Danmarks Tekniske Universitet

DTU

Implementation of Baum-Welch Algorithm

Authors

Carlota Carbajo Moral - s202424
Celia Burgos Sequeros - s202423
Christian Holm Johansen - s202770
Isabel Diaz Pines-Cort - s202406

June 22, 2021

Abstract

Hidden Markov Models (HMMs) have been widely used in the context of bioinformatics for various sequence analysis applications. In this project we illustrate the ability of the Baum-Welch algorithm to estimate the parameters of a simple HMM (the unfair casino problem) in an unsupervised manner. Furthermore, the predictive ability of the estimated model parameters is evaluated by means of the Viterbi and posterior decoding algorithms, and an explanation is given as to why the first can make prediction mistakes and how the latter might give more insight into the credibility of the predictions. Lastly, we argue for the extension of the Baum-Welch algorithm into biological sequence analysis and present the challenges that arise from transitioning to more complex models, as well as its advantages versus other methods commonly used for biological sequence analysis.

Contents

1 Introduction

1.1 Hidden Markov Models

In the context of bioinformatics, Hidden Markov models (HMMs) have a wide range of applications, such as sequence alignment, protein structure prediction and identification of functional motifs (domains). When a system is modelled using HMMs, it is assumed to be a Markov process with unknown parameters and the goal is to determine the value of the hidden parameters from the observable ones. Therefore, HMMs can be defined as models in which the distribution that generates an observation depends on the state of an underlying and unobserved Markov process [1] [2].

According to Ephraim and Merhav [3], the technical definition of a HMM is:

> "A hidden Markov process (HMP) is a discrete-time finite-state homogeneous Markov chain observed through a discrete-time memoryless invariant channel."

This definition can be broken down as follows: the number of possible states in the HMM is finite. The probabilities in the model are constant over time, which means that they are time homogeneous processes (they have stationary probabilities). HMM are stochastic memoryless processes, which means that the distribution of the next state in the models depends uniquely on the current state (they lack memory) [3] [1].

The HMMs consist of two processes:

- The process of moving between states, which is a finite-state Markov chain that generates the sequence of states of variables. It is characterized by the initial state probabilities and the state transition probabilities between variables. It is a hidden process (unobserved) because the variable states cannot be directly observed from the data [3] [1].
- The process of emitting an output sequence. It is characterized by the emission of one character of a given set of characters (or alphabet) from each state, with a probability distribution that only depends on the state. The unobserved process mention before is observed through the sequences of emitted symbols occurring in this process [3] [1].

Taking this definition into account a HMM will be specified by the following elements: a set of states, an alphabet with the symbols that can be emitted, the initial probability distributions associated to each of the states i.e. the probability of starting in each state, the transition probabilities of moving from one state to the next and the emission probability that a concrete symbol is observed in concrete state [1].

HMM were initially introduced in 1966 and were known as probabilistic functions of Markov chains. After some years of active research in this field of statistics, researchers developed the forward-backward recursions for calculating the conditional probability of a state given an observation sequence from a general HMM. They also presented a computationally efficient iterative procedure for maximum likelihood estimation of the parameters of a general

HMM using the forward-backward recursions. This procedure is known as the expectation-maximization (EM) algorithm, but it is also referred to as the Baum-Welch algorithm [3].

1.2 The Unfair Casino Problem

In this project we aim to implement the aforementioned Baum-Welch algorithm in order to estimate the model parameters of a model with two hidden states and six possible categorical observable outcomes seen in Figure 1. This model corresponds to the unfair casino problem. In this setting, we have two dice, a fair and a loaded dice. While the fair dice has an equal probability of generating each of the six possible outcomes (1, 2, 3, 4, 5, 6), the probability of rolling a 6 is higher than the probability of rolling any other number with the loaded dice. This rather simple case will serve as a proof of concept that training an HMM in a unsupervised manner is possible using the Baum-Welch algorithm, which can then be extended to also work on unlabelled sequence data in the field of bioinformatics.

HMM can be used for the characterization of certain parts of nucleic acids or proteins sequences. Each place in a sequence can be considered as a state space, which will have several possible emissions to choose from (4 in the case of nucleic acids and 20 in the case of proteins). Each state of the biological sequence is sequentially dependent on the adjacent element (nucleotide or amino acid). Therefore, each position in the sequence can be represented with a variable that takes a value of one of the states in the state space for that concrete place in the sequence [1].

