Journal of Information Technology Review

DLINE JOURNALS

Print ISSN: 0976-3511
Online ISSN: 0976-2922
JITR 2025: 16 (3)
https://doi.org/10.6025/jitr/2025/16/3/106-112

Incremental Hidden Markov Models for Real-Time Workload Characterization with Improved Backwards Approximations

Tiberiu S. Chis, Peter G. Harrison
Department of Computing, Imperial College London
South Kensington Campus, London, UK

ABSTRACT

The paper introduces an incremental approach to training Hidden Markov Models (HMMs), particularly aimed at modeling discrete-time workloads such as I/O traces. Traditional HMM training, notably using the Baum-Welch algorithm, requires the full dataset in advance, which limits its applicability in real-time applications. To address this, the authors develop IncHMM, a model that updates its parameters incrementally as new data arrives, making it suitable for dynamic workloads.

Central to their method is creating two approximations of the backwards (β) variables in the Forward-Backwards algorithm, which enables the model to process new observations without reprocessing the entire dataset. The first approximation assumes convergence properties in state probabilities, while the second employs matrix inversion techniques with fallback strategies when certain conditions are not met. Both are tested on real-world I/O traces that have been pre-processed using the K-means clustering algorithm.

Simulation results show that the incrementally trained HMMs closely match the statistical properties (mean and standard deviation) of the raw data, especially for read operations. Write discrepancies are noted and attributed to lower variance or model limitations. The authors suggest future improvements, including refining the clustering algorithm and testing it on other time series, such as hospital arrival data, to enhance versatility and accuracy.

Keywords: Hidden Markov Models, Backwards Approximations, Real-time workload

Received: 3 November 2024, Revised 5 February 2025, Accepted 27 February 2025

1. Introduction

A hidden Markov model (*HMM*) is a bivariate Markov chain that captures details regarding the progression of a time series. Initially introduced by Baum and Petrie in 1966 [1], HMM_s can accurately model workloads for

discrete time processes, making them useful as adaptable benchmarks to elucidate and forecast the intricate behavior of these processes. When developing an *HMM*, three primary challenges must be tackled: First, given the model parameters, determine the likelihood that the *HMM* produces a specific sequence of observations, which is addressed using the Forward-Backward algorithm; Second, given a sequence of observations, identify the most probable set of model parameters, achieved through statistical inference via the Baum-Welch algorithm, which incorporates the Forward-Backward algorithm; Third, ascertain the sequence of hidden states that is most likely to produce a sequence of observations, which is resolved through posterior statistical inference in the Viterbi algorithm. In this document, we present an incremental variation of the Baum-Welch algorithm by developing two approximations of the Forward-Backwards algorithm. This approach enables us to process incoming I/O trace data incrementally and update our *HMM* parameters "on-the-fly" as new trace data becomes available. The *HMM* utilizing this incremental Baum-Welch algorithm (Inc*HMM*) yields the necessary parameters to construct a discrete Markov arrival process (*MAP*), which we denote as our Workload Model. For our findings, we validate two Workload Models using averages derived from the raw and Inc*HMM*-generated traces. Lastly, we contrast our findings with existing research in the domain, pinpointing potential enhancements for the future.

2. Forward-Backward Algorithm

The Forward-Backward algorithm, which is used in our incremental Baum-Welch algorithm, solves the following problem: Given the observations $O = (O_1, O_2, \ldots, O_T)$ and the model $\lambda = (A, B, \pi)$[1], calculate $P(O|\lambda)$ (i.e. the probability of the observation sequence given the model), and thus determine the likelihood of O. Based on the solution in [5], we explain the "Forward" part of the algorithm, which is the -pass, followed by the "Backward" part or the β pass. We define the forward variable $\alpha_t(i)$ as the probability of the observation sequence up to time t and of state q_i at time t, given our model λ . In other words, $\alpha_t(i) = P(O_1, O_2, \ldots, O_t, s_t = q_i|)$, where $i = 1, 2, \ldots, N$, N is the number of states, $t = 1, 2, \ldots, T$, T is the number of observations, and s_t is the state at time t. The solution of $\alpha_t(i)$ is inductive:

1. Initially, for $i = 1, 2, \ldots, N$: $\alpha_1(i) = \pi_i b_i(O_1)$

2. Then, for $i = 1, 2, \ldots, N$ and $t = 2, 3, \ldots, T$: $\alpha_t(i) = [\Sigma^N_{j=1} \alpha_{t-1}(j)a_{ji}]b_i(O_t)$ where $\alpha_{t-1}(j)a_{ji}$ is the probability of the joint event that $O_1, O_2, \ldots O_{t-1}$ are observed (given by $\alpha_{t-1}(j)$) and there is a transition from state q_j at time t-1 to state q_i at time t (given by a_{ji}), and also $b_i(O_t)$ is the probability that O_t is observed from state q_i.

3. It follows that: $P(O|\lambda) = \Sigma^N_{i=1}\alpha_T(i)$

where we used the fact that $\alpha_T(i) = P(O_1, O_2, \ldots, O_T, s_T = q_i|\lambda)$.

Similarly, we can define the backward variable, $\beta_t(i)$ as the probability of the observation sequence from time $t + 1$ to the end, given state q_i at time t and the model λ. Then, $\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots O_T | s_t = q_i, \lambda)$ and the recursive solution is:

1. Initially, for $i = 1, 2, \ldots, N$: $\beta_T(i) = 1$

[1] A is the state transition matrix, B is the observation matrix, and π is the initial state distribution.

2. Then, for $i = 1, 2, \ldots, N$ and $t = 2, 3, \ldots, T$: $\alpha_t(i) = \Sigma^N_{j=1}\, a_{ij}\, b_j(O_{t+1})\beta_{t+1}(j)$

where we note that the observation O_{t+1} can be generated from any state q_j.

With the α and β values now computed, we attempt to create an incremental version of the Baum-Welch algorithm, which will use both of these values.

3. Incremental Baum-Welch Algorithm

Given the model $\lambda = (A, B, \pi)$, the Baum-Welch algorithm (*BWA*) trains a *HMM* on a fixed set of observations $O = (O_1, O_2, \ldots, O_T)$. By adjusting its parameters A, B, π, the *BWA* aims to maximise $P(O|\lambda)$. As explained in Section 2.3.2 of [6], the parameters of the *BWA* are updated iteratively by the following formulas:

1. Initially, for $i = 1, 2, \ldots, N$: $\pi'_i = \gamma_1(i)$

2. For A: $a'_{ij} = \dfrac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{j=1}^{N}\sum_{t=1}^{T-1} \xi_t(i,j)}$

3. For B: $b_j(k)' = \dfrac{\sum_{t=1, O_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$

where $\xi_t(i,j) = \dfrac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)}$ and $\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$

We can now re-estimate our model using $\lambda' = (A', B', \pi')$ where $A' = \{a'_{ij}\}$, $B' = \{b_j(k)'\}$ and $\pi' = \{\pi'_i\}$. However, this re-estimation only works on a fixed set of observations, and a useful upgrade for the *BWA* would be to handle infrequent, higher density, additional loads mainly for on-line characterization of workloads [2]. To have an incremental *HMM* automatically updating its parameters as more real time workload data becomes available would achieve this, as well as consistently analyse processes over time in a computationally efficient manner. This new model will be a hybrid between a standard *HMM* and an incremental *HMM* which updates the current parameters A, B, π based on the new set of observations. Therefore, after the standard *HMM* has finished training on its observation set, we aim to calculate the α, β, ξ and γ variables on the new incoming set of observations. For example, if we have trained a *HMM* on T observations and wish to add new observations to update our model incrementally, we notice that $\alpha_{T+1}(i) = [\Sigma^N_{j=1}\, \alpha_T(j)a_{ji}]b_i(O_T)$. Since we possess the values of $\alpha_T(j)$, a_{ji} and $b_i(O_T)$, the new values can be computed quite easily using the forward recurrence formula. However, to find $\beta_{T+1}(i)$ is not so easy as it relies on the backward formula with a onestep lookahead $\beta_{T+1}{}^{(i)} = \Sigma^N_{j=1}\, \alpha_{ji}\, b_j(O_{T+1})\beta_{T+2}(j)$ and unfortunately we do not have $\beta_{T+2}(j)$. Therefore an approximation for the variables is needed, preferably a forward recurrence formula similar to the formula. The new ξ and γ variables (and therefore the new entries a'_{ij} and $b_j(k)'$) can be calculated easily once we have the complete α and β sets. Building on previous work seen in Section 4.8.3 of [6], we attempt to find several new approximations for the β values.

