Chapter 14

# MINING MALICIOUS CORRUPTION OF DATA WITH HIDDEN MARKOV MODELS

Daniel Barbará, Rajni Goel and Sushil Jajodia

**Abstract** Data mining algorithms have been applied to investigate a wide range of research issues recently. In this paper we describe an alternative technique of profiling databases via time series analysis to detect anomalous changes to a database. We view the history of modifications in the database as a set of time series sequences. We then examine the application of Hidden Markov models (HMMs)as a mining tool to capture the normal trend of a database's changes in transactions. Rather than examining each record independently, our technique accounts for the existence of relations among groups of records, and validates modifications to the sequence of transactions. The algorithm is adaptive to changes in behavior. Experiments with real data-sets, comparing various options for the initial HMM parameters, demonstrate that the distribution of the change in acceptance probabilities of anomalous values is significantly different from that of acceptance of transactions expected by the model.

## 1. Introduction

In spite of the efforts in the security community in coming up with techniques to detect intrusions, the area of insider threats has not been sufficiently covered. Detecting insider attacks is a challenging task, since the intruders use authorized channels to perpetrate the attack. In this paper, we describe a solution to the problem of detecting malicious modifications to data. We call a transaction that produces these modifications a *Malicious Database Transaction* (MDT). (Notice that the actual change in the data does not have to be produced by a real transaction, but could be produced by malicious manipulation of the physical database, but we use the term in a loose sense.) An ideal system would be able to monitor and analyze the data, detecting and reacting in a timely way to MDTs. We aim here for a technique that recognizes MDTs with a high probability. The motivation to develop our system is to provide additional protection against intrusions when traditional access-control-based

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35697-6_26

E. Gudes et al. (eds.), *Research Directions in Data and Applications Security*

security methods, such as intrusion detection tools, fail (i.e., provide the detection layer in Information Warfare [9]).

This paper proposes a novel use of time series and Hidden Markov Models (HMM). Using HMMs, the technique builds database behavioral models, capturing the changing behavior over time, and uses them to recognize patterns that humans are not necessarily good at discerning. The model employs relations among similar events among past and current behavior. The system then validates illegitimate modifications in a transactional database (occurrence of an MDT) in real-time. Though HMMs have previously been utilized in numerous applications [11, 13, 16], we propose constructing an HMM to validate and reject modifications in a sequence rather than to match patterns and classify.

An MDT could be the work of an intruder that has bypassed the access controls, or a legitimate user misusing the system. We establish the patterns that hold for data in the absence of malicious behavior, so later these patterns can be used to check possible MTDs. But, instead of modeling unique individual behaviors as other profiling architectures do, we propose to study the nature of the database, and develop a model useful for capturing the otherwise unobservable patterns in the overall data transactions over time. The conjecture is that inside intruders who, being authorized to change the data, do so in "strange" ways will cause drastic changes that will contrast with the patterns previously profiled. These changes can be then flagged as intrusions and investigated further.

One problem is the difficulty of modeling the behavior of the entire database due to enormous size restrictions. To reduce the complexity, we experimented with clustering data items that exhibit similar patterns of change. This provided us with a set of time series sequences representing activity of each record as well as the capability to view how the sequences behave with respect to other group sequences during a stretch of time. The technique abstractly views the database transactional changes as a sequence of events, in which a time is associated with the occurrence of each event. Often, groups of records in a database share similar values for several attributes [5, 8]. So, each record's transactional behavior is examined not independently, but rather in the context of other similar records. An underlying hypothesis is that the database transactions occur in a similar manner at various intervals of time, leading to repeated sequences of behavior.

The technique introduces a unique concept of delta alpha ($\Delta\alpha$) to determine the validity of a transaction at time $T+1$, knowing the history of the database up to time $T$. To determine whether or not a given sequence of $T+1$ measures is valid, we take the difference between the acceptance probability ($\alpha$) of the sequence up to time $T$ and the acceptance probability of the entire $T+1$-length sequence. This difference, $\Delta\alpha$, for valid $T+1$ elements is the threshold upon which validation of an atypical modification is alarmed. The algorithm can

be generalized to validate any modification to an element in the sequence of transactions, not just future values.