2 Materials and Methods

2.1 Sequence generation

In order to obtain training data for the hidden Markov model some data was simulated. This was done by firstly creating a hidden Markov model as seen in Figure 1. Here we defined the probabilities, that we wanted to re-estimate using the Baum-Welch algorithm. To generate a sequence an initial state was chosen randomly with a probability of 1/2 for each state, and then the first number in the sequence randomly according to the emission probabilities for the selected state. Thereafter the model then extended this sequence randomly by first selecting the next state according to the transition probabilities and then choose a number according to the emission probabilities. This was then repeated until the sequence had grown sufficiently large, which was defined as a length of 40 in our case.

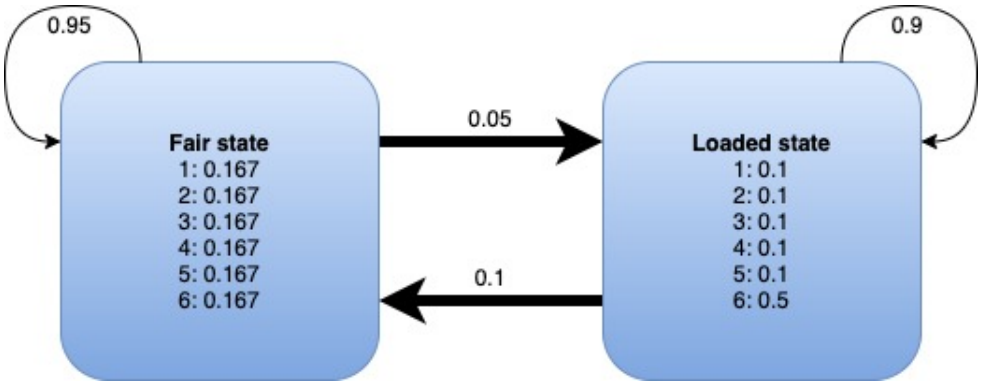


Figure 1: HMM with probabilities used in order to generate sequences for training and testing. Emission probabilities are shown inside each state, while transition probabilities are shown at its corresponding arrow

Running the program described we could now simulate sequences using the model shown in Figure 1. We simulated 100 sequences for training, and 100 additional observations for testing the model trained by the Baum-Welch algorithm. For the 100 test observations, we also extracted the underlying sequences of states that generated the sequences. These state sequences could then be used to compare with the sequences of states generated by our estimated model.

2.2 Baum-Welch Algorithm

The Baum-Welch algorithm is an unsupervised method of estimating the parameters of a given hidden Markov model [4]. The algorithm therefore tries to answer the question of: given a set of sequences and a model architecture, what is the optimal transition probability and emission probabilities for this model. This algorithm will however not necessarily find the optimal probabilities and can converge on a local maximum [4].

To use the Baum-Welch algorithm the probability of choosing specific states at specific time points are needed [5]. However due to the large number of possible paths this is too computationally expensive to calculate directly and therefore one can instead use the forward and backward algorithm to calculate it efficiently using dynamic programming.

2.2.1 Forward Algorithm

The forward algorithm is useful for calculating $P(S_i = k, X_1...X_i|w)$, which is the probability of the having the sequence $X_1...X_i$ and ending at state k at time-point i [6]. This calculation can be done quite efficiently using dynamic programming and equation 1.

$$\alpha_k(i+1) = e_k(x_{i+1}) \sum_l \alpha_l(i) a_{lk} \tag{1}$$

Here the $\alpha_k(i)$ is the probability of observing the sequence and ending in state k at time i, $e_k(x_i)$ is the emission probability of observation x_i in state k, and a_{lk} is the transition probability from state l to state k [7].

