation experiments for our GA-HMM. In the same table, we show the fitness values of the test set for the model proposed by Petersen *et al.* (2003) and trained using Baum–Welch. The hand-crafted model is trained only with the Baum–Welch algorithm. Of 175 sequences in their model, for each test 140 sequences are used for the Baum–Welch training and 35 sequences are used to calculate the fitness. Because the Baum–Welch algorithm finds a locally optimal solution, the final results depend on the initial values of the transition and emission probabilities. For the hand-crafted model and the GA model, we conducted 10 experiments with different initial values.



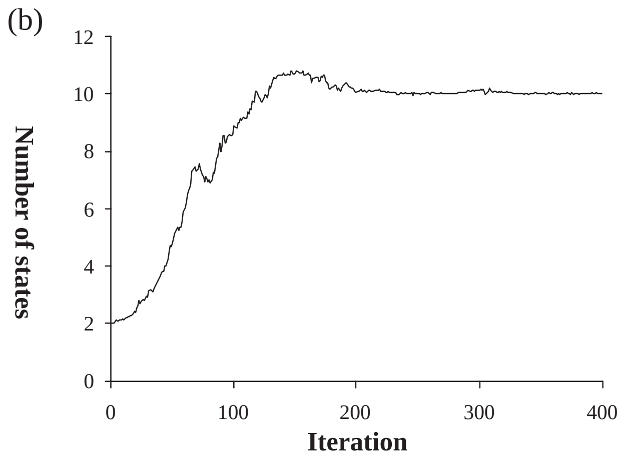


Fig. 7. The results during GA-HMM training: (**a**) shows the fitness value of fittest individual on each iteration and (**b**) shows the average number of states for periodic signals. The GA started with a population consisting of two states. After 150 generations the HMM have a length of 10 states. Although the length does not significantly change thereafter, the fitness continues to improve indicating that the finer structure is being fine tuned.

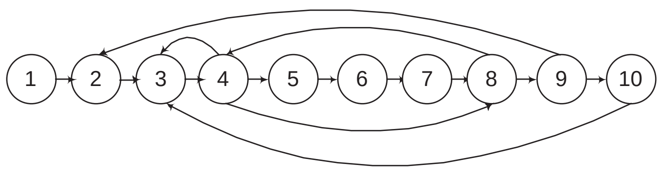


Fig. 8. After training the *C.jejuni* sequences, GA-HMM found one structure model for the periodic signal.

The performance of the HMM found using the GA and the hand-designed HMM are roughly similar. Even though it does not show better performance in Test 3, it produces slightly better result in the other tests. The paired t-test was conducted to assess the statistical difference (Mitchell, 1997). This analysis compares the means of two groups to see if those two groups are statistically different. Let $X_{GA}(i)$ and $X_P(i)$ be the results for the GA-HMM and Petersen *et al*. (2003). Then, the t-value for the k samples can be defined by

Table 2. Comparison between Baum–Welch and GA-HMM.

Test	Petersen's model	GA-HMM
1	0.8855(3.0e−6)	0.8863(1.3e−6)
2	0.8854(1.1e−6)	0.8861(1.9e−6)
3	0.8887(3.9e−6)	0.8877(2.4e−6)
4	0.8796(5.6e−7)	0.8797(4.5e−7)
5	0.8713(2.0e−6)	0.8767(4.3e−6)

The results show the fitness value for the test sequences ($-1/L(S_{test};\theta)$) and variance for five different partitionings of the data.

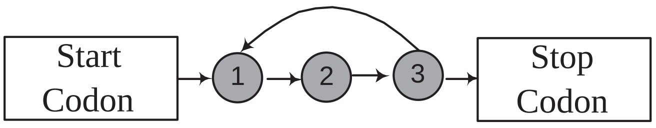


Fig. 9. HMM architecture for *C.jejuni* coding region.

$$t\text{-value} = \frac{\bar{d}}{\sqrt{\sigma_d^2/k}}, \tag{5}$$

where $\bar{d}$ is the mean of the differences $X_{GA}(i) - X_P(i)$, and σ_d is the SD of this mean.

