Applying the Hidden Markov Model to Labelling CTI Event Sequences

Chris Davey

Dec 2014

Contents

1 Overview

The Markov Model provides a framework for the determination of state of the current set of observed evidence variables [1]. It models all potential states and the probability of transition into those states given the evidence observed so far. This summary illustrates the usage of the concept applied to the arrival of CTI events (computer telephony interface) in order to superimpose a higher level determination of state labels upon the low level sequence of events (the evidence variables) allowing a mapping between CTI event sequences and State labels to be defined by use of a Hidden Markov Model (HMM) as opposed to the hard coding of control flow within a programming language. This document provides a motivation for the technique and presents a summary of the Hidden Markov Model encoding, and provides an example of methods for training such a model.

2 Motivation

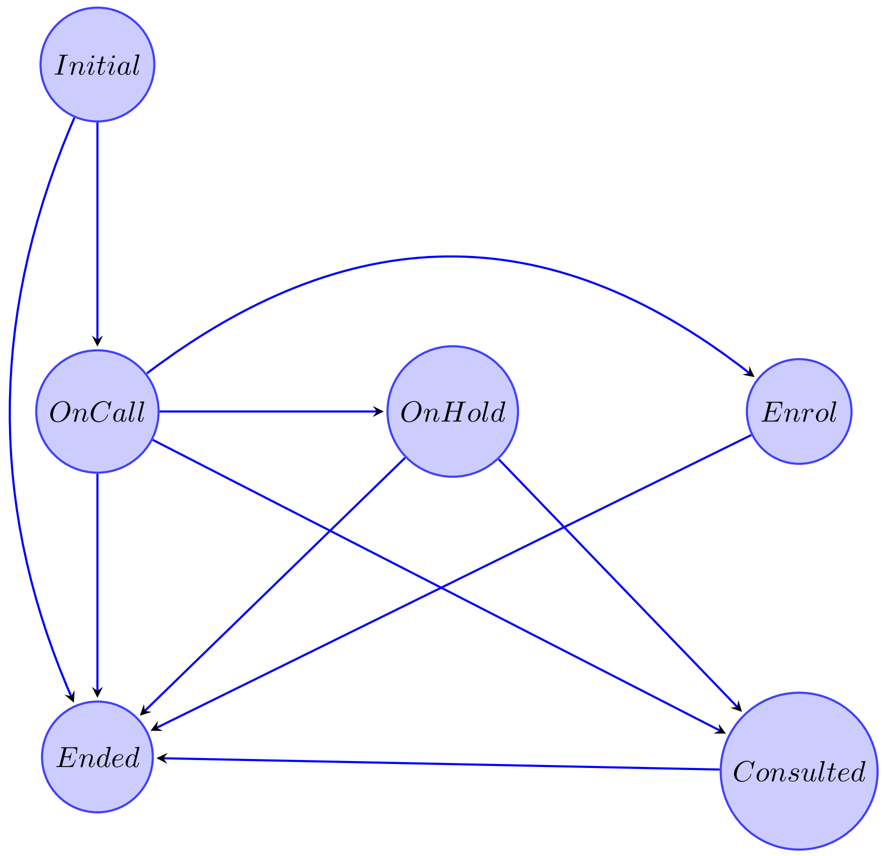
In writing procedural object oriented applications in an imperative language to deal with event streams issued from CTI systems, it has been common practice to encode the sequence of events as a series of imperative branching statements or as a deterministic state machine. These branching statements depend upon state data or object, which may be used to convey a model of the call state and are often intermixed with control statements or triggering of events which enact upon the state data, based upon hard coded assumptions about the CTI event stream. This method of writing a program encodes the interpretation of a time dependent input signal (the series of CTI events) as a series of sequential statements that describes variation in a static manner. These are the logical procedures written within the program that deal with the current state data and the action performed in relation to reacting to the current interpretation of the current CTI event.

