Hidden Markov Models

Magnus Karlsson

Background

Hidden Markov chains was originally introduced and studied in the late 1960s and early 1970s. During the 1980s the models became increasingly popular. The reason for this is two-folded. Firstly, the hidden Markov models are very rich in mathematical structure and hence can form the theoretical basis for a wide range of applications. Secondly, the models have, when applied properly, turned out to be highly successful. Some of the notable applications are speech recognition and bioinformatics in particular protein modelling.

In this work, basics for the hidden Markov models are described. Problems, which need to be solved are outlined, and sketches of the solutions are given. A possible extension of the models is discussed and some implementation issues are considered. Finally, three examples of different applications are discussed.

The vast majority of the theoretical results in this work is a summary of the results in Rabiner (1989). The example in speech recognition is due to Rabiner (1989) , the example of protein modelling is due to Krogh et al. (1994) and finally an application in fatigue analysis is due to Johannesson (1999).

What is Hidden Markov Models?

Hidden Markov models (HMM) can be seen as an extension of Markov models to the case where the observation is a probabilistic function of the state, i.e. the resulting model is a doubly embedded stochastic process, which is not necessarily observable, but can be observed through another set of stochastic processes that produce the sequence of observations. To get a better understanding for this the following example might be useful:

Example

Consider a room with N urns. Within each urn there are a large number of coloured balls. We assume that there is M different colours in total. Furthermore, assume that an urn is initially chosen according to some probability distribution. From this urn, a ball is chosen at random, and its colour is recorded as the observation. The ball is then replaced in the urn from which it was selected. A new urn is selected according to a random selection process associated with the current urn.

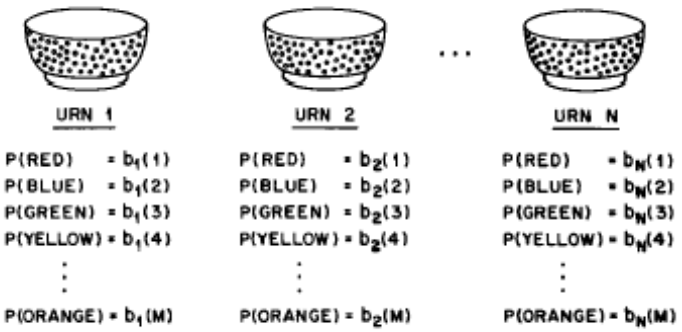

Figure. An N-state urn and ball model, which illustrates the general case of a discrete symbol HMM. From Rabiner (1989).

The ball selection process is repeated for the new urn, after which the next urn is selected according to a selection process associated with the second urn, and so forth. The entire process generates a finite observation sequence of colours, which we would like to model as the observable output of an HMM. We can now see that we have an underlying Markov chain, where each state corresponds to the selection of a particular urn. This chain is however not observable, but can be observed through the sequence of colours which obviously is a probabilistic function of the embedded Markov chain, since a colour is chosen randomly depending on the state which we are currently in, i.e. the urn, which we are currently choosing the ball from.

Description of HMM

Rabiner (1989) suggest that a HMM can be described by the following:

1. N, the number of states in the model. Although the states are hidden, for many practical applications there is often some physical significance attached to the states or to sets of states of the model. In the example with the balls and urns above, N corresponds to the number of urns. We denote the individual states as $S = \{s_1, s_2, ..., s_N\}$, and the state at time n as Z_n.
2. M, the number of distinct observation symbols per state. The observation symbols correspond to the physical output of the system being modelled. In our example above M corresponds to the number of colours of the balls. We denote the individual symbols as $V = \{v_1, v_2, ..., v_M\}$, and the symbol at time n as X_n.
3. The state transition probability matrix $P = \{P_{ij}\}$, where

$$P_{ij} = P(Z_{n+1} = s_j | Z_n = s_i), \qquad 1 \le i, j \le N \tag{1}$$

4. The observation symbol probability distribution in state s_j, $B = \{b_j(k)\}$, where

$$b_j(k) = P(X_n = v_k | Z_n = s_j) \qquad 1 \le j \le N, 1 \le k \le M \tag{2}$$

5. The initial state distribution $\pi = \{\pi_i\}$ where

$$\pi_i = P(Z_0 = s_i), \qquad 1 \le i \le N \tag{3}$$

It can be seen from the above that a complete specification of an HMM requires specification of the two model parameters (N and M), specification of the observation symbols and the specification of the three probability measures P, B and π. For convenience, we use the compact notation

$$\lambda = (P, B, \pi) \tag{4}$$

It should be noted here that the above discussion has considered only the case when the observations is characterised as discrete symbols. In principle, this is however not necessary. The symbols or outputs can be either discrete or continuous, and either scalar or vector-valued. However, in all cases we need to assume that the stochastic process $\{Z_n\}$ is a Markov chain having the property that $X_k^- = \{X_0, ..., X_k\}$ and $Z_{k+1}^+ = \{Z_{k+1}, Z_{k+2}, ...\}$ are conditionally independent given $Z_k^- = \{Z_0, ..., Z_k\}$. We will, however, from now on assume that we have the case with discrete scalar symbols.