An issue of underflow can arise in situations where the sequence is long, since the probability of this sequence will become close to 0. In order to solve this issue we used a scaling factor by using the following equation, where we sum the α over each state.

$$c_i = \sum_k \alpha_k(i) \tag{2}$$

and then scaling the α by this factor

$$\hat{\alpha}_k(i) = \frac{\alpha_k(i)}{c_i} \tag{3}$$

In this way it can be avoided to have small alphas since all alphas now for each time-point will sum to one. Another note is that the probability of the sequence $P(X)$ is now given as the product of the scaling factors, however often this number is also very small, and it is therefore preferable to calculate the $log(P(X))$ [5].

$$log(P(X)) = \sum_i^T log(c_i) \tag{4}$$

2.2.2 Backward Algorithm

This algorithm is similar to the forward algorithm, but instead of calculating the probability up to and including the current position. It instead calculates the probability of the sequence after the current position to the end of the sequence and can be defined as.

$$\beta_k(i) = P(X_{i+1}...X_T|S_i = k, w) \tag{5}$$

Similarly to the forward algorithm this can also be calculated using dynamic programming and solving the following recursive function [7].

$$\beta_k(i) = \sum_l a_{kl} e_l(x_{i+1}) \beta_l(i+1) \tag{6}$$

Similarly to the issues of underflow for the forward algorithm, the backwards algorithm can also be prone to underflow. The solution is the same as before, where we rescale the probabilities individually for each time point by a similar approach to equation 2 and 3.

2.2.3 Training the Hidden Markov model

The first part of the Baum-Welch algorithm requires one to calculate ξ, and requires one to have calculated α and β by using the forward and backward algorithm explained previously.

$$\xi_{kl}(i) = \frac{\alpha_k(i)a_{kl}e_l(x_{i+1})\beta_l(i+1)}{\sum_k \sum_l \alpha_k(i)a_{kl}e_l(x_{i+1})\beta_l(i+1)} \tag{7}$$

$\xi_{kl}(i)$ is the probability of transitioning from state k to state l at time-point i. From this it is possible to derive the $\gamma_k(i)$ by simply taking the sum over state l for all $\gamma_k(i)$ except for $\gamma_k(T)$ [5].

$$\gamma_k(i) = \sum_l \xi_{kl}(i) \tag{8}$$

$\gamma_k(i)$ is the probability of being in state k at time-point i. However the $\gamma_k(T)$ cannot be calculated from this sum and should instead by calculated from the forward-backward values [5].

$$\gamma_k(T) = \frac{\alpha_k(T)\beta_K(T)}{\sum_l \alpha_l(T)\beta_l(T)} \tag{9}$$

Having calculated both ξ and γ the transition probabilities and emission probabilities can now be re-estimated [5].

$$\hat{a}_{kl} = \frac{\sum_r^R \sum_{i=1}^{T-1} \xi_{kl}(i)}{\sum_r^R \sum_{i=1}^{T-1} \gamma_k(i)} \tag{10}$$

$$\hat{e}_k(s) = \frac{\sum_r^R \sum_{i=1, X_i=v_s}^{T} \gamma_k(i)}{\sum_r^R \sum_{i=1}^{T} \gamma_k(i)} \tag{11}$$

In this way it is possible to re-estimate both the transition probabilities between state k and l and the emission probability of observing symbol s at state k. This is done over R number of sequences, which thereby also allows training on multiple sequences, and thereby lowering the risk of overfitting to a single sequence . This process is then done in a iterative manner by calculating ξ and γ (expectation step) and then update the transition and emission probabilities (maximization step) [6] [5].

2.3 Evaluating the model

2.3.1 In terms of model parameters

As explained previously, the Baum-Welch algorithm generates an estimation of the HMM parameters that govern the unfair casino data it was trained on. However, to generate this data in the first place, said parameters had to be predefined and thus have *true* values which predictions can be compared to. For this, we calculate the error in each parameter as the Root Mean Square Error between the predicted and true values over the course of the simulation.

$$RMSE_i = \sqrt{(true - predicted)^2} \tag{12}$$

2.3.2 In terms of predictive ability

Alternatively, the parameter estimates given by the Baum-Welch algorithm can be used to make predictions of the hidden states on an independent test set of sequence data, which is created with the sequence generation algorithm in Section 2.1. For this test set, the underlying sequences of states are also stored in order to allow comparison of the predicted and true states.

