Learning Hidden Markov Models Using Probabilistic Matrix Factorization

Ashutosh Tewari and Michael J. Giering

Abstract Hidden Markov Models (HMM) provide an excellent tool for building probabilistic graphical models to describe a sequence of observable entities. The parameters of a HMM are estimated using the Baum–Welch algorithm, which scales linearly with the sequence length and quadratically with the number of hidden states. In this chapter, we propose a significantly faster algorithm for HMM parameter estimation. The crux of the algorithm is the probabilistic factorization of a 2D matrix, in which the (i, j)th element represents the number of times the jth symbol is found right after the ith symbol in the observed sequence. We compare the Baum–Welch with the proposed algorithm in various experimental settings and present empirical evidences of the benefits of the proposed method in regards to the reduced time complexity and increased robustness.

1 Introduction

Hidden Markov Model (HMM) is a graphical model that can be used to describe a sequence of observed events/symbols. HMMs are most commonly applied in speech recognition, computational linguistics, cryptanalysis and bio-informatics [3, 7, 11]. An Expectation–Maximization (EM) algorithm proposed by Baum et al. [1], is frequently used for HMM parameter estimation. This algorithm locally maximizes the likelihood of observing the symbol sequence given the parameter estimates.

The motivation for this work arises from situations where application of HMM is ideal but impractical because of the excessive demands on computational resources. For example, Ref. [6] points out the high computational cost of HMM training in the

A. Tewari (✉) · M. J. Giering
United Technologies Research Center, East Hartford, CT 06108, USA
e-mail: tewaria@utrc.utc.com

M. J. Giering
e-mail: gierinmj@utrc.utc.com

K. Yada (ed.), *Data Mining for Service*, Studies in Big Data 3,
DOI: 10.1007/978-3-642-45252-9_3, © Springer-Verlag Berlin Heidelberg 2014

context of intrusion detection because of long sequence lengths. The reason for this computational complexity comes from the way the Baum–Welch algorithm repeatedly performs computation at every time step of the sequence. In this chapter, we propose an alternate method to estimate HMM parameters, in which the information content of the observed sequence is condensed into a 2D matrix. Thereafter, the algorithm operates on this matrix, thus eliminating the need of computation at every time step of the sequence in each iteration. We propose a generative model to explain the information contained in the count matrix and show how the HMM parameters can be derived from the proposed generative model. It should be noted that unlike the Baum–Welch algorithm, the parameters estimated by the proposed algorithm can be suboptimal in the sense of the observing entire symbol sequence. In a closely related but concurrent and independent work [9], the author proposes non-negative matrix factorization [10] based estimation of the HMM parameters (NNMF–HMM). Therefore, we benchmark our results not just with the Baum–Welch but also with the NNMF–HMM algorithm and demonstrate several orders of magnitude speed gains using synthetic and real life datasets. We refer to our algorithm as PMF–HMM, where PMF stands for probabilistic matrix factorization.

The paper is organized as follows. In Sect. 2, we present some background on HMM and set the notations to be used in the rest of the paper. We formulate the problem of HMM parameter estimation using the PMF–HMM algorithm in Sect. 3. The experimental results are provided in Sect. 4, followed by concluding remarks in Sect. 5.

2 Hidden Markov Model

In a HMM, an observed symbol sequence of length T, $O = o_1 o_2 \ldots o_T$, is assumed to be generated by a hidden state sequence of the same length, ($Q = q_1 q_2 \ldots q_T$), as shown in Fig. 1. The hidden states and the observed symbols can take values from finite sets $S = \{S_1, S_2, \ldots, S_N\}$ and $V = \{V_1, V_2, \ldots, V_M\}$, respectively. At each time step, the current state emits a symbol before transitioning to the next state and the process is repeated at every time step. Typically, the number of hidden states (N) is smaller than the number of observed symbols (M). The probability of transitioning from kth to lth hidden state, in successive time steps, is denoted as $P(q_{(t+1)} = S_l | q_t = S_k)$ or simply $P(S_l|S_k)$. The probability of emitting the jth observed symbol from the kth hidden state is given by $P(o_t = V_j | q_t = S_k)$ or $P(V_j|S_k)$. The combined parameter set can be represented as $\lambda = \{P(S_l, S_k), P(V_j|S_k)\}$. Essentially, any probabilistic model with parameters λ that satisfy the properties that $\sum_{l=1}^{N} P(S_l|S_k) = 1$ and $\sum_{j=1}^{M} P(V_j|S_k) = 1$, can be interpreted as a HMM [2]. Rabiner [12] in a comprehensive review on HMMs, points at three basic problems of interest in HMMs:

1. How to efficiently compute the probability of observed symbol sequence, $P(O|\lambda)$, given the model parameters, λ?