Three basic problems for HMMs

In order for the hidden Markov models to be useful in real-world applications Rabiner (1989) presents three basic problems:

Problem 1: Given the observation sequence $X = (x_0, x_1, ..., x_T)$, and a model $\lambda = (P, B, \pi)$, how do we efficiently compute $P(X|\lambda)$, the probability of the observation sequence, given the model?

Problem 2: Given the observation sequence $X = (x_0, x_1, ..., x_T)$, and the model $\lambda = (P, B, \pi)$, how do we choose a corresponding state sequence $Z = (z_0, z_1, ..., z_T)$, which is optimal in some meaningful sense?

Problem 3: How do we adjust the model parameters $\lambda = (P, B, \pi)$ to maximise $P(X|\lambda)$?

Problem 1 can be seen as one of scoring how well a given model matches a given observation sequence, i.e. the solution to this problem would give us a tool to choose between competing models. Problem 2 can be seen as the problem of uncovering the hidden part of the model, i.e. to find the correct state sequence. Problem 3 is the one in which we try to optimise the model parameters so as to best describe how a given observation sequence comes about. The observation sequence used to adjust the model parameters is called a training sequence. The training problem is the most crucial one for most applications of HMMs. We will now move on to some discussion on the mathematical solutions of each of the three problems above.

Solution to problem 1:

The problem is to calculate the probability of the observation sequence given the model λ. It is possible to do this in a straightforward way, but this is unfortunately computationally unfeasible, even for small values of N and T. However, there exists a more efficient procedure called the forward-backward procedure. Consider the forward variable $\alpha_n(i)$ defined as

$$\alpha_n(i) = P(X_1, X_2, ..., X_n, Z_n = s_i|\lambda) \tag{5}$$

i.e., the probability of the partial observation sequence, $(X_1, X_2, ..., X_n)$ until time n and state s_i at time n, given the model λ. We can here use induction for the problem. First for $n = 0$, we have

$$\alpha_0(i) = \pi_i b_i(X_0), 1 \le i \le N. \tag{6}$$

Induction leads to

$$\alpha_{n+1}(j) = \left[\sum_{i=1}^{N} \alpha_n(i) P_{ij}\right] b_j(X_{n+1}), 1 \le i \le N, 1 \le n \le T-1 \tag{7}$$

Since

$$\alpha_T(i) = P(X_1, X_2, ..., X_T, Z_T = s_i|\lambda) \tag{8}$$

it follows that

$$P(X|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{9}$$

Using the forward variable $\alpha_n(i)$ we have now solved the first problem above. (Note that this does not include any backward variable. The backward variable is actually not necessary for the solution and is therefore excluded here, but it will appear in the solution for problem 3.)

Solution to problem 2:

Unlike, problem 1 where an exact solution can be given, there are several possible ways of solving problem 2, i.e. finding the optimal state sequence associated with the given observation sequence. The difficulty comes from the fact that there are several different optimality criteria. One possible optimality criterion is to choose the states Z_n, which are individually most likely. This optimality criterion maximises the expected number of correct individual states, but it does not take into consideration whether the sequence of states is possible. For instance although the transition between two states is impossible i.e. $P_{ij} = 0$ for some i and j, they may still be the most likely at the very instants. This is due to the fact that the solution of this problem simply determines the most likely state at every instant, without considering the probability of occurrence of sequences of states. The most widely used criterion is instead to find the single best state sequence, i.e. to maximise $P(Z|X,\lambda)$, which is equivalent to maximising $P(Z,X|\lambda)$. An algorithm for solving this problem has been found and is called the Viterbi algorithm. This algorithm can simply be seen as the maximum likelihood estimate. The algorithm can be summarised as follows:
To find the best state sequence, $Z = \{Z_0, Z_1, ..., Z_T\}$, for the given observation $X = \{X_0, X_1, ..., X_T\}$, we need to define the quantity

$$\delta_n(s_i) = \underset{Z_0, Z_1, ..., Z_{n-1}}{\arg\max}\ P(Z_0, Z_1, ..., Z_n = s_i, X_0, X_1, ..., X_n|\lambda) \tag{10}$$

i.e. $\delta_n(s_i)$ is the best score (highest probability) along a single path, at time n, which accounts for the first $n+1$ observations and ends in state s_i. By induction we have

$$\delta_{n+1}(s_j) = \left[\max_i \delta_n(s_i) P_{ij}\right] \cdot b_j(X_{n+1}) \tag{11}$$