When faced with variation in arrival sequence of the CTI events due to the high load placed upon telephony systems under peak load, or performance testing. It is likely that CTI events will not necessarily arrive in the order previously encoded in the procedural sequences encoded in the program. The way to deal with this variation is to add further logical conditions to allow the program to encode the different variations in the order of CTI events, this may involve collecting events in a temporary buffer and delaying them until subsequences of dependent events have arrived, before

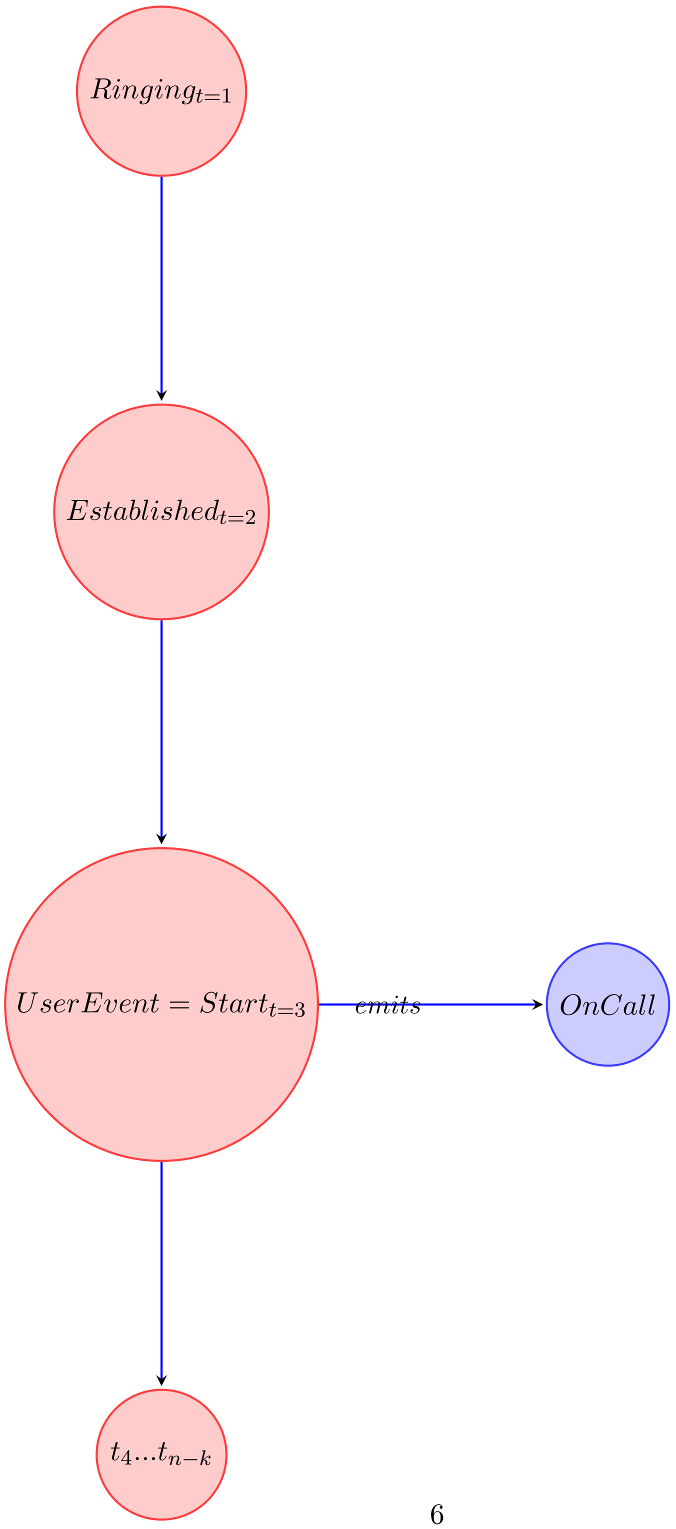
the application can update the call state data based on the interpretation of the dependent event sequences. This adds additional complexity to the program, causing maintenance and potentially leaky abstraction in relation to the programs initial design and structural integrity.
Such transitions between states have also been encoded as state machines, with transitions and explicit entry conditions for each state corresponding to the predetermined rules produced in the assumptions made about the sequence of CTI events. The Hidden Markov Model lends itself to the state machine representation, and is essentially a finite state machine with probabilities assigned to the transitions between states instead of hard coded entry conditions [4].