Consider, for example, the problem of detecting abuse of long distance telephone calls by employees. Some corporations have installed PINs in order to track long distance calls, but we know that employees share the PINs with employees in departments not authorized to make calls, and also that PINs are susceptible to dictionary attacks by unauthorized employees. A system built on user profiling would provide defense against someone inside the corporation. The current technologies do not consider clustering the corporation database by departments and modeling the calling behavior of each department. Using our approach, a corporation would train the HMM for each department (Sales, Marketing, etc.) for weekly call logs for 4-8 weeks. People with similar job and telephone call requirements would be clustered to reduce the amount of data for training of the HMM. Then, if an illegitimate user employs the PIN, she would need to know the usage behavior (trend over time) of that particular record in relation to others in the department in order to bypass the security system proposed here. This model would also detect call abuse by any new employee without requiring any historical data to train the model for that employee, because that employee is part of an existing modelled cluster. This also protects the privacy of the employees while being capable of detecting call abuse. A trained model would be tested and calibrated to detect future call abuse by the employees.

This paper proceeds as follows: Section 2 is a review of prior work in the areas of time series, anomaly detection, and applications of HMMs. Section 3 discusses our proposed techniques and architecture. We present our experiments and results in Section 4, followed by our conclusions.

## 2. Related Work

Distinguishing between normal and atypical behavior is not a new problem. Machine learning techniques have been employed to address it [6, 7, 10]. Profiling, automated modeling of user behaviors, is often utilized for anomaly detection for information security purposes. The existing models either classify actions, based on rules [11], or generate the single most likely future value [17]. Some only consider single user profiling [7, 11, 12]. Davison and Hirsh [4] present a more general framework relating to predicting a user's next action; they employ a machine learning tool, C4.5. Much of the prior work has focused on modeling individual user behavior and classifying new events as normal or illegitimate, alarming at values which are "atypical" of the model or do not follow a specific rule [7, 10]. These architectures are rule based classifications or predictive models.

Fraud and corruption in applications such as credit card processing, telecommunications, and banking continue to be a security issue. The profiling problem has been studied by Fawcett and Provost [6] within the context of fraud detection in cellular phone usage. This work focuses on learning rules pertaining to individual customers from the cellular phone usage data and then generating generalized profilers for different customer segments [17]. Adomavicius and Tuzhilin extended the concept of rules learned from user transactional histories using various data mining techniques [3]. These approaches are rule-based and use classifiers, validation of accuracy, and rule priorities in order to flag irregular transactions. They do not consider the time or the notion of building a model for sub-groups of a database. They neglect taking into account relations that logically clustered records have among one another. They provide a boolean acceptance result, rather than a probability of acceptance of upcoming transactions. Also, once the rules are established, if an attacker modifies a stored value, no alarm is raised.

Another challenge not yet fully overcome is to have the system adapt to new and changing behaviors, so that the rules or models are self-learning. If a system could automatically determine the trends, rules, and patterns in each cluster, it could utilize them to validate potential anomalous (sometimes throughout the paper we use the term "bogus" to indicate anomalous) deviations in the dataset. Hence, the task is to accurately model the differences or behavioral changes within a set of sequences, using deviations from the "patterned" changes to detect anomalous, corrupted data.

Time series forecasting has been researched for numerous applications. Traditionally, the Auto-Regressive Integrated Moving Average [2] was a form of linear time-series forecasting, and recently, non-linear techniques, such as neural networks have been explored [5]. The closest work relating to databases is that of time sequences and similarity searching among them.

HMMs are statistical models of sequential data that have been used successfully in many applications in artificial intelligence, pattern recognition, speech recognition, modeling of biological sequences, and picture and handwritten text matching [1, 13–15]. Literature indicates employment of HMMs as temporal classification systems. HMMs have been used in security research for intruder detection; They classify anomalous system calls or UNIX command lines [11, 16]. To our knowledge, we are the first to consider applying the HMM approach in the application domain of detecting MDTs with the purpose of validation rather than classification.

One approach that utilizes co-evolving sequence prediction with the concept of linear combinations and tracking windows is *Muscles* [17]. That work focuses on predictions and missing values, whereas our work is for validating a potential anomalous value to a degree of certainty. Our algorithm is not limited to generating a single most likely future occupancy. Another key factor is that

we consider the arithmetic differences in measure, not the absolute measure itself. In this manner, the model determines if an update caused a change which was significantly different than the accepted level of change.

