Abstract—In real world applications where uncertainty is crucial, hidden Markov models (HMMs) have proven a useful and efficient tool for modeling such systems. HMMs have been used by many researchers for speech recognition and have achieved better performance than other modeling tools. In keystroke dynamics, keystroke events are generally non-deterministic. As HMMs have the ability to handle stochastic processes, they can be a suitable tool to recognize the keystroke pattern of a user. The implementation of keystroke dynamics based on statistical and neural network suffer from computational complexity and also sometimes required long time to identify user. Hidden Markov model can help to find solutions of these problems. Although HMM have been used in many biometric systems but only a very few researchers have used HMM in keystroke dynamics. This paper present a detail about Markov chain and hidden Markov model with examples and also summarizes and compares researches those have used HMM in keystroke dynamics.

Index Terms—Hidden Markov Model, Markov chain, keystroke dynamics, generative model, discriminative model.

I. INTRODUCTION

In Machine learning, many approaches exist for solving the classification problem. They can be broadly categorized as generative and discriminative classifiers. For inputs $x$ with class labels $y$, a generative classifier tries to learn a model of the joint probability, $p(x, y)$ and make predictions by using Bayes rules to calculate posterior $p(y|x)$ . The class label $y$ with the highest posterior probability is chosen [1]. The main difference between generative and discriminative models is that, a generative model is a full probabilistic model of all variable whereas a discriminative model is only for target variable(s) dependent on observed variable. For example, we want to determine the language spoken by someone who is talking. Here the generative approach will be to learn each and every language and determine as to which language the speech belongs to. Contrary, discriminative approach will be finding linguistic differences without learning any language. Popular generative models are Gaussians, Naïve Bayes, mixtures of multinominal’s, Mixture of Gaussians, Mixtures of experts, Hidden Markov Models, Sigmoidal belief networks, Bayesian networks, and Markov random fields. The most popular discriminative models are Logistic regression, Support Vector Machine, Neural Networks, nearest neighbor, and Conditional Random Fields.

Over the past few years, discriminative models became very popular in keystroke dynamics. There are several advantages of using a discriminative model over generative model in keystroke dynamics. For large samples, discriminative classifiers generally outperform generative classifiers. Generative classifiers learn about the posterior indirectly and may make the wrong assumptions of the data distribution. According to Vapnik [2], “one should solve the classification problem directly and never solve a more general problem as an intermediate step (such as modeling $P(Y|X)$)”. Discriminative models are very fast in making predictions for new data; means faster classification of new data compared to generative model [3]. The generative models have a higher asymptotic error than the discriminative model as the number of training samples become larger. Although discriminative classifiers are mostly preferred by the researchers, A. Y. Ng et al. [1] have worked with both generative classifier and discriminative classifier and their result shows that, as the number of training samples increased, there can be two distinct regimes of performance. For fewer training samples, the generative model performed better and as the training samples increased, the discriminative classifier overtook the performance of generative classifier. On the other hand, Ilkey Ulusoy et al. [3] have also shown that, although a discriminative model is capable of very fast inference and able to focus on highly informative features, a generative model gives high classification accuracy, and has the ability to localize the objects and augment the labeled data with unlabelled data. Motivate from these works, this study focuses on understanding a generative classifier, namely hidden Markov model (HMM), and surveying the research in keystroke dynamics that have used HMM as a classifier.

The paper organization is as follows: Section II describes details about Markov chain and hidden Markov model with several examples, Section III presents the elements of HMM, Section IV describes the problems with HMM and their solutions. Section V is the summary of various researches in keystroke dynamics using HMM and their comparison and finally Section VI the conclusion and suggestion for future work.

II. MARKOV MODEL

A Markov model is a stochastic model which is used to model a system that randomly changes, and where the future state, given the past and the present, only depends on the present. Different types of Markov models are used depending on the system behavior and environment. For example, if the system is autonomous and is fully observable, then a Markov chain is used. If the system is autonomous but only partially observable then a hidden Markov model is used. On the other hand, a Markov decision process is used when the system is controlled and fully observable. A Markov decision process is a Markov chain where state transitions depend on the current

Md Liakat Ali, John V. Monaco, and Charles C. Tappert
Seidenberg School of CSIS, Pace University, White Plains, New York

Proceedings of Student-Faculty Research Day, CSIS, Pace University, May 1st, 2015
state and an action vector applied to the system. A partially observable Markov decision process is a Markov decision process, which is used when the system is controlled but partially observable [4].

### A. Markov Chain

Markov chain is the simplest Markov model which models a system that randomly changes through time. Markov chains and Markov process are two important classes of stochastic processes. A Markov chain is a discrete-time process whereas a Markov process is the continuous-time version of Markov chain. To understand the Markov chain (discrete Markov process) and its terminology, we will discuss two well-known examples. The first is the “Random Walker” example from Olle Häggström, 2002 [5] and second is “The Weather” example from L. R. Rabiner, 1989 [6].

