Hidden Markov Models (HMMs) and Security Applications

Rubayyi Alghamdi
Concordia University
Al-Baha University
Information Systems Security
CIISE, Concordia University
Montreal, Quebec, Canada

***Abstract*—The Hidden Markov models (HMMs) are statistical models used in various communities and applications. Such applications include speech recognition, mental task classification, biological analysis, and anomaly detection. In hidden Markov models, there are two states: one is a hidden state and the other is an observation state. The purpose of this survey paper is to further the understanding of hidden Markov models, as well as the solutions to the three central problems: evaluation problem, decoding problem and learning problem. In addition, applying HMMs in real world applications such as security and engineering will improve the classification and accuracy for the whole field.**

***Keywords*—Markov model; Hidden Markov model; HMM, Markove model; Forward algorithm; Viterbi Algorithm; Baum-Welch algorithm**

I. Introduction

The Hidden Markov models (HMMs) were originally introduced in the statistics literature in 1957. They remain one of the most popular models used for evaluating sequential and temporal data due to their efficiency in estimating parameter and doing inferences. Moreover, HMMs are also rich enough to handle real world application. In an engineering field, such as speech processing, source coding and in a security field such as credit card fraud and cloud computing, HMM can be particularly useful [1].

Section two of this survey paper will explain the Markov model and how it relates to the hidden Markov model, but in particular how it generates patterns and hidden patterns. Moreover, the limitations of the Markov process are explained in a weather example. Section three will then address the most important algorithms that HMM uses which are: the Forward Algorithm, Viterbi Algorithm, and Forward-Backward Algorithm. In section four, the paper will show how HMMs apply in real world applications with some solid examples, followed by the last section for conclusion.

II. Background

A. Markov Model

Markov model is mathematical model that make it possible to study complex systems. It is canonical and probabilistic, but specifically for temporal and sequential data. The model are named for Andrey Markov, a Russian mathematician who did some work on stochastic processes in the early nineteen century. The basic idea behind a Markov model is establishing a state of the system and then moving to a new state depending only on the values of the current state not on the previous history of the system. In other words, the future is depending only on the present [2]. The Markov model can be used to model an extraordinarily large number of applications such as weather, economic data, music and more.

Al- Baha University

B. Markov Model definition

Random valuables $\{X_n\}$ (X1,X2,X3,……,Xn) are considered a Markov chain if their joint distribution respects the following three conditions:

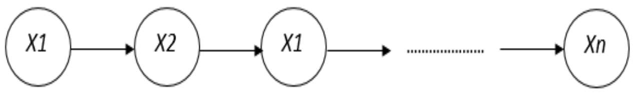

Fig. 1. Chain of observation state

1) *The state-space of this process is independent (Discrete space).*
2) *The time of this process is independent (Discrete time)*
3) *This process meets the Markov property:*

$$P(X_{n+1}=j \mid X_n=i,\ X_{n-1}=i_{n-1},\dots,X_1=i_1) = P(X_{n+1}=j \mid X_n=i) \tag{1}$$

$$P(X_1 X_2,\dots,X_n)=P(X_1)\prod_{n=2}^{N}P(X_n \mid X_{n-1}) \tag{2}$$

Thus, the chain $\{X_n\}$ is a Markov process because the value of the random variable X_{n+1} relies solely on the value X_n and it is not affected by $X_1, X_2,\dots,X_{n-1}$ variable values. Moreover, the parameter space (time) must be discrete and the state-space shall be discrete (finite) or countable. Furthermore, these processes, in which the right hand side of above the equation is independent of time, thereby leads to the set of state transition probabilities aij :

$$a_{ij}=P(X_{n+1}=j \mid X_n=i),\ 1\le i,\ j\le N \tag{3}$$

With the state transition matrix coefficients having the properties:

$$a_{ij} \geq 0 \forall j,i \quad \text{And} \quad \sum_{j=1}^{N} aji = 1 \forall i$$

Where (j) is the current state and (i) is the previous state. So, all transitions probability aij is positive and each row must sum up to one, since each row represents the probability of jumping from or staying in the same state [2].

C. Weather example:

The figure below shows all possible first order transitions between the states of the weather example.

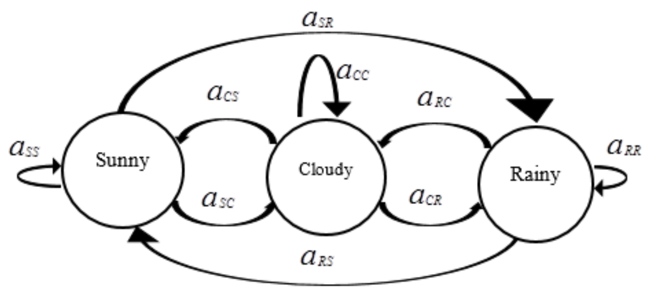

Fig. 2. 1st order transition Diagram

For a first order process with S states, there are S^2 transitions between the states as it is possible for any to follow another or return in the same state. Indeed, the probability of moving from one state to another is aptly called state transition probability. These S^2 probabilities may be collected together into a state transition matrix and do not vary in time, which is an important assumption. The state transitions matrix below shows possible transition probabilities for the weather example:

$$A = \{aij\} \quad i \quad \begin{array}{c|ccc} & \multicolumn{3}{c}{\overbrace{\hphantom{\text{Sunny Cloudy Rainy}}}^{j}} \\ & \text{Sunny} & \text{Cloudy} & \text{Rainy} \\ \hline \text{Sunny} & a11 & a12 & a13 \\ \text{Cloudy} & a21 & a22 & a23 \\ \text{Rainy} & a31 & a32 & a33 \end{array}$$