3 Concepts

This document proposes the use of a Hidden Markov Model that is trained on data sets of labelled CTI event sequences in order to provide a classifier that is capable of emitting "state" labels for time sequences of CTI event streams. Instead of encoding logical branching within the program to describe the relationship between an event stream and an abstracted call state, the HMM is trained to emit state labels in response to the incoming CTI events at each stage of the call. Specifically, the abstract call state is concerned with the modelling of call recording of the calling party (not the answering party). Additionally the answering party (a call centre agent) may potentially transfer the call amongst other call centre agents. The call recording state will need to be started, paused, stopped or restarted depending on the evidence provided by the incoming CTI stream. The abstract call recording states are illustrated below (note these do not reflect any specific implementation, but serve as an illustrative example only in the context of call recording for capturing audio in order to enrol a caller within a voice collection system).



The above call recording states are for illustrative purposes, but demonstrate an example state space without the evidence variables for CTI events, these variables represent the non-observable state variables (labels). The additional evidence variables (combining CTI events and their associated data) are the observable variables that are time dependent and the assocations that define the resulting evidence/state space must be learnt from collected data. For example one such sequence may be.


$Ringing_{t=1}$
$Established_{t=2}$
$UserEvent = Start_{t=3}$
$emits$
$OnCall$
$t_4 ... t_{n-k}$

The diagram above illustrates that at each time t an evidence variable (shown in red) is observed, and at points in the sequence a state variable is emitted (shown in blue) from the model (note those points where a state is not emitted we can call a "Null" state). The state variable identifies the current state of the call given the stream of sequences that have occured so far. The transitions between sequences represent two discrete conditional probability distributions The first relating each state variable X to each of the other state variables x and the second relating each state variable X each evidence variable e given the stream of previous events in time $1:t$, the next predicted state at time $t+k+1$ and prior states in time $t+k$ [1].

$$P(X_{t+k+1}|e_{1:t}) = \sum_{x_{t+k}} P(X_{t+k+1}|x_{t+k})P(x_{t+k}|e_{1:t})$$

Hence the model in the discrete case consists of the state transition probability distribution. From our example the states consist of the set:

$$\{Null, Initial, OnCall, OnHold, Enrol, Ended, Consulted\}$$

Note the addition of the $Null$ state which represents a state that has no action associated with it, this state indicates that the model is waiting for further evidence before it can determine the next appropriate state (to allow the model to emit a state at every incoming CTI event). The state transition table would have the following structure.

$t+1 \sim t$	Null	Initial	OnCall	OnHold	Enrol	Ended	Consulted
Null							
Initial							
OnCall							
OnHold							
Enrol							
Ended							
Consulted							

And is essentially a tabular representation (an $M \times M$ matrix) of the graphical form presented earlier from this table we can determine the prior conditional distribution of $P(X_{t+1}|x_t)$. Duda et al, refer to the transitions in this table as a_{ij} [4], the matrix in Bishop is termed A [3] so we have the matrix A.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1j} \\ \vdots & \vdots & \vdots & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ij} \end{bmatrix}$$

The prediction process described briefly above will update the priors given the time t, the task of learning the state transition table will be to determine the priors from the training data.
The second set of priors is the conditional distribution of the evidence variable at time $t = 1$ and the state variable. For the purpose of the example, the set of evidence variables are listed below, however in actual fact the evidence variables will be a much larger set, note that the evidence variables are a combination of event name, and some type attributes contained within the event, a larger model will also include the ”party” that originates the event in terms of ”party A”, ”party B”, ”party C” for instance. For the purpose of illustration let the CTI evidence variables be the following:

- Ringing(inbound)
- Dialing(consult)
- Establised()
- AttachedDataChanged()
- UserEvent(Started)
- UserEvent(Stopped)
- OffHook()
- Held()
- Retrieve()

Note that the set of evidence variables above are a simplified example, in practice, additional details such as parties may be encoded in the evidence variables.
The resulting joint distribution would have the following structure.