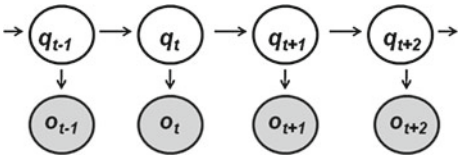

Fig. 1 Generative process of a Hidden Markov Model. The *grey* and *white nodes* represent the observed symbols and hidden states, respectively

2. Given an observed symbol sequence, O, and the model parameters, λ, how do we choose a hidden state sequence which is optimal with respect to some metric?
3. Given observed symbol sequence, O, how do we estimate the parameters, λ, such that $P(O|\lambda)$ is maximized?

The third problem of HMM parameter estimation is more challenging than the other two as finding the global optimum is computationally intractable. The Baum–Welch algorithm solves this problem and guarantees attainment of a local optimum of the observed sequence likelihood. In this chapter, we propose a faster method to estimate the HMM parameters, which also provides a locally optimal solution but for a different objective function. In Sect. 3, we provide the mathematical formulation of the PMF–HMM algorithm.

3 PMF–HMM Algorithm

3.1 Problem Formulation

Let $P(V_i, V_j)$ represent the bivariate mass function of observing the symbol pair $\langle V_i, V_j \rangle$ in an HMM process. The empirical estimate, $\hat{P}(V_i, V_j)$, of this bivariate mass function can be derived from the observed symbol sequence using Eq. (1).

$$\hat{P}(V_i, V_j) = \frac{1}{(T-1)} \sum_{t=1}^{T-1} \mathbb{I}_{V_i}(q_t) \times \mathbb{I}_{V_j}(q_{t+1}) \tag{1}$$

where the indicator function, $\mathbb{I}_{V_i}(q_t)$, is a binary function which outputs 1 only when the observed symbol at time t is V_i. The square matrix $\hat{P}(V_i, V_j)$, which is the maximum likelihood estimate of the bivariate mass function $P(V_i, V_j)$, contains the normalized frequency with which different symbol pairs appear in the sequence O. Consider a process that can generate such pairs of symbols.

- The current observed symbol V_i, at some arbitrary time, makes a transitions to the hidden state S_k with a probability $\hat{P}(S_k|V_i)$.
- In the next time step, the kth hidden state emits the observed symbol V_j with a probability $\hat{P}(V_j|S_k)$.

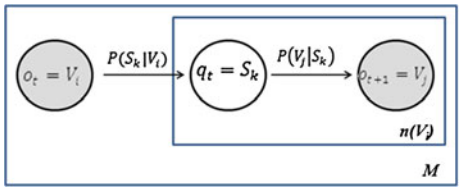

Fig. 2 The proposed graphical model that governs the generation of the pairs of symbols in the observed sequence. The *grey* and *white nodes* represent the observed symbols and hidden states, respectively. M is the total number of observed symbols and $n(V_i)$ is the number of times symbol V_i appears in the sequence

This process of generating all pairs of observed symbols is depicted as a graphical model in Fig. 2. It should be noted that this process is fundamentally different from the generation process of observed symbols in a HMM (Fig. 1). Based on the graphical model in Eq. (2), $\hat{P}\left(V_i, V_j\right)$ can be factorized as:

$$\hat{P}\left(V_i, V_j\right) \approx \hat{P}(V_i) \sum_{k=1}^{N} \hat{P}(S_k|V_i)\hat{P}(V_j|S_k) \tag{2}$$

where, $\hat{P}(V_i)$ is the marginal distribution of observed symbols, which can be estimated empirically as $\hat{P}(V_i) = \sum_j \hat{P}(V_i, V_j)$. In Sect. 3.2, we demonstrate how a fairly popular algorithm in the field of text-mining can be used to perform this factorization to estimate the remaining two parameters $\hat{P}(S_k|V_i)$ and $\hat{P}(V_j|S_k)$.