1) The Random Walker Markov Model

Assuming at time 0, a random walker stands in corner $s_1$ of four street Corners $s_1$, $s_2$, $s_3$ and $s_4$, as shown in Figure 1. The random walker can move to either $s_2$ or $s_4$ street-corner, as they are adjacent to street-corner $s_1$. To decide where to move, he flips a fair coin at time 1 and moves to either $s_2$ or $s_4$ according to whether coin comes up heads or tails. He flips the coin again at time 2 to decide which adjacent corner to move to with a decision rule such as move clockwise if it is head or vice-versa. By flipping the coin and applying the decision rules, the random walker decides the next move at times 3, 4, ..., $t$. For each $t$, let $q_t$ be the index of the street-corner where the walker stands at time $t$. As the walker starts from $s_1$ at time 0, we have

$$P(q_0 = 1) = 1$$

At time 1, he will move either $s_2$ or $s_4$ with the probability $\frac{1}{2}$ each, so we have

$$P(q_1 = 2) = \frac{1}{2}$$

And

$$P(q_1 = 4) = \frac{1}{2}$$

Now at time $t \geq 2$, let assume the walker stands at corner $s_2$. As the coin-flipping mechanism decides where to move next, we have the conditional probabilities

$$P(q_{t+1} = s_1 | q_t = s_2) = \frac{1}{2}$$

And

$$P(q_{t+1} = s_3 | q_t = s_2) = \frac{1}{2}$$

As the coin-flip at time $t + 1$ is independent of all previous coin-flips, so it is also independent of $q_0, ..., q_t$, and we will get the same conditional probabilities if we condition the process up to time $t$, i.e.,

$$P(q_{t+1} = s_1 | q_0 = i_0, q_1 = i_1, ..., q_{t-1} = i_{t-1}, q_t = s_2) = \frac{1}{2}$$

and

$$P(q_{t+1} = s_3 | q_0 = i_0, q_1 = i_1, ..., q_{t-1} = i_{t-1}, q_t = s_2) = \frac{1}{2}$$

for any choice of $i_0, ..., i_{t-1}$. That means the conditional distribution of time $q_{t+1}$ given time $(q_0, ..., q_t)$ depends only on time $q_t$ and not times $(q_0, ..., q_{t-1})$. This phenomenon is called the memoryless property also known as the Markov property. Moreover the conditional distribution of $q_{t+1}$ the is same for all $n$ because the mechanism that decide where to move next is same at all times. This is known as time homogeneity, or simply homogeneity.

From the above discussion, we write a general definition of discrete-time Markov chain as follows:

Let $A$ be a $(k \times k)$-matrix with elements $a_{ij}$; $i, j = 1, ..., k$. A random process $(q_0, q_1, q_2, ...,)$ with finite state space $S = \{s_1, s_2, ..., s_k\}$ is called Markov chain with transition matrix $A$, if for all $t$, all $i, j \in \{1, ..., k\}$ and all $i_0, ..., i_{t-1} \in \{1, ..., k\}$, we have

$$P(q_{t+1} = s_j | q_0 = i_0, q_1 = i_1, ..., q_{t-1} = i_{t-1}, q_t = s_i) = P(q_{t+1} = s_j | q_t = s_i) = a_{ij}$$

where $a_{ij}$ are the transitional probabilities, the conditional probability of being in state $s_j$ given the previous state was $s_i$. So, the above random walker is an example of Markov chain with state space $\{1, ..., 4\}$ and transition matrix,

$$A = \begin{bmatrix}
0 & 1 & 0 & 1 \\
1 & 2 & 0 & 1 \\
0 & 1 & 2 & 0 \\
1 & 2 & 0 & 1 \\
\end{bmatrix}$$

The transition matrix must satisfy $a_{ij} \geq 0$ for all $i, j \in \{1, ..., k\}$ and $\sum_{j=1}^{k} a_{ij} = 1$ for all $i, j \in \{1, ..., k\}$.

This implies that

$$P(q_{t+1} = s_1 | q_t = s_1) + P(q_{t+1} = s_2 | q_t = s_1) + \cdots + P(q_{t+1} = s_k | q_t = s_1) = 1$$

Figure 2 shows the transition graph for the random walker.
Markov chain which consists of nodes representing the states of the Markov chain; and arrows that represent the transition probabilities.