$t = 1$	Null	Initial	OnCall	OnHold	Enrol	Ended	Consulted
Ringing(inbound)							
Dialing(consult)							
Establised()							
AttachedDataChanged()							
UserEvent(Started)							
UserEvent(Stopped)							
OffHook()							
Held()							
Retrieve()							

The format is an $N \times M$ matrix indicating the conditional distribution $P(x_j|e_i)$ denoted as a matrix B where $B_{ij} = P(e_i|x_j)$ at time t the state x_t be emitted given the evidence (or visible) variable e_j. [4]

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1j} \\ \vdots & \vdots & \vdots & \vdots \\ b_{i1} & b_{i2} & \dots & b_{ij} \end{bmatrix}$$

The task of the learning algorithm will be to learn these distributions over t. During the task of prediction, each table represents a prior distribution in state x, and is updated after evaluating the priors. Given that t also represents a sequence ending in state x_t the entire table represents only 1 possible sequence. When working with multiple sequences in training data the training data may represent a set of matrices B_k where k represents the unique sequence of evidence variables. Hence, the initialisation of the training data deals with multiple sequences where the occurance of e_i defines the equality of the sequences. A normalisation process is used during the training phase when working with multiple sequences.

4 Learning

The goal of learning is to produce a hidden markov model containing the following components [5].

- N the total number of distinct states.
- M the total number of distinct evidence variables.
- π the prior state distribution of $P(x_i)$ at $t = 0$.

- A the transition matrix where $a_{ij} = P(x_i|x_j)$ with $i = t, j = t+1$ and $1 \leq i, j \leq N$.
- B_k the state evidence transition matrices (for $1..K$ evidence transition sequences) where $b_{ij} = P(x_j|e_i)$ with $1 \leq i \leq M, 1 \leq j \leq N$.

The HMM is represented as the tuple below [5].

$$HMM = \lambda = (A, B, \pi)$$

The matrices A, B and the vector π are constructed from the initial training data set. Then the learning algorithm is applied until convergence in order to build the updated model.

The question arises how to initialise the initial model parameters. Given the problem domain we will consider state transitions as bernoulli variables and the occurance of evidence variables in a given sequence of length t also as Bernoulli variables. In order to estimate the transition matrix A where $a_{ij} = P(x_j|x_i)$ it is possible to use the count of the frequency of the transition from state x_i to state x_j over the count of all transitions out of state x_i [2].

$$a_{ij} = \frac{c_{ij}}{\sum_k c_{ik}}$$

The initialisation of π can be constructed in a similar manner by the frequency of x_i over the frequency of all states.

$$\pi_i = \frac{c_i}{\sum_k c_k}$$

The initialisation of the matrix B estimates the probability of observing evidence e_i whilst in state x_j at time t where $b_{ij} = P(e_i(t)|x_j(t))$. There are several potential methods of estimating b_{ij} one such method is to consider the occurance of the evidence variable within the sequence as an independent event as well as being independent of the resulting state. The frequency of the evidence variable can then be determined from it's appearance in all sequences.

$$f_i = \frac{c_i}{\sum_k c_k}$$

Then for every sequence in the observed training data the distribution of e_i can be estimated for each state x_t terminating in state x_T using the following.

$$b_{ij,t} = \frac{c(e_i, x_j)}{\sum_k^T c(e_i, x_k)}$$

That is estimated by the frequency of the event whilst in state x_t over the frequency of the event in all states.