3.2 Probabilistic Factorization of Count Matrix

Hofmann proposed an EM algorithm for the probabilistic factorization of word count matrices in the field of text mining [4, 5]. In his seminal work, a count matrix was defined on a text corpus (a collection of documents) such that the entries represented the frequencies of the occurrence of different words (from a finite dictionary) in different documents present in the corpus. Hofmann's model, known as *Probabilistic Latent Semantic Analysis* (PLSA), is a widely used method to perform automated document indexing. Although PLSA was proposed to factorize the word-count matrices, it is applicable to any matrix having the co-occurrence information about two discrete random variables. The key assumption in PLSA is the conditional independence of a word and a document given the latent/hidden topic. The generative model shown in Fig. 2, has the same assumption i.e. a pair of observed symbols occur in a sequence, independently, given the in-between hidden state. As a result, the EM algorithm, proposed by Hofmann, renders itself available to perform the factorization shown in Eq. (2). The algorithm is implemented iteratively to estimate the model parameters $\hat{P}(V_j|S_k)$ and $\hat{P}(S_k|V_i)$ using the following steps:

E Step: In this step, the probability distribution of the hidden states is estimated for every pair of observed symbols given the current parameter estimates.

$$\hat{P}(S_k|V_i, V_j) = \frac{\hat{P}(S_k|V_i)\hat{P}(V_j|S_k)}{\sum_{k=1}^{N} \hat{P}(S_k|V_i)\hat{P}(V_j|S_k)} \tag{3}$$

M Step: In this step, the model parameters are updated from the probabilities estimated in the *E step*.

$$\hat{P}(V_j|S_k) = \frac{\sum_{i=1}^{M} \hat{P}(V_i, V_j) \times \hat{P}(S_k|V_i, V_j)}{\sum_{i=1}^{M}\sum_{j=1}^{M} \hat{P}(V_i, V_j) \times \hat{P}(S_k|V_i, V_j)} \tag{4}$$

$$\hat{P}(S_k|V_i) = \frac{\sum_{j=1}^{M} \hat{P}(V_i, V_j) \times \hat{P}(S_k|V_i, V_j)}{\sum_{j=1}^{M} \hat{P}(V_i, V_j)} \tag{5}$$

This EM process converges, after several iterations, to a local optimum that maximizes the log-likelihood function given by Eq. (6).

$$\ell = \sum_{i=1}^{M}\sum_{j=1}^{M} \hat{P}(V_i, V_j)\left(\hat{P}(V_i)\sum_{k=1}^{N} \hat{P}(S_k|V_i)\hat{P}(V_j|S_k)\right) \tag{6}$$

where, $\hat{P}(V_i, V_j)$ is the empirical estimate of the bivariate mass function of a pair of observed symbol given by Eq. (1), while, the term in the bracket is the same bivariate mass function but estimated assuming the generative model shown in Fig. 2. It can be shown that the maximization of the log-likelihood function (Eq. (6)) amounts to the minimization of *Kullback–Leibler* distance between the two joint mass functions i.e. $D_{KL}\left(\hat{P}(V_i, V_j)||\hat{P}(V_i)\sum_{k=1}^{N} \hat{P}(S_k|V_i)\hat{P}(V_j, S_k)\right)$.

3.3 Estimation of HMM Parameters

The HMM parameters consists of the emission probabilities, $P(V_j|S_k)$, and transition probabilities, $P(S_l|S_k)$. The emission probabilities, $P(V_j|S_k)$, are directly estimated in the *M Step* (Eq. 4) of the EM algorithm. However, the transition probabilities do

not get estimated in the proposed generative model. Nevertheless, these probabilities can be obtained using a simple trick. To get the transition probability from kth to lth hidden states, we can enumerate all the possible paths between these two states (via all observed symbols) and aggregate the probabilities of all such paths, as shown in Eq. (7).

$$P(S_l|S_k) = \sum_{i=1}^{M} P(V_i|S_k)P(S_l|V_i) \tag{7}$$

Here we list four key differences between the Baum–Welch algorithm and the PMF–HMM algorithm for the HMM parameter estimation.