Another important characteristic of the Markov chain is initial distribution, which tells us how the Markov chain starts. The initial distribution is represented as a row vector $\pi^0 \in \mathbb{R}^k$ given by $\pi^0 = (\pi^0_1, \pi^0_2, \ldots, \pi^0_k) = (P(q_0 = s_1), P(q_0 = s_2), \ldots, P(q_0 = s_k))$, and $\sum_k \pi_i^0 = 1$.

So in our random walk example above, we have at time $t=0$, $\pi^0 = (1, 0, 0, 0)$: Similarly we can find the distribution of Markov chain at time $t$. $\pi^t = (\pi^t_1, \pi^t_2, \ldots, \pi^t_k) = (P(q_t = s_1), P(q_t = s_2), \ldots, P(q_t = s_k))$, and $\sum_k \pi_i^t = 1$. For example at time $t=1$, the distribution of Markov chain is $\pi^1 = (0, \frac{1}{2}, 0, \frac{1}{2})$.

2) The Three-state Markov Process of Weather

To understand Markov model better, we will go for another example - the three-state Markov process of the weather [7].

Assume that on any given day, the weather is observed as being one of the following states:

- State 1: rain
- State 2: cloudy
- State 3: sunny

Transition probabilities between states are described by the transition matrix $A$

$$A = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

Figure 3 shows the transition graph for the three-state Markov process of weather which consists of nodes representing the states of the Markov model. The arrows represent the transition probabilities.

For example on the day (1), the weather is sunny in state 3, and we want to find the probability that the weather for next 7 days will be sunny-rainy-rainy-cloudy-sunny. Which means, we want to find the probability of the observation sequence, $O = (s_{3}, s_{3}, s_{3}, s_{1}, s_{3}, s_{2}, s_{3})$

$$P(O|Model) = P(s_{3}|s_{3})P(s_{3}|s_{3})P(s_{1}|s_{3})P(s_{3}|s_{3})P(s_{1}|s_{3})P(s_{2}|s_{3})P(s_{3}|s_{3})$$

$$= \pi_3 \cdot a_{33} \cdot a_{31} \cdot a_{13} \cdot a_{32} \cdot a_{23} = 1 \times 0.8 \times 0.8 \times 0.1 \times 0.4 \times 0.3 \times 0.1 \times 0.2 = 1.536 \times 10^{-4}$$

We can also use this model to answer another interesting question: Given that the model is in known state, what is the probability that it will stay in that state for exactly $d$ days? That means, we have to find the probability of the observation sequence, $O = (q_1 = s_i, q_2 = s_i, \ldots, q_d = s_i, s_{d+1} = s_j$ and $s_j \neq s_i$) of the given model, which is

$$P(O|Model, q_1 = s_i) = (a_{ii})^{d-1}(1-a_{ii}) = p_i(d)$$

Where $p_i(d)$ is the probability density function of duration $d$ in state $i$. Based on this, we can calculate the expected number of observations in a particular state as

$$E[d|s_i] = \sum_{d=1}^{\infty} dp_i(d) = \sum_{d=1}^{\infty} d(a_{ii})^{d-1}(1-a_{ii}) = \frac{1}{1-a_{ii}}$$

![Fig. 3. A Markov model with 3 states and state transition probabilities.](attachment:image)

So in the above weather Markov model, the expected number of consecutive days of sunny weather is $(1/1-0.8) = 5$, for cloudy 2.5 and for rain, the expected number of consecutive days is 1.67.

B. Hidden Markov Model

So far we have discussed about Markov chain model which assume that each state uniquely corresponded to an observable event and the state of the system can be trivially retrieved when an observation sequence is made. The HMM is an extension of Markov chain where observation of the system is a probabilistic function of the state. So HMM has an underlying stochastic process that is not observable (hidden), but can only be observed through another set of stochastic processes that produce a sequence of observation. [8].

To differentiate HMM from Markov chain, we will consider the following modification of weather example [8].

In previous weather model we have seen that, the weather was directly observable. Now assume that we were kidnapped and kept in a locked room for several days. We cannot observe the weather directly—the only evidence we have is, a kidnapper who brings food has also brought an umbrella. Let us assume that the probability of bringing an umbrella is 0.8, 0.3, 0.1 given the weather is rainy, cloudy and sunny, respectively. Before we were kidnapped, we could observe the weather and the weather Markov process was:

$$P(q_1, q_2, q_3, \ldots, q_n) = \prod_{i=1}^{n} P(s_i|s_{i-1})$$

But now as the actual weather is hidden and the probability of states of weather corresponds to seeing umbrella ($u = True$ or False), we can use Bayes’ Rule:

$$posterior = \frac{likelihood \times prior}{evidence}$$

$$P(q_1, \ldots, q_n|u_1, \ldots, u_n) = \frac{P(u_1, \ldots, u_n|q_1, \ldots, q_n)P(q_1, q_n)}{P(u_1, \ldots, u_n)}$$