The Viterbi algorithm is used to decode the most likely sequence of states for each dice roll sequence given the model. Checking all possible paths is computationally unfeasible, so the Viterbi algorithm uses a dynamic approach instead where the optimal path to time step *i+1* is assumed to contain the optimal path to time step *i*. As such, the probability of being in state l at time *i+1* is calculated from the emission probability at time i + 1, the highest possible probability of the path at time i ending in state k and the transition probability of going from state k to state l [7].

$$P_l(i+1) = p_l(i+1) \cdot \max_k(P_k(i) \cdot a_{kl}) \tag{13}$$

In order to avoid underflow in these calculations, probabilities and the formula above are transformed into logarithmic space.

Keeping track of the transitions between states, it is possible to backtrack and obtain the most probable sequence of states that explains each observed sequence of dice roll outcomes. To compare these with the true underlying states and measure the predictive ability of the HMM, the error is calculated as the proportion of mismatches between the two.

The posterior decoding approach with the forward-backward algorithms can also be used to estimate the posterior probabilities associated to each position and state given the trained model.

$$P(\pi_i = k|x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{\alpha_k(i)\beta_k(i)}{P(x)} \tag{14}$$

Where $\alpha_k(i)$ and $\beta_k(i)$ correspond to the probability of generating the sequence up to including x_i in state k (forward) and the probability of generating the rest of the sequence starting from state k (backward). *P(x)* is the probability of observing that specific sequence.

Once the posterior probability of having a concrete observation in a concrete position is calculated, it is possible to evaluate the probabilistic model given by looking at the probabilities in each position of a sequence and the ability of the model to select one state or the other.

3 Results

3.1 Baum-Welch on the Unfair Casino Problem

A set of 100 sequences was generated using the method described in 2.1. These sequences were then fed to the Baum-Welch algorithm in order to recover the transition and emission probabilities, where the probabilities had been initialized randomly. The resulting model is shown in Figure 2.

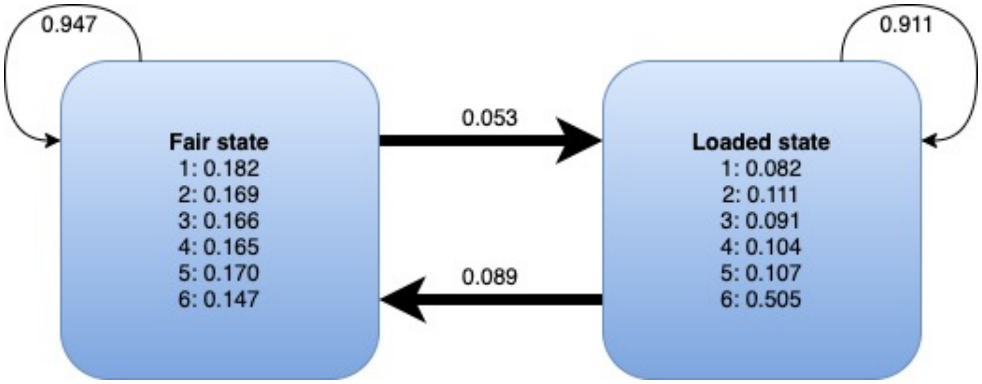


Figure 2: HMM with estimated probabilities after converging using the Baum-Welch algorithm. Emission probabilities are shown inside each state, while transition probabilities are shown at it's corresponding arrow

Comparing the final estimated probabilities with the underlying probabilities used to generate the sequences shown in figure 1, it is clear that the Baum-Welch enabled us to recover the structure of the model, which clearly has a loaded state with a high probability of rolling a six, and a fair state with a roughly equal chance of each observation. This results shows that it is possible to use the Baum-Welch algorithm to train a HMM in an unsupervised manner, thereby not requiring labelled data in order to train a HMM.

3.2 Evaluating the model

3.2.1 Model parameters

As a first step in the model evaluation the RMSE between the true model parameters, i.e. the parameters that generated the data, and the Baum-Welch estimated parameters was computed for every iteration of the algorithm and is presented in Figures 3, 4a and 4b.