- Baum–Welch operates on the entire symbol sequence, while the later operates on the count matrix derived from the symbol sequence.
- The number of parameters that are estimated in the PMF–HMM algorithm is $2MN$ while the Baum–Welch estimates $N(M + N)$ parameters.
- Baum–Welch maximizes the likelihood of the entire observed sequence given the model parameters i.e. $P(O|\lambda)$ as opposed to Eq. (6), which is maximized by the PMF–HMM algorithm.
- The time complexity of PMF–HMM is, $O(T) + O(IM^2N) \approx O(T)$, for very long sequences, while for Baum–Welch algorithm the complexity is $O(IN^2T)$. The symbol I denotes the number of iterations of the respective EM algorithms.

In Sect. 4, we experimentally show that despite having these differences, the PMF–HMM algorithm estimates the HMM parameters fairly well.

3.4 Non-Degenerate Observations

In a HMM, an observed symbol can be expressed as a degenerate probability mass function supported on the symbol set. At any arbitrary time, the mass will be concentrated on the symbol that is being observed. Consider a scenario when there exists some ambiguity about the symbol being observed. This ambiguity can cause the probability mass to diffuse from one symbol to others resulting in a non-degenerate mass function. Such a situation can arise during the discretization of a system with continuous observations. The proposed algorithm, which operates on the count matrix, is inherently capable of handling this type of uncertain information. Figure 3, juxtaposes the scenarios when the observations are degenerate and non-degenerate respectively, for a system with six observed symbols. For the former case, the system makes a clean transition from 3rd to 4th symbol. The outer product of the two probability mass functions in successive time steps results in a 6×6 matrix with a single non-zero entry at (3, 4)th position. To obtain the count matrix for the entire sequence of length T, the outer products in successive time steps can be aggregated as shown in Eq. (8), which is equivalent to Eq. (1).

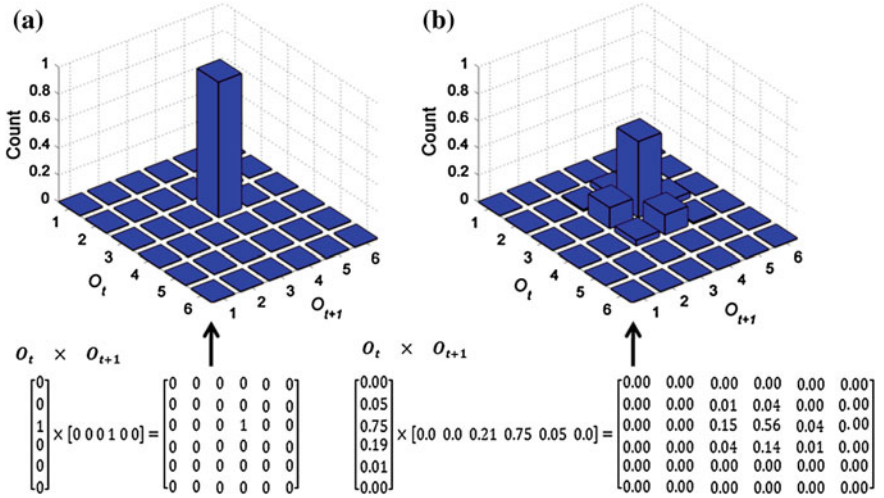

Fig. 3 **a** Generation of a count matrix by degenerate observations. The count value is localized at a single position in the matrix. **b** Generation of a count matrix by non-degenerate observations. The count value gets distributed in a neighborhood

$$\hat{P}(V_i, V_j) = \frac{1}{T-1} \sum_{t=1}^{N-1} O_t \otimes O_{t+1} \tag{8}$$

For the non-degenerate case, the count value simply gets diffused from the (3, 4)th position to the neighboring positions as shown in Fig. 3b. Nevertheless, Eq. (8) can still be used to compute the count matrix. Once the count matrix is obtained, the PMF–HMM algorithm can be applied to estimate the parameters.

4 Experiments

In this section, we present some empirical evidence of the speed gains of the PMF–HMM over the Baum–Welch algorithm, using synthetic and real-life datasets.