Limitation of a Markov process

In some cases, a Markov process may not be able to describe the events. Consider, in the weather example, that a person who has no ability to see the weather outside somehow acquires a piece of seaweed. This seaweed is affected by the weather, making it perhaps in turn soggy, damp, and dry. In this way, some information leads one to observe the situation of the current state of the weather, forming a link between the seaweed and weather. The pattern should therefore break up into two parts: the observable and the hidden. A more realistic problem presents itself in speech-to-text processing. This application needs words to interact with the systems and these words are effected by some factors such as the vocal chords, size of throat, position of tongue, and several other things. In both examples, the number of hidden states and the number of observed states can be different. This motivation leads to the hidden Markov model [3].

III. Extension to Hidden Markov Model

Hidden Markov Models model time series data. They are used in a huge number of applications such as speech recognition, pattern recognition and data accuracy. The key difference is that a hidden Markov model is a traditional Markov model that assumes the process is modeled with hidden states [4]. Regularly, the state would be visible to the observer making the state transition probabilities the only parameters. However, in a hidden Markov model the observer cannot see the state, but the output is visible depending on the state [5]. Hidden Markov models also have two types of states:

1) The first type of state is (wj). This set of states are the hidden states, and cannot be observed.

2) The second type of state is (vk). This set of states is actually visible, allowing each one to be associated with a state (wj). A hidden Markov model can thereby have a number of hidden or visible states.

3) An initial state π_i

$$P(v_1, v_N, ... w_1, w_n) = P(\pi_1) \prod_{i=2}^{N} P(w_n \mid w_{n-1}) \prod_{n=1}^{N} P(v_n \mid w_n) \tag{4}$$

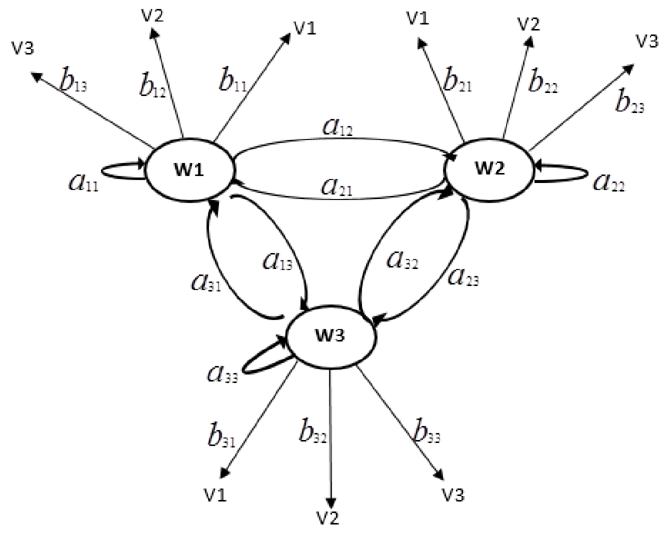

Fig. 3. HMM Transition Diagram

In figure (3), there are three states w={ $w1,w2,w3$} and the transition from state wi at (t-1) to wj at time (t) is given by transition probability matrix aij as in Markov process. The transition then takes the submission of aij because there are transitions between any hidden state wj . In addition, each hidden state can has one or more visible states (v) where bjk gives the probability of the emission matrix. i.e. the process then emits a symbol vk according to the output of the probability of bjk in the current state wj :

$$bjk = P(V_k \mid W_j)$$

The model can be mathematically defined as set: $\lambda = (\pi, w, v, A, B)$. Where λ denotes the HMM, and the vector $\pi = \{\pi_j\}$ is the initial state of probability distribution, A= aij transition probability matrix, and B= bjk emission probability matrix. By taking the submission of bjk because there are always transitions from any hidden state wj to visible states, vk and the submission must be equal to one:

$$\sum_{k=1}^{M} bjk = 1 \forall j$$

Given that, in such a hidden Markov model there are three important issues to be addressed.