The initial step in learning are the state transition distribution and evidence and state distributions as described above, however as t increases the probability distribution becomes a distribution of $P(x_{t+k}|e_{1:t+k})$ for each of the evidence variables the prior needs to be computed in terms of the $1 : t + k$ sequences.

The method of learning is performed through the "forward backward algorithm" which splits the learning process into two parts. Firstly the backward variable represents the probability that the model is in state $x_i(t)$ at time t and will generate the remainder of the given target sequence from $t + 1 \to T$ [4]. It can be described as [5]:

$$\beta_i(t) = P(e_{t+1}, e_{t+2}, ..., e_T|x_i, \lambda)$$

The conditionality includes the λ model generated so far. β is initialised as $\beta_T(i) = 1$ and is calculated as [4]

$$\beta_i(t) = \begin{cases} 0 & \text{if } x_i \neq x_0 \text{ and } t = T, \\ 1 & \text{if } x_i = x_0 \text{ and } t = T, \\ \sum_j^N a_{ij} b_j(e_{t+1}) \beta_j(t+1) & \text{otherwise.} \end{cases}$$

Given the recursive nature of the equation, dynamic programming is used to generate and cache results. The algorithm is evaluated tail recursively. The usage of $b_j(e_{t+1})$ represents the value from B for b_{ij} where i is the index of the e_{t+1} evidence variable in the current model λ.

The second part of the algorithm is defined by the forward variable, which represents the probability that the model is in state x_i given the sequence of evidence variables observed so far $e_{1:t}$ [4].

$$\alpha_j(t) = P(e_1, e_2, ..., e_t, x_i|\lambda) \text{ [5]}$$

The α equation defined as follows [4].

$$\alpha_j(t) = \begin{cases} 0 & t = 0 \text{ and } x_j \neq x_0, \\ 1 & t = 0 \text{ and } x_j = x_0, \\ \left[\sum_i^N \alpha_i(t-1) a_{ij}\right] b_j(e_t) & \text{otherwise} \end{cases}$$

The technique will also require a dynamic programming table to cache the recursive calls when computing α_i. The term $b_j(e_t)$ represents the value

from B for b_{ij} where i is the index of the e_t evidence variable from the current model λ.
The process perform an update for the model A iteratively by estimating the probability of transitioning from state x_i to state x_j at time t and $t+1$ given the model and observations $V^{t+1} = e_1, e_2, e_3, ..., e_{t+1}$ [5] the matrix Ξ is used to contain the new estimates for the transitions between time t and $t+1$.

$$\mathbf{\Xi} = \begin{bmatrix} \xi_{11} & \xi_{12} & \cdots & \xi_{1j} \\ \vdots & \vdots & \vdots & \vdots \\ \xi_{i1} & \xi_{i2} & \cdots & \xi_{ij} \end{bmatrix}$$

The equation for ξ_{ij} is given as follows [5].

$$\xi_{ij} = P(x_i(t), x_j(t+1)|V^{t+1}, \lambda)$$

$$= \frac{\alpha_i(t) b_j(e_{t+1}) \beta_j(t+1)}{P(V^{t+1}|\lambda)}$$

$$= \frac{\alpha_i(t) b_j(e_{t+1}) \beta_j(t+1)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i(t) a_{ij} b_j(e_{t+1}) \beta_j(t+1)}$$

The expected number of times that state x_i transitions to state x_j is determined by summing over the values of ξ_{ij} [4] [5].

$$\gamma_i(t) = \sum_{j=1}^{T} \xi_t(i,j)$$

And the total number of expected times a state x_i is visited is given by summing over γ_i for all time [4] [5].

$$\sum_{t=1}^{T} \gamma_i(t)$$

The matrix A can be updated with new estimates for the state transitions as follows [4].

$$\hat{A} = \hat{a}_{ij} = \frac{\gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

And the matrix B can be updated with new estimates for the evidence variables and states by calculating the ratio between the frequency that a particular evidence variable e_k is emitted and any evidence variable is emitted [4].

$$\hat{B} = \hat{b}_{ij} = \frac{\sum_{t=1,e_k}^{T} \gamma_k(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

The learning process repeats until convergence, that is the changes in previous values of a_{ij} and b_{ij} decreases below a threshold θ [4].