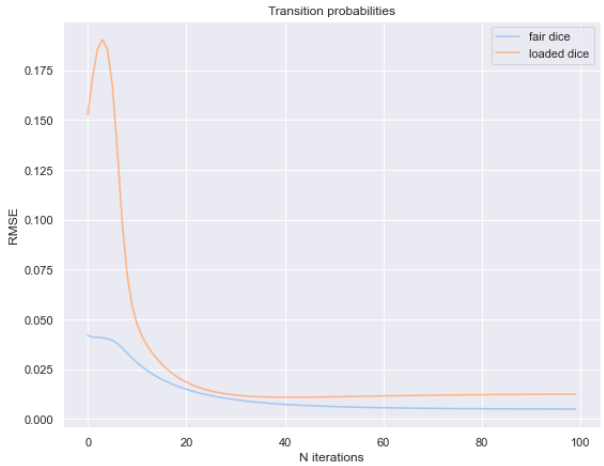
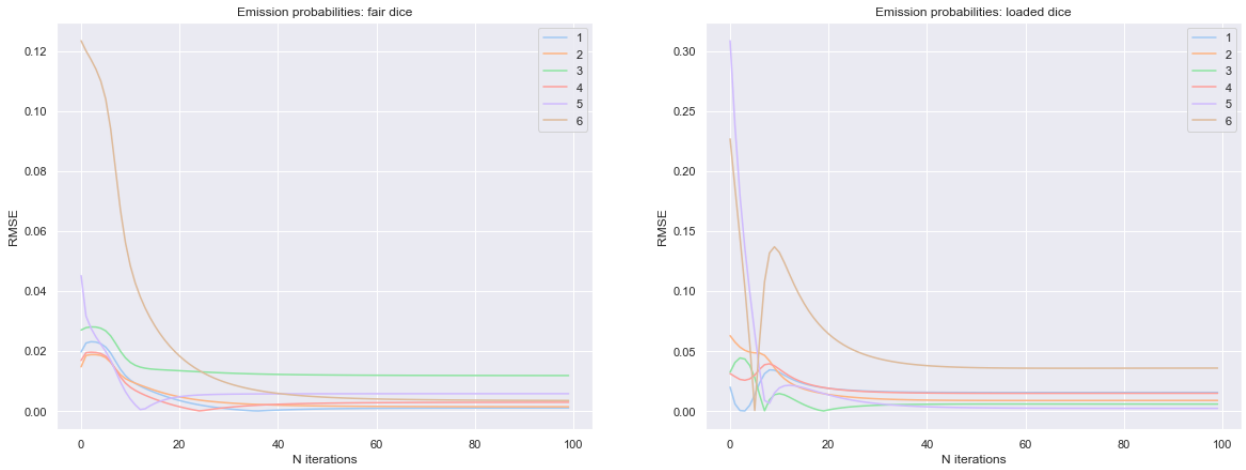


Figure 3: RMSE between the transition probabilities for the fair (blue) and loaded (orange) dices estimated for each iteration of the Baum-Welch algorithm and the true transition probabilities.

As the transition probabilities sum up to one for each state, when one increases, the other one undergoes a reciprocal decrease. The RMSE will therefore have the same value for both probabilities and it suffices with plotting one for each state.

Some fluctuations in the RMSEs can be seen for approximately the first 20 iterations, after which the errors stabilize around 0. It is clear that the model starts with a high RMSE due to the randomly initialized probabilities. However, after some iterations the model starts to learn from the data, and the model parameters become more alike the underlying parameters, thereby reducing the RMSE.



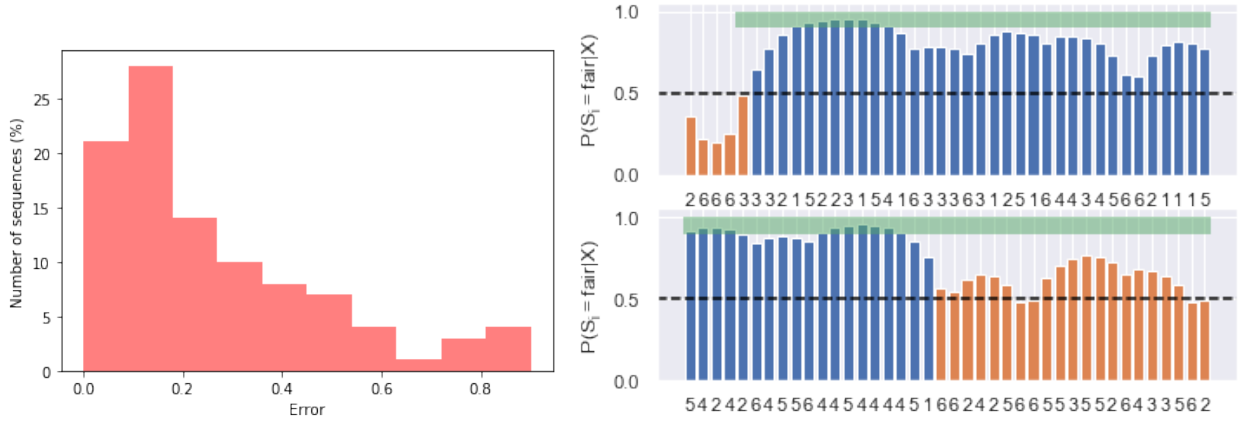
(a) Fair dice. **(b)** Loaded dice.