A. The Three Basic Problems for HMMs

1) Evaluation problem

When the hidden Markov module is specified, it will contain: the number of hidden states, the number of visible states, and transition and emission probabilities. In such cases, many numbers of sequences of HMMs ($\lambda_1, \lambda_2, \lambda_3, \cdots \lambda_c$) have these parameters { π, *w*, *v*, aij , bjk }. With observation sequence (v^T) and model (λ), what is the probability that (λ) generated (v^T)? The solution to this valuation problem must choose the model that best matches the observations.

$$P(v^T \mid \lambda)$$

2) Decoding problem

Given the observation sequence (v^T) and the model (λ), there must be an optimal corresponding state sequence (w^T). The solution is to find which sequence (w^T) generated sequence (v^T). This problem attempts to uncover the hidden part of the model. In most cases, there is no correct solution, but usually optimal criteria are used to find best possible one. With an HMM model (λ) the first step is trying to find out all possible sequences of the hidden states w^T. The second step is trying to find out for each hidden state w^T the probability that a particular w^T has generated this visible sequence state v^T [6] by:

$$P(v^T \mid \lambda) = \sum_{r=1}^{r\max} p(v^T \mid w_r^T)\, p(w_r^T) \quad (5)$$

Where w_r^T = {*w(1), w(2),…, w(T)*} means all possible sequence of hidden states of length (*T*). The index (*r*) indicates one of the possible sequences of w^T and (*rmax)* is the number of possible sequences that can be generated. With *N* numbers of hidden states, that means $r\max = N^T$ number of possible sequences of hidden states of length (*T*).

To find particular w_r^T by applying the product of the transition probabilities from (*t= 1 to T*) as:

$$p(w_r^T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}) \quad (6)$$

To compute the visible symbols v^T by taking the product of probability of the emission:

$$p(v^T \mid w_r^T) = \prod_{t=1}^{T} p(v_t \mid w_t) \quad (7)$$

The expression below is to find that given hidden Markov model λ generated the visible symbols v^T :

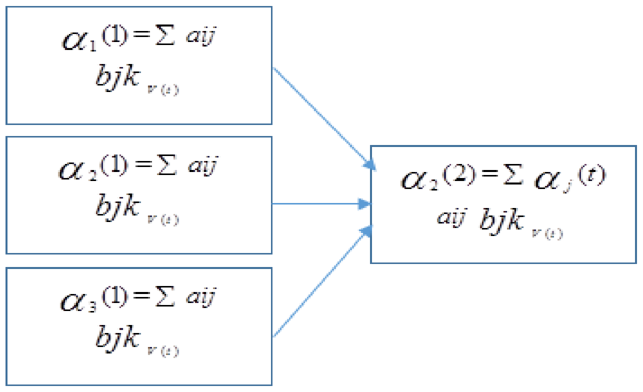
$\alpha_1(1) = \Sigma\ aij\ bjk_{v(t)}$
$\alpha_2(1) = \Sigma\ aij\ bjk_{v(t)}$
$\alpha_3(1) = \Sigma\ aij\ bjk_{v(t)}$
$\alpha_2(2) = \Sigma\ \alpha_j(t)\ aij\ bjk_{v(t)}$

Fig. 4. trellis Diagram (B)

$$P(v^T \mid \lambda) = \sum_{r=1}^{r\max} \prod_{t=1}^{T} p(v_t \mid w_t) \cdot p(w_t \mid w_{t-1}) \quad (8)$$

This expression's *O* = (N^T *.T*) complexity is substantial. Calculating the probability in this manner is therefore computationally expensive, in particular with large models or long sequences. The coming forward algorithm will solve this problem forward algorithm [6].

3) Learning problem

Giving a sequence of hidden sates (w^T) and a sequence of visible states (v^T) infers a number of transition probabilities. In other words, the transition probabilities of (aij) and (bjk) must be estimated, leading to each model working well. This 'learning problem' is one of the most important problems, not only in case of HMM, but for designing any classification [6].

IV. The Algorithms are Used in HMM

A. Forward algorithm (to solve the problem of evaluation)

The evaluation problem explains using recursive algorithms to reduce the complexity of direct expression. Given a sequence of visible symbols v^T, what is the probability that the hidden Markov model will be in a particular state at a particular time w_r^T ? Consider the particular state is $w_{r=2}^{t=2}$ so, what are the possible ways that the process can come in this state? As trellis below:

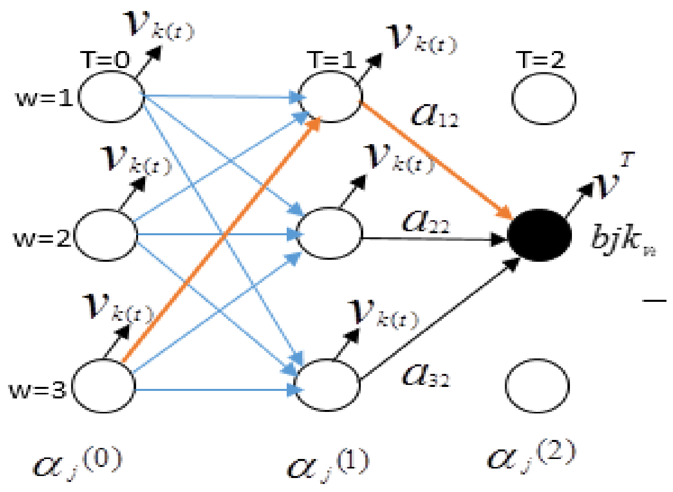

Figure4 trellis Diagram (A)

It is possible to compute $\alpha_j(t)$ By by knowing the HMM's parameters *A*, *B*, π; $\alpha_j(t)$ is also the probability for reaching an intermediate state in the trellis.