For example the day we were kidnapped, it was sunny. The next day when the kidnapper brought food, he also carried umbrella into the room. Now assuming the prior probability of carrying an umbrella on any day is 0.5, we can find the probability of rain on second day by the following steps:

We know the conditional independence of two random variables $A$ and $B$ given $C$,

$$P(A,B|C) = P(A|C) \ast P(B|C)$$

Equivalently, if $A$ and $B$ is conditionally independent given $C$ (known) and in the case $B$ doesn’t tell anything about $A$:

$$P(A|C) = P(A|B,C)$$

So we have

$$P(A,B|C) = P(A|B,C) \ast P(B|C)$$

B2.3
Now back to the weather problem,

\[ P(q_2 = s_1 | q_1 = s_3, u_2 = True) = \frac{P(q_2 = s_1, q_1 = s_3 | u_2 = T)}{P(q_1 = s_3 | u_2 = T)} \]

\( u_2 \) and \( q_1 \) independent

\[ (Bayes \ Rule) \]

\[ P(u_2 = T | q_1 = s_3, q_2 = s_1)P(q_1 = s_3, q_2 = s_1) \]

\( Markov \ assumption \)

\[ P(u_2 = T | q_1 = s_3, q_2 = s_1) = \frac{P(q_1 = s_3)P(u_2 = T)}{P(q_1 = s_3)} \]

\[ (P(A, B) = P(A | B)P(B)) \]

\[ = \frac{P(u_2 = T | q_2 = s_1)P(q_2 = s_1 | q_1 = s_3)P(q_1 = s_3)}{P(u_2 = T | q_2 = s_1)P(q_2 = s_1 | q_1 = s_3)} \]

\[ = \frac{P(q_1 = s_3)P(u_2 = T)}{P(u_2 = T)} \]

\[ = 0.8 \times 0.1 = 0.16 \]

The above example depicts how to construct a weather model via HMM. In general there are other difficulties in the modeling procedure such as finding the number of states (model size) of the model, how to choose model parameters (such as transition probabilities) and the size of the observation sequence. In the next section we will discuss the elements of HMM, three basic problems for HMMs and solutions of those problems [7].

### III. ELEMENTS OF AN HMM

An HMM can be characterized by the following set of parameters:

1. \( N \), the numbers of states in the model \( S = \{S_1, S_2, ..., S_N\} \) and the state at time \( t \) as \( q_t \)
2. \( M \), the number of discrete observation symbols per state \( V = \{V_1, V_2, ..., V_M\} \)
3. The state transition probability distribution \( A = \{a_{ij}\} \) where \( a_{ij} = P(q_{t+1} = S_j | q_t = S_i) \)
4. The emission or observation symbol probability distribution in state \( j, B = \{b_j(k)\} \) where \( b_j(k) = P(o_t = v_k | q_t = S_j) \)
5. The initial state distribution \( \pi = \{\pi_i\} \) where \( \pi_i = P(q_1 = S_i) \)

Therefore, an HMM is specified by two scalars, \( N \) and \( M \), and three probability distribution \( \lambda = (A, B, \pi) \). Given a completely specified HMM, the observation sequence \( O = \{O_1, O_2, ..., O_T\} \) can be generated by the following steps:

1. Choose an initial state \( q_1 = S_1 \) according to initial state distribution \( \pi \)
2. Set \( t = 1 \)
3. Generate observation \( o_t = v_k \) according to the emission or symbol probability distribution in state \( S_i \), i.e., \( b_j(k) \)
4. Move to a new state \( q_{t+1} = S_j \) according to the state transition probability distribution for state \( S_i \), i.e., \( a_{ij} \)
5. Set \( t = t + 1 \) and return to step 3 until \( t \geq T \)

### IV. THE THREE BASIC HMM PROBLEMS AND SOLUTIONS

Given an HMM model, there are three basic problems that must be solved for the model to be useful in the real-world applications.

#### A. The three basic problems for HMMs

The three basic problems for HMMs are the following:

- **Problem 1 (Evaluation problem):** Given observation sequence \( O = \{O_1, O_2, ..., O_T\} \), and a model \( \lambda = (A, B, \pi) \), how do we efficiently compute \( P(O | \lambda) \), the likelihood of the observation sequence given the model? The solution is given by the Forward and Backward procedures.

- **Problem 2 (Decoding Problem):** Given observation sequence \( O = \{O_1, O_2, ..., O_T\} \), and a model \( \lambda = (A, B, \pi) \), how do we choose a corresponding state sequence \( Q = \{q_1, q_2, ..., q_T\} \) that is optimal such as best explains the data? The solution for this problem is provided by the Viterbi algorithm.