Evaluating the current wide range of tools for identifying unusual behavior, the need to provide a degree of validity still exists. Our research combines the many avenues explored by other researchers in that we consider the advantage of clustering (as did MUSCLES), and the need to explore new tools to profile (as did Terrane Lane), as well as a tool that has the ability to be adaptive (Hirsh) and furthermore the novel approach to validate, not just classify or predict. The following problems are not all addressed by any single state of the art system:

1 Lack of adaptiveness: Techniques are unable to account for changing behavior of users nor the seasonal aspect of the database. Models are rarely incrementally updated.

2 Lack of record correlation. Most of the techniques do not consider the relation of records having commonalities in attributes. Thus, they miss capturing knowledge of how the transactions within a database behave over time.

3 Prediction algorithms only return the single most likely future value. They do not address degrees of validity of upcoming transactions. This may be a cause of high false alarms unless a list of possible range of values is specified.

4 The existing algorithms utilize the domain values of the measures as they appear in the database, rather than modeling the change in a database transaction over the time. A model should be able to correctly validate the increments of change from time $T$ to $T + 1$.

## 3. Our Technique

In order to manage the tracking of the transactions in the entire database, any data reduction technique can first be applied. We utilize clustering for this purpose. We create clusters on attribute values to represent various (disjoint) regions of the database. So, the history of the database, $S$, transactions, now is represented by $m$ clusters containing $n$ sequences over a period of $t$.

*Definition*: Given a set of sequences $S_k$, each sequence $s_{ij} \in S_k$ is an ordered time series of data such that $1 \leq i \leq n$ and $1 \leq j \leq t$. Each sequence in $S_k$ is ordered by time, i.e. $p \leq q \Leftrightarrow s_{ip}$ value occurred before $s_{iq}$ $^{\text{th}}$ value.

Each $S_k$ represents a cluster and collectively all $S_k$ encompass the information upon which the security violation alarms will be issued. Given a set of $k$

sequences in the set $S$ where each sequence has an element at each time unit, $t$, each cluster is a set of time series. The problem becomes how to validate the accuracy of the sequence value at time $t + 1$ as following the trend of ordered values for sequences in the set $S$.

We choose HMMs as the tool to model the behavior of clusters in the database. The relation among the measure value changes in the records at specified time granularities are captured in the hidden Markov models. These state-based statistical models can output the probabilities of acceptance of valid future values of the records. We construct an HMM to accept specific historical sequences with high probability. The sequences represent the change in the database measure value from one time granularity to the next. The objective is then to see if the arithmetic difference of a measure from time $T$ to $T + 1$ is accepted to some degree by the model. This HMM will give the probability of acceptance ($\alpha$) of an unknown sequence. If the change in acceptance probability ($\Delta\alpha$) is outside the established threshold, a system acceptance results from different training strategies, as well the effects of varying the number of states in the HMM model.

Our belief is that this change in any one record can be evaluated as valid or invalid based on both all the past information about this record and the activities occurring in other "similar" sequences in the database. For example, if a bank account balance is being tracked and the values during a day to day observance were as follows: 40, 50, 52, 56, 58, 41, 39, then the sequence we would model is 10, 2, 4, 2,-17, -2. Our assumption is a presence of an underlying trend in the transactions of the database in question. The system must monitor all records of the database continuously, and set off alarms when suspicious activities are found.

## 3.1 HMM

We utilized a tool implementing the left-to-right HMM model [13]. (In a left-to-right HMM, the underlying state sequence associated with the model has the property that as time increases, the state index increases or stays the same). This type of modeling has been applied to time signals such as those found in speech (e.g., see [13]) and cursive handwritten text [1].

We use a variation of Rabiner's HMM notation [13]. An HMM is doubly stochastic process that contains a non-observable underlying stochastic process (hidden) that can be uncovered by a set of stochastic processes that produce the sequence of observed symbols [1].

An HMM is a nondeterministic state machine. At each time increment, it emits a symbol from a finite alphabet and potentially transitions to a different state. In a left-to-right HMM, this different state is the unique next state. Any given string has a finite probability of being emitted by a given HMM. Math-

ematically, an HMM is a tuple $\langle \Sigma, Q, a, b, \pi \rangle$, respectively, an alphabet, a set of states, transition probabilities, emission probabilities, and the initial state distribution.