To calculate $\alpha_j(t)$ probability for reaching an intermediate state, one must take the sum of all possible paths to that state (every path can be computed as: multiply aij by $bjk_{V(t)}$ and by $\alpha_j(t)$ as shows in figure (4 -B)). This more or less means using $\alpha_j(1)$ to calculate $\alpha_j(2)$, using $\alpha_j(2)$ to calculate $\alpha_j(3)$, and so on until *t* = *T*. The $\alpha_j(t)$. [3] [6].

$$\alpha_j(t) = \begin{cases} 0 & t = 0 \text{ and } j \text{ not initial state} \\ 1 & t = 0 \text{ and } j \text{ is initial state} \\ \left[\sum_i \alpha_j(t-1)\,aij\right] bjk_{V(t)} & \end{cases}$$

To wire formal steps for forward algorithm [6]:

```
Initialize: t =0, $bjk_{V(t)}$, $aij$, $v^T$, $\alpha_j(0)$.
  For   t = t+1
    $\alpha_j(t) = bjk_{V(t)} \cdot \sum_{i=1}^{N} \alpha_i(t-1)aij$
    Until t = T
  Return $P(v^T|\lambda)$ = $\alpha_0(T)$  for final state
  End
```

B. Viterbi Algorithm to solve problem of decoding

This algorithm has ability finds the probable sequence of a hidden state through which the process has made the transition while generating v^T's sequences. Simply, at every step time *T*, the algorithm considers the most probability state among w_r^T. At the end of the process there will be a number of the most probability sequence of hidden states w_r^T that generating v^T's sequences. Figure (4-A) considers the most probability state on step *t=0* is state w_3^0 but only after generating the first symbol of sequences v^T, in step *t=1* is state w_1^1 after generating the first two symbols of sequences v^T. This goes on until having reached the final state in the sequence, the observing state w_2^2. As a result, the path

($\longrightarrow$) is: $w^T = \{w_3^0, w_1^1, w_2^2\}$.

To wire a formal step for the Viterbi algorithm [6]:

```
Initialize:     path= { } t = 0, j = 0;
      For   t = t+1, j = j+1;
         For  j = j+1;
               $\alpha_j(t) = bjk_{V(t)} \cdot \sum_{i=1}^{N} \alpha_i(t-1)aij$;
         Until  j =N;
        j' = arg max $\alpha_j(t)$
     Append $w_{j'}$ to path
   Until t= T
  Return Path
End
```

Then, applying bays' rule to classified the sequence of the symbols v^T that has been generated by model λ [2]

$$P(\lambda|v^T) = \frac{p(v^T|\lambda)\cdot p(\lambda)}{p(v^T)} \quad (9)$$

This quantity $P(\lambda \mid v^T)$ can be used for classification. For example, by computing two models (λ_i) and (λ_j) in the same way, the result is $\lambda_i > \lambda_j$ then obviously the v^T belongs to class λ_i.

$$P(\lambda_i \mid v^T) \ > \ P(\lambda_j \mid v^T) \therefore v^T \in \lambda_i$$

C. Forward-Backward or Baum-Welch algorithm (to solve learning problems)

The learning or training process of a hidden Markov model is actually supervised learning. Indeed, the number of sequences of visible symbols and hidden states are already known. Therefore, the goal is to estimate the transition probability matrix aij and bjk with an algorithm similar to forward algorithm, called backward algorithm. The idea of backward algorithm is finding the probability that the process will be in particular state $w_i(t)$ and thereby generate the remaining part of the sequence of visible symbols v^T. The definition of this algorithm is [6]:

$$\beta_i(t) = \begin{cases} 0 & w_i(t) \neq w_0 \text{ and } \boldsymbol{t=T} \quad \text{where } w_0 \text{ is final state} \\ 1 & w_i(t) = w_0 \text{ and } \boldsymbol{t=T} \text{ initial state} \\ \sum_j \beta_j(t+1) aij . b_{jkv(t+1)} \ bjk_{v(t)} & \end{cases}$$

To wire formal steps for backward algorithm:

```
Initialize: $\beta_i(T)$ , $bjk$ , $aij$ , $v^T$ , , t = T.
For    t = t-1
    $\beta_i(t) = \sum_j \beta_j(t+1) aij . b_{jkv(t)}$
  Until  t = 1
Return $P(v^T)$ = $\beta_i(0)$ for the known initial state.
End
```

Using $\beta_i(t)$ and $\alpha_j(t)$ to find the probability transition is not correct because the exact value of transition aij and bjk is not known. This means using γ_{ij} to define the probability of transition from state $w_i(t-1)$ to state $w_j(t)$ for a particular sequence v^T by initially using a random value of aij and bjk to estimate $\beta_i(t)$ and $\alpha_j(t)$ as below:

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1) a_{ij} b_{jk} \cdot \beta_j(t)}{p(v^T \mid \lambda)} \tag{10}$$