- **Problem 3 (Learning problem):** How do we adjust the model parameters \( A, B, \pi \) to maximize the likelihood \( P(O | \lambda) \). The solution is given by the Baum-Welch re-estimation procedure.

#### B. Solution to problem 1

Our goal is to compute the likelihood \( P(O | \lambda) \) of the observation sequence \( O = \{O_1, O_2, ..., O_T\} \), given the model \( \lambda \).

Naïve solution: the most straightforward way to find the likelihood is through enumerating every possible state sequence of length \( T \). For example let us consider such a fixed state sequence \( Q = \{q_1, q_2, ..., q_T\} \). The probability of the observation sequence \( O \) for the state sequence \( Q \) is

\[ P(O | Q, \lambda) = \prod_{t=1}^{T} P(O_t | q_t, \lambda) = \prod_{t=1}^{T} b_{q_t}(O_t) \]

The probability of the state sequence \( Q \) is

\[ P(Q | \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} ... a_{q_{T-1} q_T} \]

The joint probability of \( O \) and \( Q \), which means the probability that \( O \) and \( Q \) occur simultaneously, is

\[ P(O, Q | \lambda) = P(O | Q, \lambda)P(Q | \lambda) \]

The Probability of \( P(O | \lambda) \) is obtained by summing the joint probability over all possible state sequences \( q \) giving

\[ P(O | \lambda) = \sum_{q_1, q_2, ..., q_T} P(O | Q, \lambda)P(Q | \lambda) \]

\[ = \sum_{q_1, q_2, ..., q_T} \pi_{q_1} a_{q_1 q_2} b_{q_2}(O_2) ... a_{q_{T-1} q_T} b_{q_T}(O_T) \]

This approach is computationally unfeasible because, the sum is over all state paths. Even for small values of \( N \) and \( T \) and the complexity is \( O(N^TK) \). However, there exists a computationally simpler algorithm called Forward-Backward procedure to computes \( P(O | \lambda) \). We only need the forward part of the procedure to solve problem 1.
Forward procedure: consider the forward variable $\alpha_t(i)$ defined as
$$\alpha_t(i) = P(O_1, O_2, ... , O_t \mid q_t = S_t)$$
Computation of this variable can be efficiently performed by inductively as follows:
1. Initialization: $\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N$
   In this step, initializes the forward probabilities as the joint probability of the state $S_t$ and the initial observation $O_t$.
2. Induction: $\alpha_{t+1}(i) = \sum_{j=1}^{N} \alpha_t(j) a_{ij} b_j(O_{t+1})$ \quad $1 \leq t \leq T - 1$ and $1 \leq j \leq N$
   Here $\alpha_t(i)$ is the probability of the joint events that $O_1, O_2, ... O_t$ are observed. The product $\alpha_t(i) a_{ij}$ is the probability of joint events that $O_1, O_2, ... O_t$ are observed and state $S_j$ is reached at time $t+1$ via state $S_i$ at time $t$. Summing this product over all the $N$ possible states gives us the probability of $S_j$ at time $t+1$ with all the accompanying previous partial observations. Finally, $\alpha_{t+1}(j)$ is obtained by multiplying the summed quantity by the probability $b_j(O_{t+1})$ for all the states $j$.
3. Termination: $P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$. This step gives the desired calculation by sum of the terminal forward variables $\alpha_T(i)$.

Backward procedure: consider the backward variable $\beta_t(i)$ defined as
$$\beta_t(i) = P(O_{t+1}, O_{t+2}, ... , O_T \mid q_T = S_t)$$

Here $\beta_t(i)$ represents the probability of the partial observation sequence from $t+1$ to the end, given state $S_j$ at time $t$ and the model $\lambda$. Computation of this variable can also be computed through induction as follows:
1. Initialization: $\beta_T(i) = 1$ \quad $1 \leq i \leq N$
2. Induction: $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$ where $t = T-1, T-2, ... 1$ and $1 \leq i \leq N$
3. Terminate: $P(O \mid \lambda) = \sum_{i=1}^{N} \beta_T(i)$ \quad $t = T-1, ... 1$

C. Solution to problem 2
Our goal is to find state sequence $Q = \{q_1, q_2, ... q_T\}$ that is optimal. We have to define an optimality measure since several criteria are possible. The most widely used criterion is to find the single best state sequence path, i.e., maximize $P(Q \mid O, \lambda)$ which is equivalent to maximizing the posterior $P(Q \mid O, \lambda)$. The well known Viterbi algorithm is a formal technique for finding this single best state sequence which is based on dynamic programming methods.