3.1 First β Approximation

The first approximation for the β variables will assume that, at time t and for state i, we have that $\beta_t(i) = \delta(t, i)$ is

a decay function which tends to 0 as $t \to 0$. Therefore, for a sufficiently large observation set and at a sufficiently small t, we obtain the approximate result $\delta(t, i) - (t, j) \approx 0$, where i and j are different states. This then gives the near equality $\delta(t, i) \approx \delta(t, j)$ and hence by our earlier assumption we have the important approximation:

$$\beta_t(i) \approx \beta_t(j) \qquad (1)$$

Let us now transform our β recurrence formula $\beta_t(i) = \Sigma^{N}_{j=1}\, \alpha_{ji}\, b_j(O_{t+1})\beta_{t+1}(j)$ into matrix form, using the notation $b_j = b^j(O_{t+1})$ for ease of use. Since we are using two states in our Workload Model, we set $N = 2$. It then follows that

$$\begin{pmatrix}\beta_t(1)\\ \beta_t(2)\end{pmatrix} = \begin{pmatrix}a_{11}b_1 & a_{12}b_2\\ a_{21}b_1 & a_{22}b_2\end{pmatrix}\begin{pmatrix}\beta_{t+1}(1)\\ \beta_{t+1}(2)\end{pmatrix}$$

then pre-multiply by $(a_t(1)\ a_t(2))$:

$$\begin{pmatrix}\alpha_t(1) & \alpha_t(2)\end{pmatrix}\begin{pmatrix}\beta_t(1)\\ \beta_t(2)\end{pmatrix} = \begin{pmatrix}\alpha_t(1) & \alpha_t(2)\end{pmatrix}\begin{pmatrix}a_{11}b_1 & a_{12}b_2\\ a_{21}b_1 & a_{22}b_2\end{pmatrix}\begin{pmatrix}\beta_{t+1}(1)\\ \beta_{t+1}(2)\end{pmatrix}$$

and multplying out we get

$$\sum_{i=1}^{2}\alpha_t(i)\beta_t(i) = \begin{pmatrix}\alpha_t(1)a_{11}b_1 + \alpha_t(2)a_{21}b_1 & \alpha_t(1)a_{12}b_2 + \alpha_t(2)a_{22}b_2\end{pmatrix}\begin{pmatrix}\beta_{t+1}(1)\\ \beta_{t+1}(2)\end{pmatrix}$$

where by defintion of $\alpha_{t+1}(i)$ it follows that

$$\sum_{i=1}^{2}\alpha_t(i)\beta_t(i) = \begin{pmatrix}\alpha_{t+1}(1) & \alpha_{t+1}(2)\end{pmatrix}\begin{pmatrix}\beta_{t+1}(1)\\ \beta_{t+1}(2)\end{pmatrix}$$

We notice that $\sum_{i=1}^{2}\alpha_t(i)\beta_t(i) = P(O \mid \lambda) = \sum_{i=1}^{2}\alpha_T(i)$ where T is the total number of observations. Quite fittingly, the term $P(O|\lambda)$ is already calculated for us from the -pass. Finally, assuming that $t + 1$ is sufficiently small and using (1) we can deduce that $\beta_{t+1}(1) \approx \beta_{t+1}(2)$, giving us

$$P(O \mid \lambda) \approx \begin{pmatrix}\alpha_{t+1}(1) & \alpha_{t+1}(2)\end{pmatrix}\begin{pmatrix}\beta_{t+1}(1)\\ \beta_{t+1}(1)\end{pmatrix}$$

we then factor out $\beta_{t+1}(1)$

$$P(O \mid \lambda) \approx \beta_{t+1}(1)\begin{pmatrix}\alpha_{t+1}(1) & \alpha_{t+1}(2)\end{pmatrix}\begin{pmatrix}1\\ 1\end{pmatrix}$$

and multiply out the *RHS*

$$P(O \mid \lambda) \approx \beta_{t+1}(1)[\alpha_{t+1}(1) + \alpha_{t+1}(2)]$$

which gives our final approximation result:

$$\beta_{t+1}(1) \approx \beta_{t+1}(2) \approx \frac{P(O \mid \lambda)}{\sum_{i=1}^{2}\alpha_{t+1}(i)} \qquad (2)$$