After defining γ_{ij} one must then find the expected number of transitions from $w_i(t-1)$ to $w_j(t)$ at any time in the sequence v^T by:

$$\sum_{t=1}^{T} \gamma_{ij}(t). \tag{11}$$

After that, finding the total expected number of transitions from state w_i to any state by:

$$\sum_{t=1}^{T} \sum_k \gamma_{ik}(t). \tag{12}$$

There are two quantities: the total expected number of transitions from state w_i to state w_j in the sequence v^T and the total number of expected transitions from state w_i to another state. The next step is finding the transition probability aij [6][4]:

$$\hat{aji} = \frac{\sum_{t=1}^{T} \gamma ij(t)}{\sum_{t=1}^{T} \sum_k \gamma_{ik}(t)} \tag{13}$$

One can also find transition probability bjk in the same way by applying:

$$\hat{bji} = \frac{\substack{\sum_{t=1}^{T} \sum_l \gamma_{il}(t) \\ v(t) = v(k)}}{\sum_{t=1}^{T} \sum_l \gamma_{il}(t)} \tag{14}$$

V. Real-World Applications

A. Security

1) Credit Card Fraud Detection

Demand for access to online shopping is increasing rapidly with the most popular method of payment being the credit card. Indeed, more than 350 million transactions per year are made in USA including online and regular purchases. As the number of credit card users increases, opportunities for attackers to steal credit card details also increase [8]. There are two types of credit card purchases:

1- Physical credit card: This entails physically using the card in-store to purchase items. To make a fraudulent transaction in such case the attacker need to physically steal the credit card.

2-Virtual credit card: This is use of the credit card without it being physically present; some sensitive information about a credit card is needed here in order for a thief to capitalize on someone else's card (card number, expiration date, secure code) [8].

Virtual credit card fraud can be detected based on the analysis of existing purchase data usually found through studying card behaviors. Card behaviors are stored on the cardholder file. By analyzing purchase patterns, it is possible to figure out the abnormal patterns, then detect the fraud. If human behavior is modeled in a perfect way, any abnormal behavior will be detected because the attacker does not have the same behavior of user.

The hidden Markov model has the power to model much more complicated stochastic processes than Markov model. The key idea of this paper is to build a model with multiple layers of behaviors based on HMM and enumerating methods for normal detection. The bank who issued the credit card has an FDS system detailing when any transaction is made, so that any detection will first send go to FDS for verification purpose. The DFS then receives card details and transaction details such as value of purchase, though the types of items bought are hidden. If the transaction is unknown to the FDS, then it will try to find any abnormal information in the transaction based on the spending profile of the cardholder. This might include shipping address, billing address, etc. If the FDS confirms the transaction as fraud, it raises an alarm, whereby issuing bank will reject the transaction [7] [8]. As the cardholder is using his credit card and making various purchases with different amounts over time, this sequence of types can be used it to create an HMM model. To map the credit card transactions to the HMM model there are a few steps:

1) Define the range of price for example M = {high, medium, low} and this range is considered as K-means in the clustering algorithm.

2) Define the observation symbols Vk, k= {1, 2, 3,….., M} where v is the price of the item. By applying the K-means algorithm on the sequence of observation symbols, the v can be clustered in one of the categories v = {high, medium, low}.

3) The cardholder purchasing is dependent on his need, and every transaction amount is dependent on the corresponding type of purchase, making it so that the transaction amount can be considered as state on the model.

4) Each type of purchase (groceries, electronics items, miscellaneous, etc.) links to a line of business with a corresponding merchant. The information about the merchant's line of business is not known to the bank running the FDS. On the other hand, because this information is required on the stage of registration of the merchant, the merchant is known to the acquiring bank. In addition, some stores have a wide range of varying items. In such cases, the store will be considered miscellaneous; there is no need to determine the actual type of items because any assumption about the information will not be accepted from the bank or from the FDS anyway.

5) The last step is to determine the probability matrix A, B and the initial state – a very important aspect for the Baum-Welch algorithm; that algorithm will then determine these parameters.

1.2. Dynamic Generation of Observation Symbols:

This paper applies the clustering algorithm (K-means) found on each past cardholder transition to find the observation symbols. In fact, the database of the bank, which has several types of information, stores these transitions. However, in this model the amount of the transition is used. In the example mentioned in this paper (table 2) the K-means equals observation symbols (M=3) so k= { c_l, c_m, c_h } [9]. These means are responsible for clustering the amount of the new arriving transaction. FDS generates the observation symbol with:

$$O_x = V_{\arg\min|x-ci|}$$

Where x is the amount the new transaction shows having been spent.

1.3. Spending Profile of Cardholders:

A spending profile for normal behavior of cardholders builds over time. As in the example there are three categories {high, medium, low} and cardholders will assign to corresponding categories based on spending habit. For example corresponding (1) spends high amount when he uses his credit card, then he can be in group {high}. The clustering finishes the profile. Consider that p_i is the percentage of the total number of cardholders' transitions that belong to the mean c_i then the spending profile for cardholder (u) is:

$$SP(u) = \arg\max(\, p_i)$$

The spending profile (SP) of the cardholder thus determines that the most transactions are in group {high, medium, low}. This paper uses the Baum-Welch algorithm to estimate the HMM parameters (probability matrix A, B and the initial state) for each cardholder. The basic idea is that a cardholder's already existing spending profile can make the initial estimate more accurate. In the three categories {high, medium, low} and base on the cardholder spending profile, the initial state can be chosen. In the phase of training HMM model, there are three steps:

1) Initialization of HMM parameters,

2) Forward procedure

3) Backward procedure.