This statistic has $n-1$ degrees of freedom. The t-values for the each test set are given in Table 2. In this experiment, the degrees of freedom is 4. With this result, the significance level at which two distributions differ can be determined. The t-value obtained is 1.09. By using the t-value and the degrees of freedom, the probability of null hypothesis can be obtained from the table for t-distribution. To get more precise probability we interpolate the values in the table of t-distribution. The probability of this result, assuming the null hypothesis, is 0.336. This result says that about 66 times out of 100 cases a statistical difference between two algorithms can be found. From this test, we can conclude that the GA-HMM has slightly better performance than the hand-crafted model overall.

Experiment II: coding region model of *C.jejuni*

To see if a GA is capable of finding biologically interpretable solutions, we performed a second experiment using 556 sequences from the coding regions from *C.jejuni*. The sequence data comprises a start codon (ATG), some number of codons and a stop codon (TAA, TAG or TGA). A simple HMM architecture for detecting this region would consist of a three-state loop (Fig. 9). A third-order state model was used to model the codon region. That is, instead of a state emitting

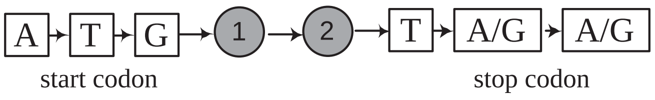


Fig. 10. Initial HMM architecture.

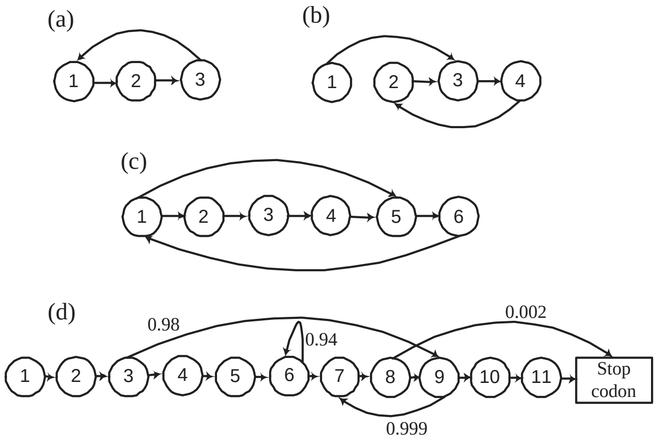


Fig. 11. The result of simulation for the *C.jejuni* coding region: (**a**) has loop (1-2-3); (**b**) has a path 1-3-4, (2-3-4); (**c**) has (1-5-6) and (1-2-3, 4-5-6) loops; and (**d**) has 1-2-3, (9-7-8), 9-10-11 or 1-2-3, 4-5-6, (6-6-6), 6-7-8, (9-7-8), 9-10-11.

a symbol from a single base unconditional distribution, the state emits a symbol, which is dependent on the three previous bases in the sequence, through a conditional probability distribution. We again used a two-state HMM as an initial model for the coding region as shown in Figure 10. The transition and emission probability for the start codon and stop codon was not evolved. Each state is in third order. The GA was run for 700 generations. Of the 566 sequences, 160 sequences were used for evaluation. The other parameters are set as in Experiment I.

Figure 11 shows some of the HMM architectures produced by the GA in this experiment. Even though some results were hard to interpret, almost every simulation produced an interpretable model. Figure 11a shows the same result as the common model. Figure 11b has a path 1-3-4,(2-3-4)—here, bracket indicate the a loop. Figure 11c has (1-5-6) and (1-2-3, 4-5-6) loops. Figure 11d has 1-2-3, (9-7-8), 9-10-11 or 1-2-3, 4-5-6, (6-6-6), 6-7-8, (9-7-8), 9-10-11 loops. These HMM solutions, although complicated, predominantly generate triplets. We performed 10 experiments and on every experiment with coding regions we could find the clean three-state loops as shown in Figure 11 a–c. Figure 11d shows the worst result we have obtained. The numbers shown are transition probabilities. Other transition probabilities are easily calculated, because the sum of outgoing transitions from a state is always 1. Because of the transition probabilities, the most probable loops are 1-2-3, (9-7-8) or 1-2-3, (9-7-8), 9-10-11.