To actually retrieve the state sequence, we need to keep track of the argument which maximised the above the above equation, for each n and j. We do this with the array $\psi_n(s_j)$. The procedure for finding the best state sequence now follows as:

1) Initialisation:

$$\begin{aligned} \delta_0(s_i) &= \pi_i b_i(X_0), 1 \le i \le N \\ \psi_0(s_i) &= 0 \end{aligned} \tag{12}$$

2) Recursion:

$$\begin{aligned} \delta_n(s_j) &= \left[\max_{1 \le i \le N} \delta_{n-1}(s_i) P_{ij}\right] \cdot b_j(X_n),\ 1 \le n \le T,\ 1 \le j \le N \\ \psi_n(s_j) &= \underset{1 \le i \le N}{\arg\max}\left[\delta_{n-1}(s_i) P_{ij}\right],\ 1 \le n \le T,\ 1 \le j \le N \end{aligned} \tag{13}$$

3) Termination:

$$\begin{aligned} P^* &= \max_{1 \le i \le N}\left[\delta_T(s_i)\right] \\ Z_T^* &= \underset{1 \le i \le N}{\arg\max}\left[\delta_T(s_i)\right] \end{aligned} \tag{14}$$

4) State sequence backtracking:

$$Z_n^* = \psi_{n+1}(Z_{n+1}^*),\ n = T-1, T-2, ..., 1, 0. \tag{15}$$

The best sequence according to the Viterbi algorithm is thus found as $Z^* = (Z_0^*, Z_1^*, ..., Z_T^*)$. It should be noted that apart from the backtracking step the Viterbi algortihm is rather similar to the forward calculation used in problem 1.

Solution to problem 3:

The by far most difficult of the three problems is to determine a method to adjust the model parameters $\lambda = (P, B, \pi)$ to maximise the probability of the observation sequence given the model. This problem is in fact not possible to solve using a finite observation sequence as training data, but we can choose $\lambda = (P, B, \pi)$ such that $P(X|\lambda)$ is locally maximised using an iterative procedure such as the Baum-Welch method. (Equivalent results will be found using the EM method.) We start of with introducing a backward variable $\beta_n(i)$ defined as

$$\beta_n(i) = P(X_{n+1}, X_{n+2}, ..., X_T | Z_n = s_i, \lambda) \tag{16}$$

i.e. the probability of the partial observation sequence from $n+1$ to the end, given the state s_i at time n and the model λ. Again we can solve for $\beta_n(i)$ inductively, as follows:

$$\beta_T(i) = 1, 1 \le i \le N \tag{17}$$

and induction leads to

$$\beta_n(i) = \sum_{j=1}^{N} P_{ij} b_j(X_{n+1}) \beta_{n+1}(j) \quad n = T-1, T-2, ..., 1, 0, 1 \le i \le N \tag{18}$$

In order to describe the procedure for reestimation of HMM parameters, we also define $\xi_n(i, j)$, the probability of being in state s_i at time n and state s_j at time $n+1$, given the model and the observation sequence, i.e.

$$\xi_n(i, j) = P(Z_n = s_i, Z_{n+1} = s_j | X, \lambda) \tag{19}$$

From the definition of the forward and backward variables it follows that we can write $\xi_n(i, j)$ in the form

$$\xi_n(i, j) = \frac{\alpha_n(i) P_{ij} b_j(X_{n+1}) \beta_{n+1}(j)}{P(X|\lambda)} = \frac{\alpha_n(i) P_{ij} b_j(X_{n+1}) \beta_{n+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_n(i) P_{ij} b_j(X_{n+1}) \beta_{n+1}(j)} \tag{20}$$

where the numerator is simply $P(Z_n = s_i, Z_{n+1} = s_j, X|\lambda)$ and the division by $P(X|\lambda)$ gives the desired probability measure. We also need to define $\gamma_n(i)$ as the probability of being in state s_i at time n, given the observation sequence and the model. It follows that

$$\gamma_n(i) = \sum_{j=1}^{N} \xi_n(i, j) \tag{21}$$

If we sum $\gamma_n(i)$ over the time index up to time $T-1$ we get a quantity, which can be interpreted as the expected number of transitions made from state s_i. Similarly, summation of $\xi_n(i, j)$ up to time $T-1$ can be interpreted as the expected number of transitions from state s_i to state s_j. We can also sum $\gamma_n(i)$ over the time index up to time T, which can be

interpreted as the expected number of times in state s_i. Using this, we can get a method for reestimation of the parameters in an HMM. The reestimation formulas can be found as