This stage will eventually create an HMM for each cardholder. After learning HMM, the second step is trying to detect fraudulent transactions. This will be done by taking the symbols from a cardholder's training data and forming an initial sequence of symbols. Let ($O_1, O_2, \dots, O_R$) be one such sequence up to time *T*, using HMM to compute the probability acceptance of this sequence as follows:

$$\alpha_1 = p(O_1, O_2, O_3, \cdots O_R \mid \lambda)$$

Where α_i the probability and *R* is the length of the sequence. Let O_{R+1} represent a new transaction at a certain time *T+1*. To compute the acceptance probability, fires drop the O_1 from the previous sequence and add O_{R+1} to this sequence. One can then re-enter it to HMM as a new sequence as follows:

$$\alpha_2 = p(O_2, O_3, O_4 \cdots O_{R+1} \mid \lambda)$$
$$\Delta\alpha = \alpha_1 - \alpha_2$$

If $\Delta\alpha > 0$, the HMM has therefore accepted the sequence, indicating a likelihood of fraudulent activity on the credit card. To determine if the new transaction is fraudulent or not, one can simply evaluate if the percentage change in the probability is above a determined threshold.

$$\Delta\alpha \,/\, \alpha_1 \geq threshold$$

Figure	**credit card fraud detection using HMM**	**credit card fraud detection technique**
Figure 5 (a)	-TP is close to credit card the fraud detection technique. - FP is the same for both models.	
Figure 5 (b)	- Two models have accuracy and an average of TP-FP spread. - same exhibit with variation in μ	
Figure 6 (a)	- TP rate is low but increasing - FP rate is low	- TP rate is low - FP rate is obviously high
Figure 6 (b)	- accuracy is 80%	- accuracy is 60%
Figure 7 (a)	- TP rate is low but it is still increasing - FP rate is low	- TP rate is low - FP rate is obviously high
Figure 7 (b)	- accuracy is 80%	- accuracy is 80%
Figure 8 (a)	- TP rate is low - FP rate has changed little, still low	- FP rate is obviously high and higher than TP
Figure 8 (b)	- accuracy is still around 80% - negative value with μ = 2.5	- accuracy is below 40% - negative value with μ = 0.5

For example, if transaction O_{R+1} is fraudulent the bank will decline the transaction and the FDS will reject the symbol. If it is not, the symbol will be added to the normal behavior used to capture any change in spending behavior of a cardholder in a future transaction [8].

1.4. Comparative Performance:

Measuring the performance of HMM models needs a substantial amount of real world data, but because this information is sensitive, banks will never open the information up to researchers. The paper tries to evaluate the model and compares it with a credit card fraud detection technique proposed by Stolfo et al [9]. The methods used in this experiment are:

- Metrics True Positive (TP): to detect and identify fraudulent transactions correctly.

- (FP): means the genuine transactions identified as fraud transactions.

- TP-FP: find the difference between two values to measure the effectiveness of the systems.

- Accuracy: the total number of transactions detected correctly, both fraudulent and genuine.

The experiments are based on a mixed sequence of fraudulent transactions with a sequence of genuine transactions. The cardholder's profile captures these transactions. Using three profiles { high, medium, low} and the value of the three profiles as P1(55,35,10), P2(70,20,10), P3(95,3,2). The experiment figure results can be found in [9].

As a result, using the HMM for fraudulent activity detection is more accurate than other techniques. Also, the model does not need to have prior knowledge about the fraud transactions to build the learning of the model; this is important because banks consider such information sensitive. In addition, this model can validate the transaction offline so it could not possibly affect credit card transaction processing performance, which needs an online response [9].

2) Predicting Multistage Attacks in Cloud Systems

The Internet provides people with a lot of services to make life easier. One of these services is cloud computing. While this is mostly a helpful and advantageous service, attackers have more opportunity to exploit vulnerabilities with malicious code. Indeed, most security technologies have no ability to raise an early alert about such attacks. This paper is provides a prediction technique based on the hidden Markov model so as to foresee multi-staged cloud attacks. Cloud computing is more difficult in the field of security than in a traditional platform due to shared resources between unknown users and the lack of control of data location. Intrusion Detection Systems (IDS) cannot handle such challenges because they do not incorporate risk assessment and prediction models [7]. This paper therefore implements an Autonomous Cloud Intrusion Detection Framework (ACIDF) so as to focus on such an issue. The purpose of this approach is to evaluate system vulnerability overall, providing control over the system for protection and the ability to respond when threats are predicted or detected.

a) Description of ACIDF:

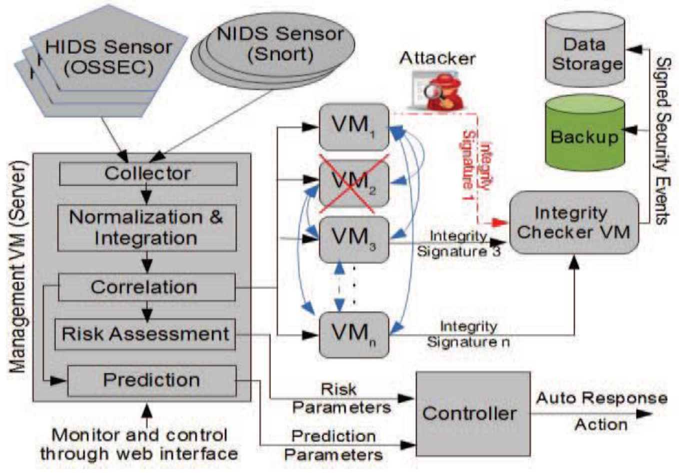

Fig. 5. overview of ACIDF diagram

There are six processes in the model as follows:

1- **Collection process**: this process has three fundamental functions: collecting logs, monitoring network packets, and scanning hosts. The collection process is responsible for collecting all logs and events from sensors such as Host Intrusion Detection Systems (HIDS) and Network Intrusion Detection Systems (NIDS), then redirecting them to the integration process.

2-Integration process: the purpose of this process is to integrate and normalize all collected data from the previous process into the Intrusion Detection Message Exchange Format (IDMEF) protocol to identify their correlation.

3- **Correlation process:** this process creates a correlation between a huge number of normalized events and logs from several sensors and finds the suspected ones by comparing each event with rules of attack. This at the same time reduces false positives.

4- **Risk Assessment process:** the process uses the formula below to evaluate the risk of every group of alerts and then decides if the risk is larger than or equal to one. If it is, an alarm will be fired:

*RISK= (AssetValue * AlertPriority * DetectionReliability)/ NF (1)*

5-Prediction process: works with the correlation process. In case of an attack the prediction component will respond.

6-Auto response process: in this process, the controller uses a fuzzy logic approach to choose the suitable response to protect hosts and the network from attacks.

The outputs of IDS are a huge number of disordered alerts and changes considerably. IDS need a model to deal with this challenge. The best way is using a hidden Markov model. This model works with streaming inputs and can predict future threats in IDS. This paper condones using HMM to trace and evaluate the attacker's actions while proceeding. The hidden state in this model is the sequence of event states that matches the rule of the attacks. However, the observation states are the name of the attack.

b) Explanation of the prediction model

The paper divides the perdition model into 10 elements:

1- **States:** there are four states**:** Hale (*H*) the system is normal with no perceived threat, Investigate (*I*) there is malicious code attempting to compromise the system, Attack (*A*): the malicious code has been executed, and Penetrate (*P*): the malicious code successfully compromised the system.

2- The system is in one of these states. Moreover, like in HMM, states can transition freely between themselves. That will lead to detecting single attacks and predicting multistage attacks.

3- **Observations**: $O = o1, \ldots, oK$, represents the alerts which come from sensors. There are four levels for these states depending on the risk of each alert Low, Medium, High, and Very high or (*L, M, H, V*). This section later describes the alert severity function.

4- **State Transition Probability Matrix (*P*):** explains the probability of moving among states. To create states and calculate the transition possibilities there are three steps:

a) Define each attack in a sequence of states, depending on the signature.

b) Determine possible signatures possibly being used by more than one attack and create sequence states, then minimalize the states as much as possible.

c) Calculate the transition probability between states using the Forward-Back algorithm.

5- **Observation Transition Probability Matrix (*Q*):** Calculate the transition probability matrix for observation states.

6- Initial State Distribution Vector (*π)*: for indicating the initial state.

7- **Alert Observation Probability Matrix (*Å*):** if a state has a particular alert, this matrix will detect it. (*Å*) is built based on the training data in the attack dataset.

8- **Assets Cost Matrix (*C*):** using cost vector to find out possibility of consequences for each state in question format.

8- **The Output or emission probability Matrix (*Y*):** showing the output of each sequence of attack states

9- **Alert Severity Function:** This function explains why the severity of each alert is at a particular state. It can be computed using Eq.1, Eq.2 and 3. The result is then mapped to one of the four observation stats (*L, M, H, V*) to clarify the current state of the system.

$$AR^S = (AC^S * AP * DR^S)/NF^S \quad (2)$$

$$= (AC^S * \mathrm{C}_{\mathrm{Severity}} * \mathrm{N}_{\mathrm{Occurance}} * A_{\mathrm{Frequency}} * DR^S)/NF^S \quad (3)$$

10- **HMM Prediction Algorithm:** The algorithm is for computing the alert risk and then mapping this risk to an observation states (*L, M, H, V*).

ACIDF predicts against attacks based on the hidden Markov Model by considering signatures of attacks as a sequence of hidden states. Any future attack can be stopped with this model, based on collecting alerts. Furthermore, “the model uses a training algorithm to find the transition, output probabilities, and other prediction parameters.” [10].

B. Engineering

1) Improving time series classification using HMM

This paper proposes a method called a multiple classifier to improve the accuracy of time series by adding an HMM model. This will take into account the temporality of the data, and execute a second stage classification. Moreover, a single classifier system is less powerful than multiple classifier systems. This method assumes that the machine generating the data works under a number of hidden states. That assumption works in many time series datasets, such as the sensors of gas drilling systems.