The β approximation seen in (2) produced very good results in our simulation. To achieve this simulation, we obtained a network trace (aka raw trace) from NetApp servers made up of time stamped *I/O* commands (single Common Internet File System *reads* and *writes*). We then partitioned this raw trace into one second intervals (aka binned trace) counting the number of reads and writes present in each interval or "bin". This binned trace was then filtered through a *K*-means clustering algorithm (assigning 7 clusters, i.e. K=7) and we obtained a discrete time series (aka observation trace) where each point is an integer between 1 and 7. This observation trace was given as a training set of 7000 points (i.e. 7000 seconds) to a *HMM*. Afterwards, 3000 new observations were added to this set, evaluating the 3000 points using our new b approximation. Thus, we were able to create the Inc*HMM*, which stored information on 10000 consecutive observation points. Statistics on a raw trace of 10000 observations were compared with those of an Inc*HMM*-generated trace(using our model parameters *A, B,* π and a random distribution to generate this trace) also of size 10000. The results are summarised below in Figure1:

Reads/bin	Writes/bin
Raw Mean: 111.350	Raw Mean: 0.382
IncHMM Mean: 111.278	IncHMM Mean: 0.366
Raw Std Dev: 254.942	Raw Std Dev: 0.550
IncHMM Std Dev: 255.039	IncHMM Std Dev: 0.461

Figure 1. Statistics for raw and IncHMM traces using the first approximation

Figure 1 is divided into Reads/bin and Writes/bin to simplify analysis, where the bin is simply a one second interval. For example, a "Raw Mean of 111.350 Reads/bin" means that the raw *I/O* trace produced on average 111.350 read commands per second. Similarly, we analyse the average number of writes per second as our *I/O* trace contains both reads and writes. Therefore, we can see from Figure 1 that the statistics for raw reads and Inc*HMM* reads match extremely well, almost identical over the 10000 points. For the writes, there is a higher difference in the standard deviations than in the means. This is possibly due to a significant drop in the number of write procedures presented by the *I/O* trace, which the Inc*HMM* did not reproduce entirely when generating its trace.

3.2 Second β Approximation

As before, we begin with the following vectors and the 2 × 2 transformation matrix (D):

$$\begin{pmatrix}\beta_t(1)\\ \beta_t(2)\end{pmatrix} = \begin{pmatrix}a_{11}b_1 & a_{12}b_2\\ a_{21}b_1 & a_{22}b_2\end{pmatrix}\begin{pmatrix}\beta_{t+1}(1)\\ \beta_{t+1}(2)\end{pmatrix}$$

where we use $b_i = b_i(O_{t+1})$, for ease of notation.
We then pre-multiply by the inverse of the transformation matrix (D^{-1}):

$$\begin{pmatrix}a_{11}b_1 & a_{12}b_2\\ a_{21}b_1 & a_{22}b_2\end{pmatrix}^{-1}\begin{pmatrix}\beta_t(1)\\ \beta_t(2)\end{pmatrix} = I_2\begin{pmatrix}\beta_{t+1}(1)\\ \beta_{t+1}(2)\end{pmatrix}$$

where $D^{-1} D = I_2$ and I_2 is the 2 × 2 identity matrix.

By using Gauss-Jordan elimination to work out D^{-1}, the final equation is

$$\begin{pmatrix}\beta_{t+1}(1)\\ \beta_{t+1}(2)\end{pmatrix} = \frac{1}{b_1 b_2 (a_{11}a_{22} - a_{21}a_{12})} \begin{pmatrix} a_{22}b_2 & -a_{12}b_2 \\ -a_{21}b_1 & a_{11}b_1 \end{pmatrix} \begin{pmatrix}\beta_t(1)\\ \beta_t(2)\end{pmatrix}$$

where $b_1 \neq 0$, $b_2 \neq 0$ and $a_{11}a_{22} \neq a_{21}a_{12}$.