$$\overline{\pi}_i = \text{expected frequency (number of times) in state } s_i \text{ at time } (t=1) = \gamma_n(i) \tag{22}$$

$$\overline{P}_{ij} = \frac{\text{expected number of transitions from state } s_i \text{ to state } s_j}{\text{expected number of transitions from state } s_i} = \frac{\sum_{n=1}^{T-1} \xi_n(i,j)}{\sum_{n=1}^{T-1} \gamma_n(i)} \tag{23}$$

$$b_j(k) = \frac{\text{expected number of times in state } s_j \text{ and observing symbol } v_k}{\text{expected number of times in state } s_j} = \\ = \frac{\sum_{n=1}^{T} \gamma_n(j) \cdot I_{(X_n = v_k)}}{\sum_{n=1}^{T} \gamma_n(j)}. \tag{24}$$

The reestimation procedure now runs as follows. We define the current model as $\lambda = (P, B, \pi)$, and use that to compute the right-hand side of the above equations, which is put equal to the left-hand side. The left-hand sides are the parameters in the model and this can be used to further improve the model by repeating the procedure until a limiting point is reached.

An Extension of the standard HMMs

There are naturally many extensions to the simple scalar, discrete case, which has been introduced here. One of these interesting extensions of the standard HMMs presented here would be to model state duration, i.e. that the sequence stays in a state for a non-zero amount of time. For the standard HMMs, it can be shown that the inherent duration probability density $p_i(d)$ associated with state s_i, i.e. the probability of d consecutive observations in state s_i is of the form:

$$p_i(d) = P_{ii}^{d-1}(1 - P_{ii}), \tag{25}$$

where P_{ii} is the self-transition coefficient for state s_i. For most applications, this exponential state duration density is inappropriate. Instead, it is preferable to explicitly model duration density in some analytical form. This means that the HMM would run as follows. First an initial state s_i is chosen according to some distribution π_i, and then a duration d_0 is chosen according to the state duration density $p_i(d_0)$. Observations for the observing times $t = 0,..., d_0$ are chosen according to the joint density $b_{Z_0}(X_0, X_1,..., X_{d_0})$. Finally, the next state is chosen according to the state transition probabilities P_{ij}, where $P_{ii} = 0$ since we have determine the state duration to be exactly d_0. The procedure is then repeated for the second state and so forth. It should be noted that for the special case where $p_i(d) = P_{ii}^{d-1}(1 - P_{ii})$, the situation is equivalent to the standard HMM. The formulation with state duration density cannot be directly applied to the solution of the three problems described above, but assuming that entire duration intervals are included in the observation sequence it is possible to find similar solutions to the problems.

It should, however, be noted that there a number of drawbacks with the incorporation of duration densities. One is the increase of computational load. Another noteworthy problem is that, in general, a larger training data set is required, since fewer state transitions are made with this model compared to the standard HMM.

Implementation issues for HMMs

There are a number of details to pay attention to when implementing the HMMs. Examples of these are scaling issues, initial parameter estimates, and insufficient training data. The issues are sketched and some ideas about solutions are given here.

Scaling

In order to see why scaling is of importance when implementing the reestimation procedure, consider the definition of the forward variable $\alpha_n(i)$. It can be seen in the definition that $\alpha_n(i)$ consists of the sum of a large number of terms, each of the form

$$\prod_{s=0}^{n-1} P_{Z_s Z_{s+1}} \prod_{s=0}^{n} b_{Z_s}\left(X_{Z_s}\right) \tag{26}$$

where $Z_n = s_i$. Since each of the factors in the product generally is significantly less than 1 it can be seen that as n starts to get big each term in $\alpha_n(i)$ starts to head exponentially to zero. This means that after a sufficiently long time any computer will run into problems with precision range. For this reason a scaling procedure is necessary. The basic procedure, which is used, is to multiply $\alpha_n(i)$ with a scaling coefficient independent of i, with the goal of keeping the scaled $\alpha_n(i)$ within the dynamic range for each value of n i.e. $0 \le n \le T$ .The suggested scaling in Rabiner (1989) is to multiply $\alpha_n(i)$ with a factor

$$c_n = \frac{1}{\sum_{i=1}^{N} \alpha_n(i)} \tag{27}$$

The scaled coefficients are thus found as

$$\hat{\alpha}_n(i) = c_n \alpha_n(i) \tag{28}$$

A similar scaling is done for the backward variables $\beta_n(i)$ using the same scaling factor, i.e.

$$\hat{\beta}_n(i) = c_n \beta_n(i) \tag{29}$$

It can then be shown that when calculating $\overline{P}_{ij}$ due to cancellations we get the same results when using $\hat{\alpha}_n(i)$ and $\hat{\beta}_n(i)$ instead of $\alpha_n(i)$ and $\beta_n(i)$ respectively. The only really important change in the solutions of the problems listed above comes in the calculation of $P(X \mid \lambda)$, since one cannot simply sum up the $\hat{\alpha}_T(i)$ terms since they are scaled already. However, it turns out that it is still possible to calculate logarithm of $P(X \mid \lambda)$. In the Viterbi algorithm it turns out that no scaling is necessary if one uses logarithms in the four steps of the algorithm. This means that one will arrive at $\log\left(P^*\right)$ rather than P^*, but with less computing and no numerical errors.