Viterbi Algorithm: To find the single best state sequence, we define the quantity
$$\delta_t(i) = \sum_{S_{t-1}} \max \{ \delta_{t-1}(i) a_{ij} \mid b_j(O_t) \}\quad 1 \leq t \leq T$$

Where $\psi_{t+1}(j)$ the state at time $t$, a transition to state is $S_j$ maximizes the probability $\delta_{t+1}(j)$. The complete procedure for finding the best state sequence by Viterbi algorithm as follows:
1. Initialization: $\delta_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N$
   $\psi_1(i) = 0$ (no previous states)
2. Recursion: $\delta_t(j) = \max_{1 \leq s \leq N} [\delta_{t-1}(s) a_{sj} b_j(O_t)]$ \quad $2 \leq t \leq T; 1 \leq j \leq N$
   $\psi_t(i) = \arg \max_{1 \leq s \leq N} [\delta_{t-1}(s) a_{sj} b_j(O_t)]$
3. Termination: $P^* = \max_{1 \leq s \leq N} [\delta_T(s)]$
   $q^*_T = \arg \max_{1 \leq s \leq N} [\delta_T(s)]$
4. Path Backtracking: $q^*_t = \psi_{t+1}(q^*_{t+1}), t = T-1, T-2, ... 1$

Viterbi algorithm is similar to Forward procedure except that it uses maximization over previous states instead of summation.

D. Solution to problem 3
Problem 3 was the learning problem, that means training the HMM to encode observation sequence such that HMM should identify a similar observation sequence in future. To find $\lambda = (A, B, \pi)$ that maximize $P(O \mid \lambda)$.

General algorithm:
1. Initialize $\lambda_0$
2. Compute new model $\lambda$, using $\lambda_0$ and observe sequence $O$
3. Update $\lambda_0 = \lambda$
4. Repeat steps 2 and 3 until $\log P(O \mid \lambda) - \log P(O \mid \lambda_0) < \epsilon$

Baum-Welch algorithm:
Step 1:
- Let $\xi(i, j)$ be a probability of being in state $i$ at time $t$ and at state $j$ at time $t+1$, given model $\lambda$ and observation sequence $O$ $\xi(i, j) = \sum_{\bar{O}_{t+1}} \gamma_t(i) \pi_{i} a_{ij} b_{j} \beta_{t+1}(j)$ $P(O \mid \lambda)$

Where $\gamma_t(i)$ is a probability of being in state $i$ at time $t$ given observation sequence $O$. $\gamma_t(i) = \sum_{j=1}^{N} \xi(i, j)$. $\sum_{t=1}^{T-1} \gamma_t(i)$ is the expected number of transitions from state $i$ and $\sum_{t=1}^{T-1} \xi_t(i)$ is the number of transition from $i$ to $j$.

Step 2:
- $\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$ ratio of expected number of transitions from state $S_t$ to $S_j$, over expected number of transitions from state $S_t$
- $\bar{b}_{ij} = \frac{\sum_{t=1}^{T} \gamma_t(i) \pi_{i} a_{ij} b_{j} \beta_{t+1}(j)}{\sum_{t=1}^{T} \gamma_t(i)}$ ratio of expected number of times in state $j$ and observing symbol $v_k$ to expected number of times in state $j$
Baum-Welch algorithm uses the forward and backward algorithm to calculate the variable $\alpha, \beta$

V. RESEARCHES IN KEystroke DYNAMICS USING HMM

There are several advantages of keystroke dynamic systems. One most prominent advantage is uniqueness. Researches have shown that users can be identified or authenticated by their typing pattern. Lower implementation and deployment cost compared to other biometric authentication system have motivated many researchers and organizations interested in the keystroke dynamics. A keystroke dynamics system can be fully implemented by software and has no dependency on specialized hardware. Some other advantages of keystroke dynamics system are: transparency, noninvasiveness, and ability to continuously authenticate [9]. The main disadvantages of keystroke biometric authentication system are lower accuracy and performance compared to other biometrics authentication systems. There are various factors that cause the lower accuracy rate. However, the main factor may simply be the variation in the typing rhythm of users due to age, injury, and some other environmental factors.

Many researchers have proposed keystroke dynamics systems based on neural-networks and statistical models but their implementation is complex in computation and may take an unacceptable long time to identify a legitimate user or require excessive reconfiguration to add or remove a user to a data base of authorized users. Sometimes they also demand unrealistically consistent typing skill to distinguish between users, which limit the acceptance of those models. A keystroke dynamics-based authentication system using HMM may improve these limitations [10]. Table I shows the comparison of various researches in keystroke dynamics using HMM. All studies were conducted on standard QWERTY mechanical keyboard. The feature hold time is also called dwell time and press, release, and delay time of each key are called flight time features.