In the case that $b_i = 0$ for a state i, D has a column of all zero values, which means that D^{-1} cannot exist, and therefore a simple approximation for $\beta_{t+1}(i)$ is needed here. Considering all three cases, we present the full set of equations in (3). Underneath this, Figure 2 summarises the results of the simulation with the β approximation from (3):

$$\begin{pmatrix}\beta_{t+1}(1)\\ \beta_{t+1}(2)\end{pmatrix} = \begin{cases} \begin{pmatrix} 1.0 \\ \frac{\beta_t(2)}{a_{22}b_2} \end{pmatrix}, & \text{if } b_1 = 0 \\ \begin{pmatrix} \frac{\beta_t(1)}{a_{11}b_1} \\ 1.0 \end{pmatrix}, & \text{if } b_2 = 0 \\ D^{-1} \begin{pmatrix}\beta_t(1)\\ \beta_t(2)\end{pmatrix}, & \text{if } b_1 = 0, b_2 = 0, a_{11}a_{22} = a_{21}a_{12} \end{cases} \tag{3}$$

Reads/bin	Writes/bin
Raw Mean: 111.350	Raw Mean: 0.382
IncHMM Mean: 110.231	IncHMM Mean: 0.357
Raw Std Dev: 254.942	Raw Std Dev: 0.550
IncHMM Std Dev: 254.155	IncHMM Std Dev: 0.463

Figure 2. Statistics for raw and Inc*HMM* traces using the second approximation

The results obtained were satisfying, including the reads which performed very well. In comparison, the writes slightly underperformed, possibly due to the read-dominated trace or perhaps a slight misjudgement by our clustering algorithm.

4. Conclusion and Future Work

The β approximations used in this paper have been successful after statistical comparisons between raw and Inc*HMM*-generated traces. Thus, we have created two Workload Models (each with their own β approximation) which characterize data traces incrementally. Analysing current work in this field, for example Stenger et al.

in 2001 [4] (where all new β variables were given a value of 1), it is clear that either Workload Model provides a better β approximation. When comparing our models with the incremental *HMM* from [3], all three models produced accurate results, except the latter had a backward formula that was not recursive in terms of the β values. A general improvement to our models would be to increase the accuracy for the standard deviation of the Inc*HMM* writes. This may be achieved by using significantly more observations from our *I/O* trace to obtain a greater variation in write entries. Perhaps adjusting the *K* parameter for our *K*-means clustering algorithm might also improve our results. Finally, we could test the Inc*HMM* with another discrete time data trace, for example using a binned trace of hospital arrival times which stores the number of patients arriving every hour. Then, by choosing the most accurate β approximation of the two, we would obtain an incremental Workload Model capable of analysing a variety of discrete time series.

References

[1] Baum, L. E., Petrie, T. (1966). Statistical inference for probabilistic functions of finite Markov chains. *The Annals of Mathematical Statistics, 37*, 1554–1563.

[2] Harrison, P. G., Harrison, S. K., Patel, N. M., Zertal, S. (2012). Storage workload modelling by Hidden Markov Models: Application to flash memory. *Performance Evaluation, 69*, 17–40.

[3] Florez-Larrahondo, G., Bridges, S., Hansen, E. A. (2005). Incremental estimation of discrete Hidden Markov Models on a new backward procedure. Department of Computer Science and Engineering, Mississippi State University, Mississippi, USA.

[4] Stenger, B., Ramesh, V., Paragios, N., Coetzee, F., Buhmann, J. M. (2001). Topology-free Hidden Markov Models: Application to background modeling. In *Proceedings of the International Conference on Computer Vision* (pp. 297–301).

[5] Rabiner, L. R., Juang, B. H. (1986). An introduction to Hidden Markov Models. *IEEE ASSP Magazine, 3*(1), 4–16.

[6] Chis, Tib. (2011). *Hidden Markov Models: Applications to flash memory data and hospital arrival times* (Master's thesis). Department of Computing, Imperial College London, London, UK.