Initial parameter estimations

In principal there are no straightforward answer on how to choose the initial estimates of the HMM parameters. It appears as the distribution of the initial distribution π and the transition matrix P is rather insensitive. (For instance, uniform initial estimates can be used.) However for the parameters in B, the initial estimates are crucial, especially in the continuous case, i.e. when the observation symbols come from a continuous distribution. There are a number of suggestions on how to obtain good initial estimates, e.g. manual segmentation of the observation sequence into states with averaging of observations within states, and maximum likelihood segmentation of observations with averaging.

Insufficient training data

An obvious problem with the training of HMM parameters, is that the observation sequence is finite. This means that there is often insufficient numbers of occurrences of the different model events to give good parameter estimates. A natural way of solving this problem is to gather more data, but this often impossible in practical situations and therefore it is necessary to find a technique, which deals with the data at hand. One possible solution is simply to reduce the size of the model, e.g. the number of states, number of symbols per state, etc. However, in many practical situations the nature of the model is given by a physical situation and thus reduction of the model is not possible. A third possibility is to interpolate one set of parameter estimates with another set of parameter estimates from a model for which an adequate amount of training data exists. The idea is to use the training data to design two models, one corresponding to the desired one, and one which is smaller, but for which the training data is sufficient. The smaller model is created by tieing one or more sets of parameters of the initial model together. The final result is obtained by interpolation between the two models. A key issue is to understand how much weight should be put on the initial model and how much on the reduced model. There are however some results on this topic, which can provide an optimal weight.

Applications and Examples

Three examples of very different applications will be given here. The first is the perhaps most classic in the field i.e. speech recognition. The second comes from the biological area, and refers to protein modelling. Finally, a more theoretical result useful in fatigue analysis will be given.

Speech recognition

Arguably, one of the most noteworthy applications of HMMs is speech recognition. The example given here is due to Rabiner (1989) and deals with isolated word recognition. Assume there are in total V words, which are to be recognised and that there are K occurrences of each spoken word. Each occurrence of the word constitutes an observation. The observations of words are typically represented in terms of spectra and/or time signals. In order to do the isolated word recognition, there are two tasks that are necessary to perform:

1. First it is necessary to build HMMs for each word in the vocabulary, i.e. for each word v, we need to estimate the model parameters $\lambda_v = (P_v, B_v, \pi_v)$, which optimise the likelihood of the training set observation vectors for the word.

2. For each unknown word the observation sequence is analysed and calculations of model likelihoods for all possible models, i.e. all possible words, are performed. Finally, the model gives the recognised word as the one with the highest model likelihood.

One of the possible ways to perform the analysis and obtain the observation vector X is to conduct a spectral analysis. A common technique is then to use linear predictive coding (LPC) to extract observation vectors.

Protein modelling

The modelling of proteins is not as unrelated to the case with speech recognition as it first appears. A more general speech recognition when a sequence of words or phonemes is considered can be seen as a pattern recognition task. This is also true for the protein modelling case, where the task is to model a sequence of amino acids, which build up proteins. In fact the words correspond to the 20 amino acids from which protein molecules are constructed. The example of a hidden Markov model for proteins considered here is due to Krogh et al. (1994).

The structural intuition of a protein can be seen in the following way: a) A sequence of positions, each with its on distribution over the amino acids; b) the possibility of either skipping a position or inserting extra amino acids between consecutive positions; and c) allowing for the possibility that continuing an insertion or deletion is more likely than starting one. Krogh et al. (1994) construct their hidden Markov model to catch the properties listed above. The main line of the HMM contains a sequence of M states, which we will call match states, corresponding to the positions in a protein or columns in a multiple alignment. Each of the M states can generate a letter x from the 20-letter amino acid alphabet according to the distribution $P(X = x \mid Z = m_k)$, $k = 1,...,M$, i.e. each generated letter correspond to a specific amino acid. The notation $P(X = x \mid Z = m_k)$ means that each of the match states m_k, $1 \le k \le M$, have distinct distributions. In order to model the possibility of skipping the position there is a deletion state d_k for each state m_k, which is simply a dummy state. Finally, in order to model the possibility of inserting extra amino acids there are a total of $M+1$ insert states to either side of the match states, which generate letters from the amino acid alphabet in exactly the same way as the match state, but use the probability distributions $P(X = x \mid Z = i_k)$, $k = 0,1,...,M$. For simplicity purposes a dummy state has been added in the beginning and the end, denoted m_0 and m_{M+1}, which do not produce any amino acids. The situation can be seen below for the case when $M = 4$.