Although HMM have been widely used in other biometric systems from mid-twentieth century, Wendy and Chang [11] used HMM for the first time in keystroke dynamics in 2004. The reason behind using HMM was that it has probabilistic state transitions which can not only recognize the pattern but also predict the future sequence. The timing information of four events such as press, release, hold, and delay time of each key when a word is typed is collected. Participants had freedom to type user name as input. A low level keystroke collection program directly reads a high-resolution system counter within the operating system, and then converts it into milliseconds. Twenty participants entered their names repeatedly with small breaks allowed and twenty samples were collected for establishing each user profile. To test the data, each participant entered all twenty participants’ names ten times each. The authors used a discrete first-order HMM to train the model due to non-deterministic nature of the keystroke events. As there were 10 test inputs from each of twenty participants, there were two hundred impostor samples to test each individual profile. Although the actual accuracy are not clear, the authors claimed that their experimental results were nearly perfect. The authors have given a few recommendations to improve the process when establishing user profile (model for particular user) such as obtaining typing patterns from commonly and frequently used word, string of various lengths and key compositions such as pair of keys, etc.

Chang [12] extended their previous research on keystroke dynamics and proposed a method to improve the process with a histogram of the similarity measured as part of the HMM approach to find a suitable threshold. Again the result are not clear, author claimed that a 30% improvement from the previous study by using similarity histogram.

Rodrigues et al. [13] presented a new approach in keystroke dynamics through numerical keyboard. Five features such as ASCII key code and four keystroke latencies were extracted from the user input. The input is eight numerical characters collected from numerical keyboard such as cell phones, ATM machine, etc. Training samples were collected from twenty participants. Ten samples from each participant in each session of total 4 sessions, totaling in 40 samples per user were collected. For testing purpose, 30 samples were collected from ones other than the true user resulting 600 faked samples.

<table>
<thead>
<tr>
<th>Study</th>
<th>Participants</th>
<th>Samples Per User</th>
<th>Features</th>
<th>Input</th>
<th>Method</th>
<th>EER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Training</td>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rodrigues et al., 2005 [13]</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>DT, FT, ASCII key code</td>
<td>Continuous HMM</td>
<td>3.6</td>
</tr>
<tr>
<td>Vuyyuru et al., 2006 [14]</td>
<td>43</td>
<td>9</td>
<td>20</td>
<td>DT</td>
<td>Fixed-text:“master of science in computer science”</td>
<td>HMM</td>
</tr>
<tr>
<td>Jiang et al., 2007 [15]</td>
<td>315</td>
<td>15</td>
<td>13</td>
<td>DT, FT, n-graph</td>
<td>User-fixed text Minimum 9 characters</td>
<td>HMM, Gaussian</td>
</tr>
<tr>
<td>Zhang et al., 2010 [16]</td>
<td>12</td>
<td>20</td>
<td>40</td>
<td>DT, FT</td>
<td>User-fixed text Minimum 10 characters</td>
<td>HMM</td>
</tr>
</tbody>
</table>
The experiment was conducted by using statistical classifier and HMM. The basic idea of HMM model was associated with each latency measure to a state of the model and for each password; likelihood probability of the corresponding HMM model was estimated through the Viterbi algorithm. If the likelihood probability estimate is superior to a threshold value, the user is authentic and impostor otherwise. To train the system, Baum-Welch algorithm implemented in Hidden Markov Toolkit was used. For statistical classifier design and system evaluations, three sets of training samples were used for model dimensioning. The performance of the HMM classifier outperforms all the 3 statistical classifiers in experiments because the method of updating the model in HMM allowed a more efficient form for modeling keystroke dynamics. The best result achieved by the HMM is ERR 3.6%.

Vuyyuru et al. [14] have proposed a method using modified HMM which can dynamically add or remove users without retraining the whole system and can adapt to the changing typing pattern of the users. There were 43 users participated in the data collection. For training samples, each user types the string “master of science in computer science” for nine times. And for testing samples, number of sample varied from 0 to 102 for each participant with total 873 samples which implies on average 20 testing samples per user. Four features such as key hold time, press latency, release latency and key interval time were extracted from each user samples. But only hold time feature was used as feature vector because the author’s literature study found that hold times are more effective than other features for user authentication. Figure 4 shows the state transition of a user in the HMM used in the proposed model for the string “mas” with two states for each character where \{m_1, m_2\}, \{a_1, a_2\}, and \{s_1, s_2\} the sub-states for the characters m, a, and s. At time \(t=1\), the state transitions are only from the sub-states of ‘m’, either \(m_1\) or \(m_2\), to the sub-states ‘a’, either \(a_1\) or \(a_2\). A is the state transition matrix and \(\pi\) is the initial state distribution. The author assumed that the probability of the key hold times follow the Gaussian distribution. As a result the probability of observing a symbol (key hold time) \(o_t\) in state \(j\) with mean \(\mu_j\) and standard deviation \(\sigma_j\) is \(p(o_t|\mu_j, \sigma_j)\).