Given an HMM $H$, and a pattern $S$ represented by a list of $T$ symbols, i.e. $S = [s_1 s_2 \ldots s_T]$, we can compute the probability that $H$ accepts $S$ (Note that $T$ is time length for which database is profiled). Define $\alpha_t(i)$ to be the probability of seeing the partial observation sequence $s_1, s_2, \ldots s_t$ and ending at state $i$ at time $t$.

$\alpha_t$(i) = Probability($s_1, s_2 \ldots, s_t, q_t$=i)

Using the Forward-Backward algorithm, $\alpha_t(i)$ is as follows:

Probability(S) = $\sum_{i=1}^{N} \alpha_T(i)$

Note that the symbols discussed here are *output* symbols to the state machine. However, in practice, sequences are validated by computing the above probability, and the module which does this validation takes the symbols as *input.*

One major aspect to the model is the assignment of the initial probabilities. One possibility is randomly or uniformly assign the transition possibilities, then the re-estimation procedure (e.g. the Baum-Welch algorithm [13]) can be used to adjust the model parameters in order to maximize the probability of the observation pattern(s) being recognized by the model. Usually, this leads to finding a local maximum.

In determining whether the next element in the sequence is a potential alarm, we do not use directly the probability of acceptance of the HMM ($\alpha$). Rather, we employ the change of acceptance in the following way. Consider a string $o_1 \ldots o_n$, accepted by a given HMM model with probability $\alpha_n$. If we append a symbol $o_{n+1}$ to the string, we notice that the new string's acceptance probability $\alpha_{n+1}$ is slightly less than $\alpha_n$. We call this drop $\Delta\alpha$.

## 3.2 Proposed Architecture

The framework of our technique consists of three modules described below.

1 **Clustering module**: Preprocesses data achieving data reduction via clustering. One dimension of the database is chosen to be the *measure*, the value which is being tracked over time and whose validity may be in question. This module calculates the changes from measure to measure in order to establish a sequence of changes per data item. Then using these sequences, proceeds to partition them into clusters.

2 **Modeling module**: Builds the HMMs, $H_i$, one for each cluster, initializing parameters as described above, to recognize a class of patterns from that cluster. The HMM is stored along with the cluster. Various methods exist to train an HMM for a group of sequences. When the initial training is completed, it is possible to make the model adaptive to changing behavior over time. The models can be incrementally trained with the longer sequences with the extra elements of time t+1, t+2,.. As new transactions occur, the existing $H_i$ will be used to validate (see below) and then, if not a MDT, entire new sequence is used to update the $H_i$. The updated $H_i$ provides better acceptance probabilities, because the newer elements contribute to the trend. Depending on the data and application, another approach is to assign weights to newer elements when training, or, establish a window of length $w$, slide the window with time, keeping length of sequence consistent with original length.

3 **Validation module**: Identifies the sequence as anomalous or valid. This verification is dependent on $\Delta\alpha$, the acceptance probability of the sequence at time $t+1$, in relation to the acceptance probability of the sequence at prior time $t$. $\Delta\alpha$ is mathematically quantified and thresholds are established to alarm the occurrence of a sequence which produces a $\Delta\alpha$ that is significantly different from it's cluster's normal $\Delta\alpha$. At a certain time, the $\Delta\alpha$s become very similar for anomalous and legitimate values. This may lead to an analysis of when the appropriate time to regenerate the models may be.

Note that the validation module would also alarm if elements in the stored sequences were modified. Meaning, if at time $t+1$ the sequence length remained the same, but a prior value in the sequence was maliciously altered, the $\Delta\alpha$ would indicate an anomaly. Also, if the element at time $t+1$ is legitimate, but the stored element of the sequence is tampered with, the acceptance would be significantly different from that of the sequence having followed the trend of the cluster. The HMM captures the acceptance of the sequence, as an entity; thus, any part of the sequence not conforming to the trained model for the cluster is marked anomalous.

The input sequence is application-specific; in the end we just need a proposed new time sequence which is to be validated. Whether we somehow have the entire sequence or we retrieve the earlier elements when necessary depends on the implementation of the rest of the system. This is a tradeoff; an analysis of storage versus I/O costs would need to be performed.