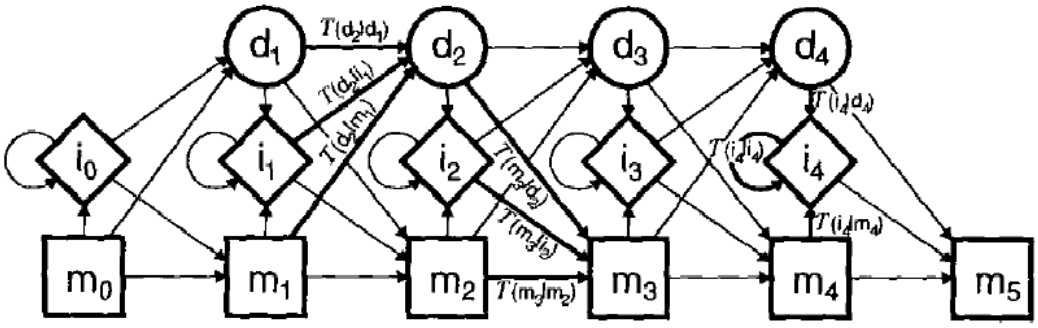

Figure. The protein model for $M = 4$. From Krogh et al. (1994).

Notice that the model allows for several extra amino acids since there is a positive self-transition probability for the insert states. From each state, there are three possible transitions. Transitions into match states or deletion states always move forward in the model whereas transitions into insert states do not. The transition probability from a state q to a state r $P(Z = r \mid Z = q)$ is here denoted $T(r \mid q)$, which corresponds to the more familiar notation P_{rq}.

A sequence from the model is generated in the following way: Starting in the dummy state m_0, choose a transition to m_1, d_1, or i_0 randomly according to the transition probabilities $T(m_1 \mid m_0), T(d_1 \mid m_0)$ and $T(i_0 \mid m_0)$. Whenever we are in an insertion or matching state a letter x corresponding to an amino acid is generated. For instance if we are in state m_k an amino acid is generated according to the probability distribution $P(X = x \mid Z = m_k)$. If on the other hand we are in a deletion state no amino acid is generated. The next state is chosen according to the possible transitions in the current state. The procedure continues until the sequence reaches the state m_k, which is the dummy end state, where no amino acid is generated. The generated sequence $x_1, x_2, ..., x_L$ is now a sequence of letters corresponding to the different amino acids, where the sequence has been found following a path of states $q_0, q_1, ..., q_N, q_{N+1}$, where $q_0 = m_0$ and $q_{N+1} = m_{M+1}$. Since the deletion states does not create any amino acids we can conclude that N (the number of states in a path) is larger or equal to L (the length of the sequence). If q_i is a match or insertion state, we define $l(i)$ to be the index in the sequence $x_1, x_2, ..., x_L$ of the amino acid produced in state q_i. The probability of the event that the path $q_0, q_1, ..., q_N, q_{N+1}$ is taken and the sequence $x_1, x_2, ..., x_L$ is generated is

$$P\left(x_1, ..., x_L, q_0, ..., q_{N+1} \middle| \text{model}\right) = T\left(m_{N+1} \middle| q_N\right) \cdot \prod_{i=1}^{N} T\left(q_i \middle| q_{i-1}\right) P\left(x_{l(i)} \middle| q_i\right) \tag{30}$$

where $P\left(x_{l(i)} \middle| q_i\right) = 1$ if q_i is a deletion state. The probability of any sequence $x_1, x_2, ..., x_L$ of amino acids can be found as the sum over all possible paths that could produce that sequence

$$P\left(x_1, ..., x_L \middle| \text{model}\right) = \sum_{\text{paths } q_0, ..., q_{N+1}} P\left(x_1, ..., x_L, q_0, ..., q_{N+1} \middle| \text{model}\right) \tag{31}$$

A way of estimating the parameters is in the model is the following: For a given set of training sequences $s(1), ..., s(n)$, one can see how well a model fits them by calculating the probability it generates them. This is simply a product of terms of the form given by the sum above, where we for each $j = 1, ..., n$, let $x_1, x_2, ..., x_L = s(j)$. The result is the likelihood function and maximising with respect to the parameters in the model leads to the best model according to the maximium likelihood method.