DISCUSSION

The experiments described here suggest that GAs are quite capable of finding reasonable HMM architectures for biological sequence analysis. Even with a rather naive implementation, the GA was able to achieve comparable or slightly superior generalization performance to a hand-designed HMMs. Both experiments show that a major drawback of automating the design of HMM architectures is that the resulting model may be difficult to interpret biologically. Although we used genetic operators that favoured the building of chain structures they nevertheless allowed considerable cross-linking within the chain. The mutation operators could be modified to strongly bias biologically interpretable models. This is an area we are currently investigating. A drawback of constraining the search is that we may inhibit the GA from discovering completely novel types of architecture. One of the merits of GAs is the capability of dealing with substructures of the solution. In the GA-HMM, the crossover operators can swap a series of states in one operation. The emission probabilities are also crossed over with the states. This enables the GA-HMM to swap any meaningful part of the HMM structure. In this sense, the proposed crossover scheme can treat such modularity, even though the strategy does not completely implement the modular structure. The Genetic Programming methods that deal with modularity and capture the hierarchical structures were developed (Yada, 1998; Yada *et al*., 1995). They used S-expressions to encode the network structures and applied genetic programming to find the optimal structure.

We prevented the GA from producing overcomplex models by measuring the fitness on a different set of data from that used for training Baum–Welch. This ensures reasonable solutions but did not entirely exclude overcomplex models from appearing. Yada *et al*. (1994) incorporated a penalty term depending on the number of states in their fitness function. Although we found such penalty terms difficult to use because they required careful parameter selection to work, they may be beneficial if applied properly. This is another area that deserves further research.

Our experiments were carried out on short sections of an HMM. It seems very unlikely in the current state of development that a GA would be able to find large HMM structures *ab initio* that are competitive with hand-designed architectures. Nevertheless, even in the short-term GAs may be able to 'tune' a hand-designed HMM especially in areas where the biological significance of a region is poorly understood.

ACKNOWLEDGEMENTS

We would like to thank Lise Petersen for her help with the sequence data, the model and her valuable comments on this work.

REFERENCES

Asai,K., Hayamizu,S. and Handa,K. (1993) Prediction of protein secondary structure by the hidden Markov model. *Comput. Appl. Biosci.*, **9**, 141–146.

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

Durbin,R.M., Eddy,S.R., Krogh,A. and Mitchison,G. (1998) *Biological Sequence analysis*. Cambridge University Press, Cambridge.

Fujiwara,Y., Asogawa,M. and Konagaya,A. (1995) Motif extraction using an improved iterative duplication method for HMM topology learning. In *Pacific Symposium on Biocumputing'96*, pp. 713–714.

Goldberg,D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, Mass.

Mitchell,T.M. (1997) *Machine Learning*. McGraw-Hill Companies, Inc.

Petersen,L., Larsen,T.S., Ussery,D.W., On,S.L.W., and Krogh,A. (2003) RpoD promoters in *Campylobacter jejuni* exhibit a strong periodic signal instead of a −35 box. *J. Mol. Biol.*, **326**, 1361–1372.

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

Stolcke,A. (1994) *Bayesian Learning of Probabilistic Language Models*. PhD thesis, University of California at Berkeley, Berkeley, CA.

Yada,T., Ishikawa,M., Tanaka,H. and Asai,K. (1994) DNA sequence analysis using hidden markov model and genetic algorithm. In *Genome Informatics*, Vol. 5, pp. 178–179.

Yada,T., Totoki,Y., Asai,K. and Ishikawa,M. (1995) Generation of hidden markov model describing complex motif in DNA sequences. *IPSJ Trans.*, 750–767.

Yada,T. (1998) *Stochastic models representing DNA sequence data—construction algorithms and their applications to prediction of gene structure and function*. PhD thesis, University of Tokyo, Tokyo.

Yao,X. (1999) Evolving artificial neural networks. *Proc. IEEE*, **87**, 1423–1447.