4.1 Synthetic Data

We kept the experimental setup identical to the one proposed in Ref. [9]. This provided us with a platform to benchmark our algorithm not just with the Baum–Welch but also with NNMF–HMM algorithm. The experiments were carried out in the MATLAB's programming environment. For the implementation of the Baum–Welch algorithm,

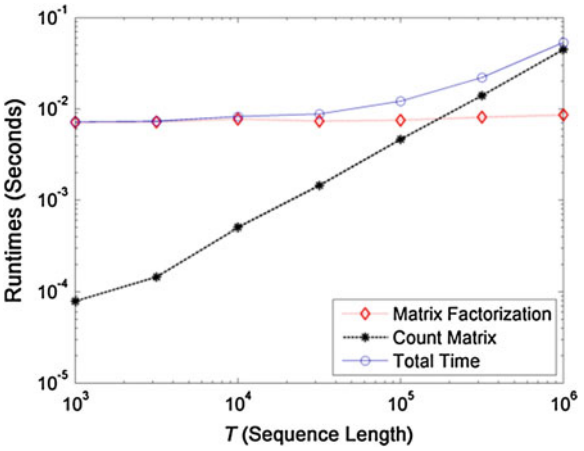

Fig. 4 Plot of the run times of the PMF–HMM algorithm versus the sequence lengths. The total time is split into its two components (1) time spent in computing the count matrix (2) time spent in the probabilistic factorization of the count matrix

we used the Statistical toolbox of MATLAB. The observed symbol sequences were generated using a hidden Markov process with the transition probabilities as shown in Eq. (9).

$$P(S_k|S_l) = \begin{pmatrix} 0 & 0.9 & 0.1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \tag{9}$$

The first and second hidden states randomly generated numbers from Gaussian distributions $\phi(11, 2)$ and $\phi(16, 3)$ respectively, while the third state generated numbers by uniformly sampling from the interval $(16, 26)$. These emission probabilities are listed in Eq. (10).

$$P(V_j|S_k) = \begin{cases} \phi(11, 2) & \text{if } k = 1, \\ \phi(16, 3) & \text{if } k = 2, \\ U(16, 26) & \text{if } k = 3. \end{cases} \tag{10}$$

The continuous observations were rounded to nearest integer to form a discrete symbol sequence. Seven different sequence lengths, $T = 10^{3+0.5x}; x = 0, 1, \ldots, 6$, were chosen for the experiments. For each sequence length, the HMM parameters were estimated with the PMF–HMM algorithm. Figure 4 plots the run times of the algorithm at different sequence length. The total runtime is split into its two constituent times 1) the time taken for populating the count matrix 2) the time taken to factorize the count matrix. As expected, the time taken for populating the count matrix varies linearly with the sequence length as indicated by the unit slope of the

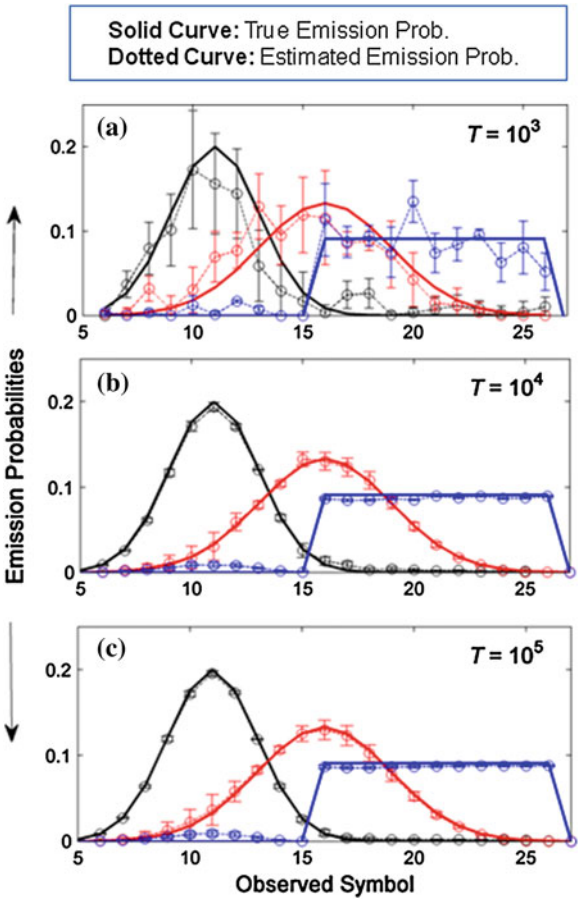

Fig. 5 Comparison of the true and the estimated emission probabilities (from PMF–HMM algorithm) at different sequence lengths (T). For short sequence (**a**) the estimates were poor and showed high variance. For longer sequences (**b** and **c**) the estimated parameters matched the true parameters quite well with high confidence