Figure 4: RMSE between the emission probabilities of the fair (a) and loaded (b) dices estimated for each iteration of the the Baum-Welch algorithm and the true emission probabilities. The legend illustrates to which possible outcome each emission probability corresponds.

3.2.2 Predictive ability of the estimated model

With Viterbi decoding we obtain the most probable sequence of underlying states for the test sequences, which we compare to the true states. As can be seen in Figure 5a, errors range from 0 (perfect prediction) to 0.9 (90% of time-steps are assigned the wrong state) but form a distribution that is clearly skewed towards smaller errors. As point of reference, 30% of the sequences have an error of <0.1, 59% of them have an error of <0.25 and 85%, of <0.5. Given that there are two hidden states in the system, we must remember that an error > 0.5 represents a prediction is worse than random.

From the results of the Posterior decoding we are able to take a closer look at the state probabilities for each time-step and sequence. As examples, two sequences from the test set are shown in Figure 5b. The two chosen sequences show how Viterbi decoding can accurately predict states when their probabilities are close to 1 or 0, but has difficulty doing so when they are closer to 0.5, indicating that many different paths are possible with almost same probability. This will be further discussed in section 4.1.



(a) **(b)**

Figure 5: (a) Error distribution of predicted state sequences of the test set when compared to true states. Errors are shown as proportions and sequence numbers as percentages. (b) Posterior fair dice state probabilities for two sequences as computed by the Posterior decoding algorithm. Vertical bars represent individual time-steps and are tagged with their corresponding dice emissions along the x-axis. Bar colors signify true underlying states, with orange being loaded and blue being fair. Green horizontal bar hovers above time-steps that are predicted by the Viterbi algorithm to be from the fair dice state.

4 Discussion

4.1 Model generation and predictive power

As shown in Section 3.2.1, the Baum-Welch algorithm is capable of estimating the HMM parameters almost perfectly. However, when we decode the hidden states of new data using this generated model, the quality of predictions varies a lot.

By looking at what types of sequences give rise to errors, it is clear that there is a common pattern. One type of error can be seen in the upper sequence of Figure 5b, where regions of transition between states present ambiguous probabilities that inevitably lead to small errors in state selection. Another type of error happens when rare events such as two consecutive state transitions occur. The Viterbi algorithm is unlikely to predict them, since they go against its "most probable path" principle; however, the true sequences can and it will sometimes happen. A third error type leads to longer mispredicted regions and is illustrated in the lower sequence of Figure 5b. With a loaded state that randomly rolls very few 6's and fair state posterior probabilities mostly hovering above 0.5, the Viterbi is more inclined to to incorrectly believe that the second part of the sequence was generated in the fair state.

Using posterior decoding thus shows to give more information about the certainty of the states due to its probabilistic results, in contrast with the Viterbi's binary predictions. Viterbi, on the other hand, will always have a valid path through the states, which the posterior decoding will not [7].

4.2 Adapting to biological sequences

So far we have looked at HMM with the rather simple unfair casino problem. However, the idea behind training the HMM using Baum-Welch can be expanded to more advanced cases with a higher relevance within bioinformatics. In regards to motif finding, the HMM can be seen as an extension of the PSSM [7], where it is possible to also model insertions and deletions in the sequence such as a profile-HMM seen in Figure 6.