When working with multiple sequences B_k the sequences are assumed to be independent of each other and a normalisation factor is introduced in the estimate for $\bar{A}$ as follows as follows [5].

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_i^k(t) b_j(e_{t+1}^k) \beta_j^k(t+1)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_i^k(t) \beta_i^k(t+1)}$$

The factor $\sum_{k=1}^{K} \frac{1}{P_k}$ is the joint distribution of observation sequence k from the set of observation sequences $(O = e_1, .., e_n)$ for each of the sequences in the input sample B, assuming independence $P(O|\lambda) = \prod_{k=1}^{K} P(O^k|\lambda)$ The normalisation factor also changes the estimate for $\bar{B}$ as follows.

$$\bar{b}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{1=1,e_j}^{T_k-1} \alpha_i^k(t) \beta_{ij}^k(t)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{k=1}^{T_k-1} \alpha_i^k(t) \beta_i^k(t)}$$

The numerator above sums the probability of transition into state x_i for the evidence variable e_j.

5 Prediction

The product of the two conditional distributions at time $t + k$ predicts the state at $t + k + 1$ as stated earlier using the following sum product.

$$P(X_{t+k+1}|e_{1:t}) = \sum_{x_{t+k}} P(X_{t+k+1}|x_{t+k}) P(x_{t+k}|e_{1:t})$$

Similar to the learning process, prediction will iteratively generate estimates for both α and β and use these to successively update the estimates for the model A. A new matrix

$$\mathbf{\Gamma} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1j} \\ \vdots & \vdots & \vdots & \vdots \\ \gamma_{i1} & \gamma_{i2} & \cdots & \gamma_{ij} \end{bmatrix}$$

Is formed by using the estimates to determine the new probability of being in state x_i at time t given the sequence of evidence variable observations $V^t = e_1, e_2, ..., e_t$ and the current model λ [5].

$$\gamma_i(t) = P(x_t|V^t, \lambda)$$

The equation to generate γ_i is given in [5] and [4] as follows.

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_j(t)}{P(x_t|V^t, \lambda)} = \frac{\alpha_i(t)\beta_j(t)}{\sum_{i=1}^{N} \alpha_j(t)\beta_i(t)}$$

Having obtained $\gamma_i(t)$ the most likely state q_t emitted at time t is determined by the state with the maximum probability for $1 \leq i \leq N$ [5].

$$q_t = \ argmax_{1 \leq i \leq N}[\gamma_i(t)], \quad 1 \leq t \leq T.$$

6 Evaluation

TODO: talk about model evaluation using test data and measurement of performance.

7 Related Methods

TODO: talk about relation to the finite state machine, reinforcement learning as a potential means of dynamically learning the policy (it shares a similar structure as the α and β but performs updates in place over time), there are also other models (eg RNN) which can be applied here.

8 Example Implementation

TODO: implementation

References

[1] Norvig, Peter. Russell, Stuart. (2003) *Artificial Intelligence A Modern Approach. Second Edition.* USA: Pearson Education, Inc. pp537-583, 724-733.

[2] Han, Jiawei. Kamber, Micheline. (2006) *Data Mining : Concepts and Techniques. Second Edition.* USA: Morgan Kaufman Publishers. pp518-526.

[3] Bishop, Christopher M. (2006) *Pattern Recognition and Machine Learning* USA: Springer Science + Business Media, LLC. pp607-635.

[4] Duda, Richard O. Hart, Peter E. Stork, David G. (2001) *Pattern Classification. Second Edition.* Canada: Wiley and Sons Inc. pp128-139.

[5] Rabiner, Lawrence R. (1989) "A Tutorial on Hidden Markov Models and Seleted Applications in Speech Recognition". USA: Proceedings of the IEEE, VOL 77, NO.2 pp257-286