log–log plot. However, the time spent in matrix factorization remained almost constant because of its insensitivity to the sequence length (complexity is $O(M^2N)$). Hence, at smaller sequence length, matrix factorization dominated the total run time but its contribution quickly faded away as the sequences grew longer. Figure 5 plots the estimated emission probabilities of the three hidden states along with the true emission probabilities as given in Eq. (10). The error bars represent the 95% confidence interval of the estimated value as a result of 20 runs of each experiment. Clearly, as the sequence length was increased, the estimated emission probabilities converged to the true values and the error bars shrank.

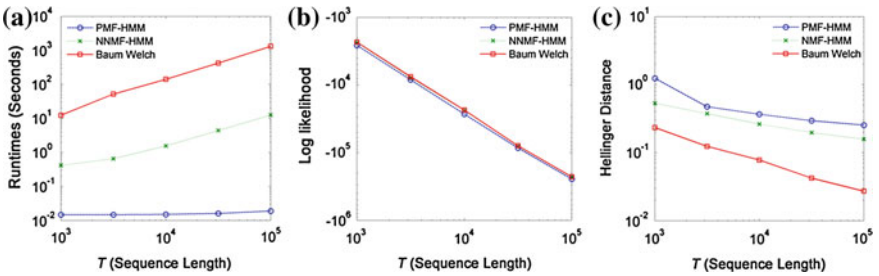

Fig. 6 Comparison of different characteristics of PMF–HMM, NNMF–HMM and Baum–Welch algorithms on the synthetic data at different sequence lengths. Algorithms runtimes, likelihood values of the sequence, post training, $P(O|\lambda)$ and *Hellinger* distances are compared (**a**, **b** and **c**)

In Ref. [9], the author compares different characteristics of NNMF–HMM algorithm with that of Baum–Welch algorithm. We add the characteristics of PMF–HMM algorithm, to these published results, so as to have a common ground to compare the three algorithms. In Fig. 6b, the likelihood values of observing the symbol sequence given the estimated HMM parameters, $P(O|\lambda)$, is plotted versus the sequence length. It is remarkable that despite having a different generative model, both the PMF–HMM and NNMF–HMM algorithms resulted in likelihood values that were at par with that of Baum–Welch algorithm. In Fig. 6c, the *Hellinger* distance between the estimated and true emission probabilities is plotted versus the sequence length for the three algorithms. As the sequences grew longer, the estimated emission probabilities converged to the true values, which is indicated by the drop in the distance values. Overall, the Hellinger distance of PMF–HMM algorithm was higher than the other two algorithms, which can also explain its marginally lower likelihood values plotted in Fig. 6(b). However, the main difference was observed in the run times of the three algorithms, where the PMF–HMM algorithm was better than the other two by a significant margin (Fig. 6a).

In Sect. 3.4, we discussed the ability of the PMF–HMM algorithm to handle non-degenerate observations. Here, we demonstrate the advantage of this ability for estimating the HMM parameter. In the previous experiment, the continuous observation values were rounded to the nearest integer to yield a discrete symbol sequence. This discretization came with a cost of some information loss. As an alternative, a soft discretization scheme can be employed, which assigns a real valued observation to multiple symbols with different memberships. One such soft discretization scheme is shown in Fig. 7, which involves defining a Gaussian kernel centered at the observation (8.35 in this case). As every symbol is bounded on both sides, the degree of membership of the real observation to a symbol can be obtained by computing the area under the Gaussian kernel between the symbol's boundaries (refer Fig. 7). Because of the use of a probability density function (the Gaussian kernel), the membership values have the desired addition to unity property. We used this soft discretization scheme to obtain non-degenerate observation vectors and computed the count matrix using Eq. (8). The standard deviation of the Gaussian kernel was