# 4. The Experiment

## 4.1 Clustering and Training

In this section we empirically validate the technique. We used three years (1998-2000) of Monthly Income Census Data for modeling the database. This data was downloaded from http://www.bls.census.gov/. Each record in the census data is described by numerous attributes and the measure of the income of the person. In order to reduce the number of dimensions, we selected only 4 attributes out of few hundreds on which to group the data and the 1 income measure. Each record in the original datafile is formatted as five integers:

- Attribute 1: Total number of household members
- Attribute 2: Person's age
- Attribute 3: Highest level of school completed
- Attribute 4: Number of hours worked at main job per week
- Measure: Total family income in last 12 months

In order to get multiple training sequences to construct the HMM, we clustered similar types of people. We manually defined 540 clusters based on the values of the four attributes. This assists in developing an accurate HMM that captures the characteristics of and relations among the sequences in the cluster.

The following procedure is implemented for each cluster. The sequences on which the model is trained are first extracted from the original database. As described in the previous section, we create sequences in which each element is the arithmetic difference in the measure. Each time series includes monthly changes in the income for each record for the 12 month period within a cluster. A training set of 1000 sequences is established by selecting the sequences with the top 3 high frequency symbols at the last position of the sequence for 12, 13 and 14 months period.

An HMM was constructed for each cluster. We examined the possible number of states which would produce an optimal model. The conclusion upon experimentation with varying number of states: increasing the number of states indefinitely does not necessarily improve results. The HMM model was trained and tested for 6, 7, 8, 9 and 10 states. The acceptance probabilities improved from 6 states to 8 states, but then declined from 8 to 10 states. Thus, an optimum number of states can be established (more states does not always mean better acceptance probabilities). The Transition Matrix is initialized with uniform probability, and HMM is constructed for each cluster.

The first step in generating test sequences was to determine the symbol frequency for the last item in the 12 month period data. The test set was generated

by selecting another exclusive 1000 sequences with the same criterion as the training set. The set of anomalous sequences were generated by replacing the last symbol in the test sequences with a symbol with lowest frequency symbol in the entire cluster.

Model output for the test of 12 and 13 months period was further refined by taking the arithmetic differences between the 12 months and 13 months real set and 12 months and 13 month anomalous set. The anomalous values were generated either at random or by evaluating the occurrence of output symbols in the history of the sequence. The results were similar. The test sequences are evaluated by the HMM and the $\alpha$ for the sequence at time $t+1$ (now one element longer) is computed. This extra element is analogous to a transaction in the database at time $t+1$. The set of $\alpha$ values resulting from the HMMs varied for test sequences within a cluster, depending on what symbol was being tested as the value at the time $t+1$. But, the distribution of *change* in the $\alpha$ ($\Delta\alpha$) for acceptance of the $t$ length to the $t+1$ length was normal for both the real valued and the anomalous valued sequences.

## 4.2 Results

We discovered that a new sequence at time $t+1$, with an anomalous added transaction, shows a drop in the HMM acceptance probability. This drop, $\Delta\alpha$, is statistically significantly different from the change in acceptance of a sequence with a legitimate value at time $t+1$.

Note also that the distributions of the $\Delta\alpha$ for anomalous values at 12 length and 13 length sequences in Figure 1 and Figure 2 show the distance between the real and the anomalous values. We used the ROC curve (Figure 3) to determine the significance of these distributions. From this, we conclude that an anomalous value sequence can be identified (alarmed) when the $\Delta\alpha$ value deviates the expected $\Delta\alpha$ for the cluster at a specific time $t+1$.

We need to compute the probability of positive test results for corrupt sequence detection by our HMM and validate that we can very clearly differentiate the good sequence from the corrupted sequences (a plot of probability of correct diagnosis vs. the probability of false alarms). In order to compute this probability, we compute the Likelihood ratio, tangent line of a Receiver Operating Characteristic (ROC) curve. The ROC curve is a plot of the true positive rate against the false positive rate of a sensor. We use the ROC curve to show the relationship between these two rates for the different possible threshold (cutpoints) of HMM tests. An ROC curve demonstrates several things:

1 The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.

2 The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

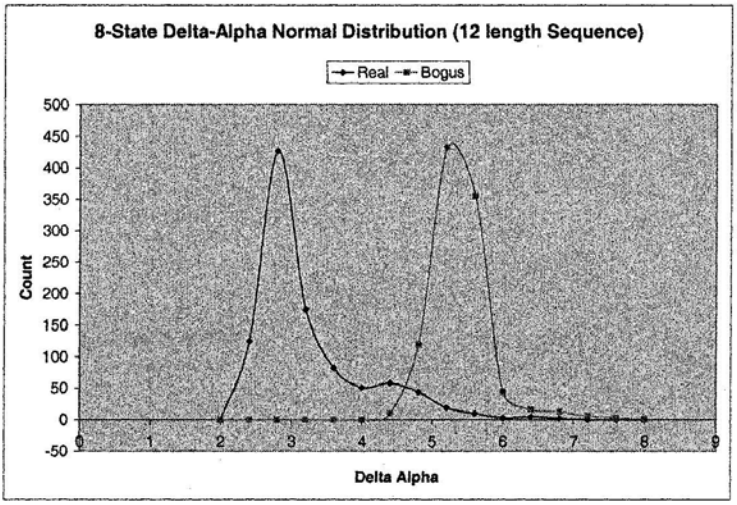


*Figure 1.* $\Delta\alpha$ normal distribution for 12 length sequence.

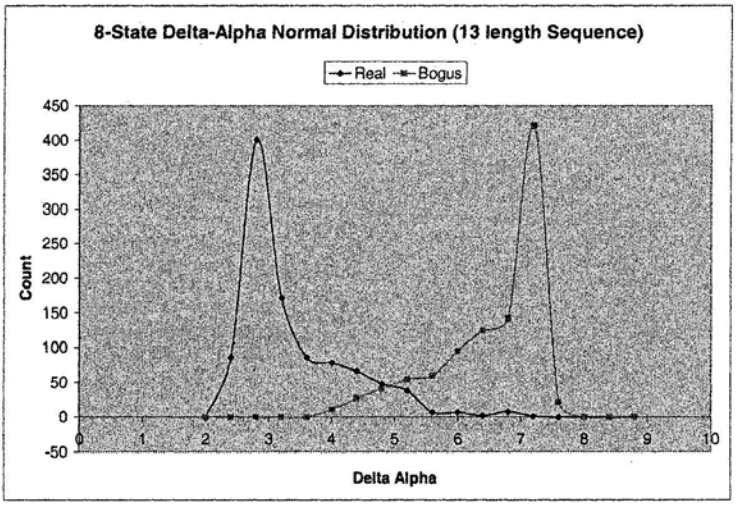


*Figure 2.* $\Delta\alpha$ normal distribution for 13 length sequence.

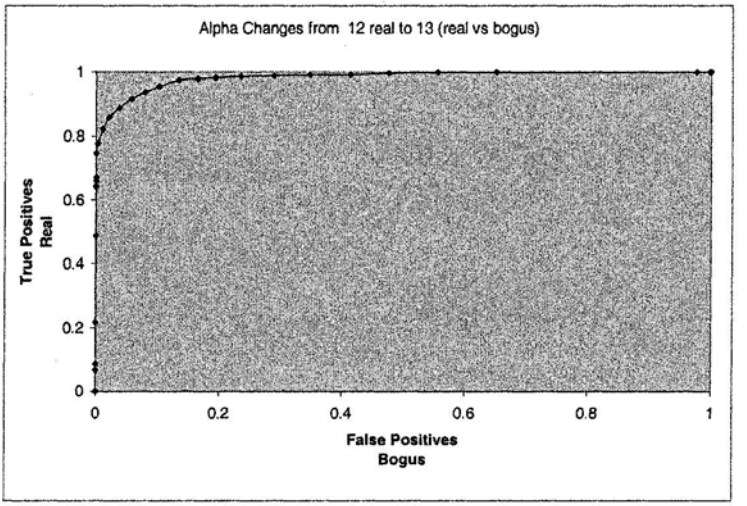


*Figure 3.* ROC curve for the true positive and false positive.

3 The slope of the tangent line at a cut point gives the likelihood ratio (LR) for that value of the test.

Accuracy is measured by the area under the ROC curve. An area of 1 represents a perfect test; an area of .5 represents a worthless test. In summary, likelihood ratios can be used to compute posttest probability of the corrupt sequence. They can more easily be applied to a series of diagnostic tests, their values convey intuitive meaning and the likelihood ratio form of Bayes theorem is easier to remember.