For each user, a distinct HMM was developed by using modified Rabiner’s re-estimation formulae of multiple observation sequences on the reference keystroke patterns. In order to reduce the total number of computations, the structure of the HMM parameters was modified and then used in the modified forward and backward procedure. Figure 5 shows the block diagram of user authentication in the authentication phase of the proposed model. Their experiment have found best false accept rate 0.74% and false reject rate 8.06% and ERR 3.04%.

Jiang et al. [15] have proposed a statistical model for web authentication with fixed-text keystroke analysis using HMM and Gaussian modeling. They claimed that the proposed system can enhance the security in authentication mechanism. Beside key press and key release event, the authors have used n-graph (Three or more consecutive keystroke events) as features. Reference profiles were created first for each user from training samples. To create training samples, 58 participants supplied 20 samples of two familiar strings (user name and password) minimum 9 characters with 15 attempts by each participant totaling 870 training samples. To generate test sample, another 257 participants attacked on each profile ranging 44 to 82 times for a total 3528 imposter test samples.

HMM was used to model the timing information of keystroke sequence as shown in the figure 6. Un-shaded circles \(q_i\) are unknown state variables and shaded circles \(y_i\) are observed state variables at time \(t\). \(A\) is the state transition probability matrix from state \(q_i\) to state \(q_j\). \(\eta\) is the state emission probability matrix \(P(y_t|q_i)\) of \(t\)-th state and \(\pi\) is the initial state probability of \(i\)-th state. For a given sequence of consecutive keystrokes, \(S=[s_1, s_2, ..., s_m]\), the number of n-graphs is \(m-m+1\), the set of n-graphs denoted as \(G=\{g_1, g_2, ..., g_{m-m+1}\}\).
and the set of durations of \( n \)-graphs is denoted as \( GD \), where 
\[
GD = \{d(g_1), d(g_2), \ldots, d(g_{m-n+1})\}
\]
where \( g'_i \) is the number of appearances of \( n+1 \)-graph in \( G \).
The state transition matrix \( A \) was calculated by the following formula:
\[
A_{gig_{i+1}} = \frac{\theta_{g'_i}}{m-n}
\]
where \( \theta_{g'_i} \) is the mean value of the duration \( d(g) \) for \( n \)-graph \( g \), and \( \sigma_g \) is the standard deviation. The initial probability \( \pi \) is the probability that the \( n \)-graph appeared in \( S \). Forward algorithm has been used to calculate the probability of keystroke sequence \( S \) for each HMM and to choose one which has the highest probability and achieved EER of 2.54%.

Zhang et al. [16] have proposed a model that uses HMM for keystroke sequence analysis and time series to compute the state output probability of HMM and modified forward algorithm to compute the user typing behavior state in authentication phase. The proposed model preprocesses user’s keystroke sequence to get the sets of digraph and keystroke duration time. It can compute initial state probability, transition probability vectors and state output probability of HMM by time series model and normal distribution model.

Author’s claim that, previous models were unable to predict user’s behavior once the user’s behavior change but the proposed model is efficient when the user’s typing behavior is steady or changes with some regularities. To generate training samples, 12 users provided two familiar strings as the user name and password (length of password is more than 10) for 20 times within 3 months. To create test sample, each user typed his/her username and password 20 times or more which were used to calculate FRR, and 20 times or more other user’s name and password to calculate FAR. The experiment found that if time series is used, whether the user’s keystroke behavior is stable or not, the EER is less than 2%.

VI. CONCLUSION

Hidden Markov models can learn from temporal data and can approximate the recurrent sequences which make it an excellent tool for modeling those systems where uncertainty is crucial. HMMs have proved effective and efficient tools in speech recognition. But their accuracy in keystroke dynamics however is still lower than some other methods. This is because of few training samples were used to train the models. HMMs are probabilistic models and therefore require substantial training data. In speech recognition, plenty of training samples were used to train the model and hence achieved better accuracy. Moreover, very limited researches have been done in keystroke dynamics using HMM and also there are several limitations on those researches. There are several practical aspects need to be considered in data collection, features selection and choosing acceptance thresholds. None of the existing researches can able to perform continuous monitoring of the system. Some of the researchers were inefficient if the user typing patterns changes irregularly. It would be interesting future work to see how HMMs perform with robust training set with number of different data collection session. Larger input string may also increase performance of HMM because it creates more representative latencies. Some other future scope for keystroke dynamics investigations using HMM are but not limited to- online continuous real-time identity verification and user authentication using touch screen-based features, etc.

REFERENCES