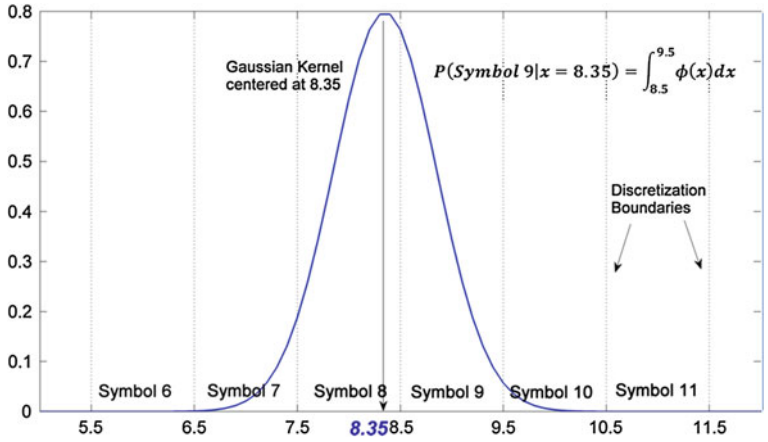

Fig. 7 Illustration of a scheme for generating non-degenerate observation vector from a continuous value. Instead of assigning the observation a specific symbol value, its membership to different symbol can be obtained by computing the area under a Gaussian Kernel centered at that value

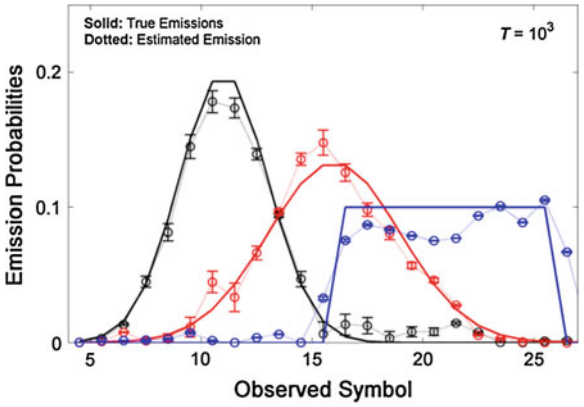

Fig. 8 Comparison of the true and estimated emission probabilities (by PMF–HMM algorithm) at the sequence length, $T = 1000$. The count matrix was obtained using the non-degenerate observations. The quality of estimated parameters is much better in comparison to case when discrete observations were used to obtain the count matrix (Fig. 5a)

fixed at 1.0 (equal to the interval width). The remaining steps for computing the HMM parameter were identical to the case of discrete symbol sequence. Figure 8 shows the estimated emission probabilities, at the sequence length of 1000, along with the true emission probabilities. This figure can be compared with the Fig. 5a, where the hard discretization scheme was employed for obtaining the count matrix. The estimated emission probabilities, resulting from soft discretization, were not just closer to the true ones but also had tighter confidence interval.

Table 1 Comparison of the time spent in building HMM classifiers by the Baum-Welch and the proposed algorithm on key stroke dynamics data. Cohen's kappa values on the test dataset are also listed for the two classifiers

	Training time	Cohen's kappa (κ)
PMF–HMM	0.47 s	0.32
PMF–HMM (non-degenerate)	1.98 s	0.35
Baum–Welch	4.3 h	0.38

4.2 Key Stroke Dynamics Data

This dataset was generated, in a study at CMU, for the analysis of typing rhythms to discriminate among users [8]. The purpose was to model the typing rhythms for separating imposters from actual users. In the study, 51 subjects were asked to type the same password 400 times over a period of few weeks. The password had eleven characters (including the enter key) and was identical for all the subjects. The recorded data consisted of the hold time (the length of time a key was pressed) and transition time (the time taken in moving from one key to the next). Therefore for every typed password, the data had 21 time entries (11 keys and 10 transitions). We used this dataset to perform a HMM based classification of a typed password into the most probable subject. The idea is to first learn a HMM for each subject from the passwords in the training set. Thereafter, classify the passwords in the test set using the *Maximum A-Posteriori* (MAP) criterion. The training and test sets were obtained after splitting the original dataset in half. As the first step, we discretized the continuous time values into 32 equi-spaced bins. Therefore, the subsequent HMMs were comprised 32 observed symbols and 21 hidden states.