Fatigue analysis

One of the major reasons for structural failure in the automotive industry is fatigue. Over the years various methods of extracting fatigue relevant data from random load-time histories have been developed. One way of dealing with this problem is to form equivalent load cycles and then use damage accumulation methods, such as the Palmgren-Miner rule. The method that has shown best results is the rainflow cycle counting method. It has become the most commonly used counting method in engineering. The way of constructing the cycles is based on counting hysteresis cycles for the load in the stress-strain plane. A definition suitable for mathematical analysis is the following, first presented by Rychlik (1987):

Definition:

From the *k*:th local maximum (value M_k) one looks at the lowest values in forward and backward directions between M_k and the nearest point at which the load exceeds M_k. The larger (less negative) of those two values, denoted by m_k^{rfc}, is the rainflow minimum paired with M_k, i.e. m_k^{rfc} is the least drop

before reaching the value M_k again on either side. Thus the *k*:th rainflow pair is $\left(m_k^{rfc}, M_k\right)$ and the rainflow range is $H_k^{rfc} = M_k - m_k^{rfc}$.

This definition is probably best understood from a figure:

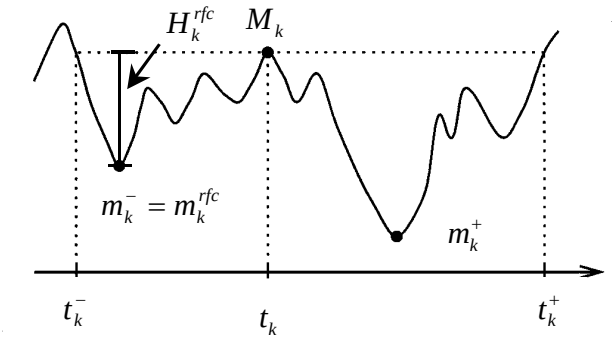

Figure. The definition of the rainflow cycle. The rainflow cycle is denoted H_k^{rfc}.

A vehicle is usually driven in very different environment, for instance it is possible to distinct between driving in curves, slopes, flat straights or performing manuoevers. These cases will create very different sort of loads, not the least because of the differences in speed. One possible way of modelling these loads is by hidden Markov chains, where the states of the underlying Markov chain corresponds to the particular driving mode. The observed load at time n is denoted X_n. For a more complete treatment of this see Johannesson (1999). We can regard the observed load signal $\{X_n\}_{n=0}^{\infty}$ as a random process with the state space $\{v_1,...,v_M\}$, such that a successive value is given by a Markov transition according to one of r possible transition matrices, corresponding to the different driving modes. Which transition matrix to choose is determined by the regime process $\{Z_n\}_{n=0}^{\infty}$ with possible values $1,...,r$. The regime process is assumed to be a Markov chain with the transition matrix $\mathbf{P} = \left(P_{ij}\right)_{i,j=1}^{N}$ having the property that $\chi_n^- = \{X_0,...X_n\}$, $Z_{n+1}^+ = \{Z_{n+1}, Z_{n+2},...\}$ are conditionally independent given $Z_n^- = \{Z_0,...,Z_n\}$. In particular the regime transitions take place independently of the previous X_n values i.e. $\mathbf{P}\left(Z_n = j \middle| Z_{n-1} = i, \chi_{n-1}^-\right)$ $= \mathbf{P}\left(Z_n = j \middle| Z_{n-1} = i\right) = P_{ij}$. The evolution of the process $\{X_n\}_{n=0}^{\infty}$ is described by the transition probabilities $q_{ij}^{(z)} = P\left(X_n = v_j \middle| X_{n-1} = v_i, Z_n = z, \chi_{n-2}^-, Z_{n-1}^-\right) = \mathbf{P}\left(X_n = v_j \middle| X_{n-1} = v_i, Z_n = z\right)$, giving the transition matrices $\mathbf{Q}^{(z)} = \left(q_{ij}^{(z)}\right)_{i,j=1}^{N}$, $z = 1,...,r$. We now have a special case of the standard HMMs where we know that the observation process $\{X_n\}$ is a Markov chain conditioning on the hidden Markov chain $\{Z_n\}$. In this case it is common to call the process $\{X_n\}$ a switching Markov chain (with Markov regime), and call the process $\{Z_n\}$ the regime process. $\{X_n\}$ itself does not satisfy the Markov property, however it can be shown that the joint process $\{(X_n, Z_{n+1})\}_{n=0}^{\infty}$ is a Markov chain, that is

$$\begin{aligned}\mathbf{P}\left((X_n, Z_{n+1}) = (v_n, s_{n+1}) \middle| (X_{n-1}, Z_n) = (v_{n-1}, s_n), ..., (X_0, Z_1) = (v_0, s_1)\right) = \\ = \mathbf{P}\left((X_n, Z_{n+1}) = (v_n, s_{n+1}) \middle| (X_{n-1}, Z_n) = (v_{n-1}, s_n)\right)\end{aligned} \tag{32}$$