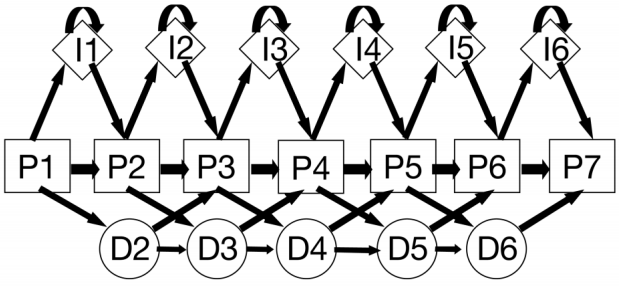


Figure 6: A profile-HMM with 7 match states. Insertion states at the top allow for amino acid insertions, and deletion states allow deletions in the motif [7]

This more advanced type of HMM can be modelled using a multiple alignment and from this calculate the probabilities of each of the states. However, the Baum-Welch algorithm allows one to generate the HMM with just the raw sequences and then train the model using them. This means that the labelling of states done by the multiple alignment is not necessary in order to train the HMM model as shown by Krogh, A. et al [8]. Profile-HMM is one example of how HMMs can be used within sequence analysis, and HMMs are used in many other aspects of sequence analysis such as epitope discovery [9]. Selecting the correct model structure and meaningful training sequences is still a requirement even when using the Baum-Welch approach for training, but using the Baum-Welch algorithm it is possible to use unlabeled data, and thereby simplifying data generation.

When transitioning from the simple unfair casino example to a more advanced biological problem, some tricks can be used to get a more reliable estimate even in cases where sequences are limited. Approaches such as pseudo counts and sequence weighting can further augment the HMM in situations where sequences are limited or the training set contain redundant sequences [7]. These changes let HMM be more robust in cases of small training set and can be used to further extend the idea of training a HMM using the Baum-Welch algorithm used here.

5 Conclusion

Here we have shown how the Baum-Welch algorithm can be used to train a HMM in an unsupervised manner. This was primarily shown through the example of the unfair casino problem, in which one of two dice were used to generate a sequence of rolls, where one had an abnormal probability of rolling a six. We showed that with Baum-Welch it was possible to estimate the probability of the two dice and transition probability for each of the dice from only a list of sequences rolled by these dice. From this we could also partly predict which die was used to roll a given number. The principle of HMM however extends far beyond rolling dice and into the world of biological sequence analysis. Therefore the principle of Baum-Welch training can be used to estimate models for i.e. motif finding without the need of a multiple alignment. This gives the HMM an additional advantage of not needing labelled data besides being able to model insertions and deletions, where a traditional weight matrix would fail.

6 References

[1] "Hidden markov models". In: *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*. Vol. 1-3. Elsevier, Jan. 2018, pp. 753–762. ISBN: 9780128114322. DOI: 10.1016/B978-0-12-809633-8.20488-3.

[2] Walter Zucchini, Iain L MacDonald, and Roland Langrock. *Hidden Markov models for time series: an introduction using R*. CRC press, 2017.

[3] Y. Ephraim and N. Merhav. "Hidden Markov processes". In: *IEEE Transactions on Information Theory* 48.6 (2002), pp. 1518–1569. DOI: 10.1109/TIT.2002.1003838.

[4] Leonard E. Baum et al. "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains". In: *The Annals of Mathematical Statistics* 41.1 (1970), pp. 164–171. ISSN: 0003-4851. DOI: 10.1214/aoms/1177697196.

[5] Lawrence R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286. ISSN: 15582256. DOI: 10.1109/5.18626.

[6] Pierre Baldi and Søren Brunak. *Bioinformatics: The Machine Learning Approach*. 2001.

[7] Ole Lund et al. *Immunological Bioinformatics*. 2018. ISBN: 0262122804. DOI: 10.7551/mitpress/3679.001.0001.

[8] Anders Krogh et al. "Hidden Markov Models in computational biology applications to protein modeling". In: *Journal of Molecular Biology* 235.5 (Feb. 1994), pp. 1501–1531. ISSN: 00222836. DOI: 10.1006/jmbi.1994.1104.

[9] Miguel Lacerda, Konrad Scheffler, and Cathal Seoighe. "Epitope discovery with phylogenetic hidden markov models". In: *Molecular Biology and Evolution* 27.5 (May 2010), pp. 1212–1220. ISSN: 07374038. DOI: 10.1093/molbev/msq008. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2857806/.