Table 1 compares the Baum–Welch and the PMF–HMM algorithm for their runtime and classification accuracy. We used both degenerate and non-degenerate variants of the PMF–HMM algorithm. The classification accuracy is quantified using *Cohen's kappa*(κ) statistics, which is a measure of inter-rater agreement for categorical items [13]. The kappa statistics takes into account the agreements occurring by chance and hence usually gives a conservative estimate of a classifier's performance. It turns out that the Baum–Welch algorithm took almost 10000X more time than the proposed algorithm for the same job. Moreover, the longer time taken by the Baum–Welch algorithm was not reflected on its classification performance on the test dataset. The kappa value, $\kappa = 0.38$, of the classifier trained by the Baum–Welch algorithm was only slightly better than that of the PMF–HMM algorithm ($\kappa = 0.32$). Moreover, the PMF–HMM's classification performance was further improved with the use of non-degenerate observations ($\kappa = 0.35$) without a impacting the training time significantly.

5 Conclusion

In this chapter we proposed a probabilistic matrix factorization based algorithm for the parameter estimation of a Hidden Markov Model. The 2D matrix, which is factorized, contains the information about the number of times different pairs of symbols occur in an observed sequence. A model is proposed that governs the generation process of these symbol pairs. Thereafter, an EM algorithm is used to estimate the HMM parameters assuming this generative model. The time required for the parameter estimation with the proposed algorithm can be orders of magnitude shorter than the Baum–Welch algorithm, thus making it attractive for time critical problems. We also discussed the ability of the proposed algorithm to handle non-degenerate observations and demonstrated the resulting improvement in the quality of HMM parameter estimates.

References

1. Baum, L., Petrie, T., Soules, G., Weiss, N.: A maximization technique occuring in statistical analysis of probabilistic function of Markov chains. Ann. Math. Stat. **41**, 164–171 (1970)
2. Eddy, S.: What is a hidden Markov model? Nat. Biotechnol. **22**, 1315–1316 (2004)
3. Fonzo, V., Pentini, F., Parisi, V.: Hidden Markov models in bioinformatics. Curr. Bioinform. **2**(1), 49–61 (2007). http://www.ingentaconnect.com/content/ben/cbio/2007/00000002/00000001/art00005
4. Hofmann, T.: Probabilistic latent semantic analysis. In: Proceedings of Uncertainty in Artificial Intelligence, UAI. Stockholm (1999). http://citeseer.csail.mit.edu/hofmann99probabilistic.html
5. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR' 99, ACM, New York, NY, USA, pp. 50–57 (1999). http://doi.acm.org/10.1145/312624.312649
6. Hu, J., Yu, X., Qiu, D., Chen, H.: A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection. Netw. Mag. Glob. Internetwkg. **23**, 42–47 (2009). http://dx.doi.org/10.1109/MNET.2009.4804323
7. Juang, B.: On the hidden Markov model and dynamic time wraping for speech recognition—a unified view. AT&T Tech. J. **63**, 1212–1243 (1984)
8. Killourhy, K., Maxion, R.: Comparing anomaly-detection algorithms for keystroke dynamics. In: 39th International Conference on Dependable Systems and Networks. Lisbon, Portugal (2009)
9. Lakshminarayanan, B., Raich, R.: Non-negative matrix factorization for parameter estimation in hidden Markov models. In: Proceedings of IEEE International Workshop on Machine Learning for Signal Processing. IEEE, Kittila, Finland (2010)
10. Lee, D., Seung, H.: Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755), 788–791 (1999). doi:10.1038/44565
11. Levinson, S., Rabiner, L., Sondhi, M.: An introduction to the application of probabilistic functions of Markov process to automatic speech recognition. Bell Syst. Tech. J. **62**, 1035–1074 (1983)
12. Rabiner, L.R.: Readings in speech recognition. Chap. A tutorial on hidden Markov models and selected applications in speech recognition., pp. 267–296. Morgan Kaufmann Publishers Inc., San Francisco (1990). http://dl.acm.org/citation.cfm?id=108235.108253
13. Uebersax, J.: Diversity of decision-making models and the measurement of interrater agreement. Psychol. Bull. **101**, 140–146 (1987)