The joint process has state space $\{(v_i, z)\}_{i=1,z=1}^{N,r}$ containing $N \cdot r$ states and transition matrix

$$\mathbf{Q} = \left(\mathbf{Q}_{ij}\right)_{i,j=1}^{N}, \text{ where } \mathbf{Q}_{ij} = \left(Q_{ij}(z,w)\right)_{z,w=1}^{r} \text{ and } Q_{ij}(z,w) = q_{ij}^{(z)} p_{zw} \tag{33}$$

The $r \times r$ matrix $\mathbf{Q}_{ij}$ describes a transition from i to j for $\{X_n\}$ where the regime process $\{Z_n\}$ may switch state. For fixed j we can define the column vector $\mathbf{q}$

$$\begin{aligned} &\mathbf{q} = (\mathbf{q}_m),\ \mathbf{q}_m = (q_{m1} \quad q_{m2} \quad \ldots \quad q_{mr})^T, \\ &q_{mz} = \mathbf{P}\left(X_n > v_j \middle| X_{n-1} = v_m, Z_k = z\right) = \sum_{l=j+1}^{N} q_{ml}^{z} \end{aligned} \tag{34}$$

containing the probabilities that $(X_{n-1}, Z_n) = (v_m, z)$ are followed by a value $X_n > v_j$. Let $\boldsymbol{\pi}$ be the stationary distribution of the joint Markov chain, with transition matrix $\mathbf{Q}$ (defined as above), then $\boldsymbol{\pi} = (\boldsymbol{\pi}_i)_{i=1}^{N}$, $\boldsymbol{\pi}_i = (\pi_{i1} \quad \pi_{i2} \quad \ldots \quad \pi_{ir})$. Finally let $\tilde{\boldsymbol{\pi}} = (\boldsymbol{\pi}_1 \quad \boldsymbol{\pi}_2 \quad \ldots \quad \boldsymbol{\pi}_i)$. Now define the following submatrices of $\mathbf{Q}$, $\mathbf{A} = (Q_{ml})$, $i \le m \le j$, $i \le l \le j$ and $\mathbf{C} = (Q_{ml})$, $1 \le m \le i-1$, $i \le l \le j$ for $i = 2,...,n-1$ and $j = i,...,n-1$. The matrix $\mathbf{C}$ contains the probabilities that the process jumps from the interval $[1, i-1]$ to $[i, j]$ and $\mathbf{A}$ that the process stays within the interval $[i, j]$. (For $j = i-1$ we define $\mathbf{A} = 0$ and $\mathbf{C} = 0$.)
Furthermore,define the column vectors $\mathbf{d} = (q_1 \quad q_2 \quad \ldots \quad q_{i-1})^T$, $\mathbf{e} = (q_i \quad q_{i.+1} \quad \ldots \quad q_j)^T$, where $\mathbf{d}$ describes a direct transition from $1,...,i-1$ to a value above j and $\mathbf{e}$ a transition from $i,...,j$ to above j. With this notation Johannesson (1999) showed the following theorem:

Theorem:
For fixed values i and j ($i = 2,...,n$, $j = i-1,...,n-1$), the rainflow counting intensity for the sequence $\{X_k\}$ is given by

$$\mu^{rfc}(i,j) = \tilde{\boldsymbol{\pi}}\left(\mathbf{d} + \sum_{k=0}^{\infty} \mathbf{C}\mathbf{A}^k \mathbf{e}\right) = \tilde{\boldsymbol{\pi}}\left(\mathbf{d} + \mathbf{C}(\mathbf{I} - \mathbf{A})^{-1}\mathbf{e}\right), \tag{35}$$

where the row vector is $\tilde{\boldsymbol{\pi}} = (\pi_1 \quad \pi_2 \quad \ldots \quad \pi_{i-1})$, the column vectors $\mathbf{d}$ and $\mathbf{e}$ are as defined above as well as the sub-matrices $\mathbf{A}$ and $\mathbf{C}$.

The rainflow counting intensity can for instance be used to calculate the expected cumulative fatigue damage caused by a load sequence.

Conclusions

A short summary of the some of the theory behind the hidden Markov models have been given. For understanding purposes, the intention has been to show relatively simple parts of the theory. However, the examples of results and very different kinds of applications showed here still give a hint of the usefulness of the hidden Markov models.

References

P. Johannesson. *Rainflow Analysis of Switching Markov Loads*. PhD thesis, Lund Institute of Technology, Lund, 1999.

A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501-1531, 1994.

L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257-286, 1989.

I. Rychlik. A new definition of the rainflow cycle counting method, *International Journal of Fatigue*, 9:119-121, 1987.