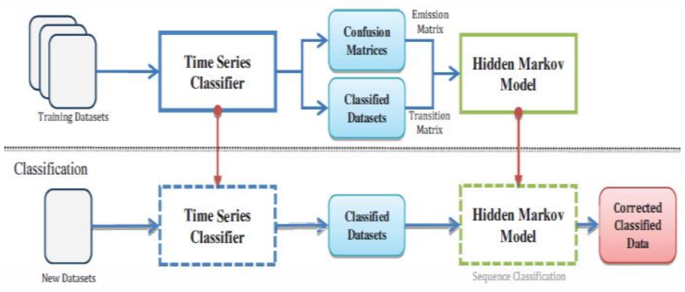

Fig. 6. overview of the model [11]

An overview of the model is in Figure (6). This model has two stages:

1-Training stage: the main purpose of this stage is training the model. The output of the classifier will generate two components: confusion matrices and classified datasets. These components will train the model.

2-Classification stage: this stage is concerned with reclassifying the sequence of classified samples by using the trained model. If there are mistakes on some classified samples, the HMM will correct them.

This model looks at the classification process itself with hidden states and observation states, making HMM suitable for that concept. HMM classifies the data based on varying temporal relations; this is the main difference between HMM and other traditional machine learning classifiers, such as SVM (support vector machines) and NN (Nearest Neighbors algorithm). In order to classify a given sequence of observations states, the most important thing is to find the best sequence of states that generated these observations. This will be done by using the Viterbi algorithm [11]. This paper does some experiments on the model by using real time datasets from different drilling scenarios. The result show that the accuracy of the classification is improved than using single classification.

VI. Conclusion

This survey paper addressed the Hidden Markov model. It explained the traditional Markov model in a simple case and shows that in some states, an observed sequence is probabilistically related to a hidden state. In addition, any real HMM has three central problems: 1) Evaluation problem: given observations sequence and hidden model, what is the probability that observations sequence was generated by that model? The forward algorithm solved this problem. 2) Decoding problem: what sequence of hidden states most probably generated a given sequence of observations? The Viterbi algorithm solved this problem. 3) Learning problem: By giving a sequence of hidden sates and a sequence of visible states, what are the transition probabilities? The forward-backward algorithm solved this problem. Also, this paper shows three real world applications for HMM: 1) Credit Card Fraud Detection; 2) Predicting multistage attacks in cloud systems; and 3) Improving time series classification. The result of these applications has shown that using HMMs provide more advantages such as better accuracy, more reliable, improve the classification, and predicts the future events for the systems.

References

[1] L. R. Rabiner, and B. H. Juang “*An Introduction to Hidden Markov Models*”, IEEE ASSP MAGAZINE, Vol. 3, no.1, 1986, pp.4 – 16.

[2] Fink, Gernot A. “*Foundations of Mathematical Statistics*” Markov Models for Pattern Recognition: From Theory to Applications.2nd ed, London, Springer, 2014, pp.32-49

[3] University of Leeds, “*Hidden Markov Model*” http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/main.htm. Nov 1st, 2015.

[4] L. R. Rabiner, "*A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*" Proceedings of the IEEE, Vol.77, No.2, 1989, pp.257-286.

[5] Alpaydin, E. “*Hidden Markov Models*” Introduction to Machine Learning, 3ed ed. The MIT Press Cambridge, Massachusetts London, England, pp.363 – 385, 2010.

[6] Richard O. Duda, Peter E. Hart and David G. Stork. “*Maximum likelihood and Bayesian estimation*” Pattern Classification 2nd ed., Wiley, New York, 2010, pp 40-55.

[7] Afroza Sultana, Abdelwahab Hamou-Lhadj, and Mario Couture, "*An Improved Hidden Markov Model for Anomaly Detection Using Frequent Common Patterns* ", IEEE ICC Communication and Information Systems Security Symposium, IEEE, Ottawa, ON, 10-15 June, 2012, pp 1113 - 1117.

[8] Ayushi Agrawal, Shiv Kumar, and Amit Kumar Mishra “*Credit Card Fraud Detection: A Case Study*” 2nd International Conference on Computing for Sustainable Global Development, IEEE, New Delhi, 11-13 March, 2015.pp 5 – 7.

[9] Divya.Iyer,Arti Mohanpurkar,Sneha Janardhan,Dhanashree Rathod,and Amruta Sardeshmukh ”*Credit card fraud detection using Hidden Markov Model*” Information and Communication Technologies (WICT), IEEE , Mumbai, India ,11-14 Dec , 2011, pp 1062 – 1066.

[10] Kholidy, Erradi, A, Abdelwahed, S, and Azab, A.”*A Finite State Hidden Markov Model for Predicting Multistage Attacks in Cloud Systems*”, IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, Dalian, 24-27 Aug, 2014, pp 14-19.

[11] Esmael, B. Arnaout, A. Fruhwirth, R.K. Thonhauser, G.” *Improving Time Series Classification U sing Hidden Markov Models”* 12th International Conference on Hybrid Intelligent Systems (HIS), Pune , 4-7 Dec, 2012, pp. 502 – 507.