In our tests, we computed the true positive rate and false positive rate for the $\Delta\alpha$ values from the trained HMM for the test set of corrupted and good sequences for various cut points and plotted an ROC curve. The shape of the curve is excellent, confirming that our tests are capable of detecting the corrupted sequences.

The threshold value of the $\Delta\alpha$ at which the sequence is labelled anomalous is determined by the users need for accuracy in marking true positives and false positives. For example,using the ROC curve, we can find that a threshold of $\Delta\alpha = 5$ will provide a true positive result with a probability of .903. This is the predictive value of a positive test, i.e. the measure of how well our test rules the legitimate sequences. In this case, the added transaction is marked as a probable MDT since it's $\Delta\alpha$ value is above the threshold value of $\Delta\alpha$ =5.

## 4.3 Analysis of the Algorithm

The concept behind the $\Delta\alpha$ may be better understood through an analogy. Consider an automobile tire service plan official who receives a claim from a customer who says his tires are worn to the point of baldness after 58,000 miles. The official cannot say whether or not the tires have been abused based on that figure alone; he must first know the average life span of this particular model of tire. If this model of tire normally lasts 50,000 miles on average, this claim is likely legitimate. If, however, it usually lasts 100,000 miles, then it is likely that the customer did not take proper care of the tires.

The difference between the average life span and the particular life span here is akin to the $\Delta\alpha$ value. The $\alpha$ for the sequence in question before the inclusion of the most recent value gives us some indication of that sequence's "potential for acceptance" by the HMM. If the actual $\alpha$ value differs too greatly from that estimated potential, something is amiss.

Assuming that the $\alpha$ probability for the current sequence is stored in memory, the only computation involved in validating a proposed extension of the sequence is one evaluation of a sequence's acceptance probability. This runs in time quadratic in the number of states in the model. The number of potential elements in the sequence places an upper bound on the possible number of meaningful states in a left-to-right HMM, and so a $T$-length sequence is validated in $O(T^2)$ time. This algorithm can be further refined to reduce the complexity of each $\Delta\alpha$ evaluation using dynamic programming to memoize results of previous calculations. This can be significantly more efficient than pattern matching as employed by other techniques.

# 5. Conclusions and Future Directions

We used HMM in a novel manner to predict and validate the transactions in real time rather than to match patterns. Data reduction in general, and clustering specifically, allows this technique to be deployed for databases ordinarily too large to be analyzed in real-time. Our model was very successful in detecting the MDT from valid new transactions. Moreover, the technique provides the administrator the freedom to select a degree of acceptable risk.

The model was improved by combining data modeling and time series analysis technique with HMM to validate or detect an MDT. The model was also refined to work with variable-length sequences and the relationships between the varying length sequences. The proposed algorithm is not limited to generating a single most likely future occupancy.

In future the model can be further extended to adapt with unseen symbols in the sequences, and to improve the performance. Also, methods will be explored to allow for efficient model adaptiveness to occur in real time.

## References

[1] W. Aref, P. Vallabhaneni and D. Barbara, On training hidden Markov models for recognizing handwritten text, *Proceedings of the Fourth International Workshop on the Frontiers of Handwriting Recognition*, 1994.

[2] G. Box, G. Jenkins and G. Reinsel, *Time Series Analysis: Forecasting and Control*, Prentice Hall, 1994.

[3] G. Das, K. Lin, H. Mannila, G. Renganathan and P. Smyth, Rule discovery from time series, *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 1998.

[4] B. Davidson and H. Hirsh, Predicting sequence of user actions, *Proceedings of the AAAI/ICML Workshop on Predicting the Future: AI Approaches to Time-Series Analysis*, pp. 5-12, 1998.

[5] D. DeCoste, Mining multivariate time-series sensor data to discover behavior envelopes, *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 151-154, 1997.

[6] T. Fawcett and F. Provost, Adaptive fraud detection, *Data Mining and Knowledge Discovery*, pp. 177-181, 1997.

[7] T. Fawcett and F. Provost, Activity monitoring: Noticing interesting changes in behavior, *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 53-62, 1999.

[8] H. Jagadish, J. Madar and R. Ng, Semantic compression and pattern extraction with fascicles, *Proceedings of the Twenty-Fifth VLDB Conference*, pp. 186-198, 1999.

[9] S. Jajodia, P. Amman and C. McCollum, Surviving information warfare attacks, *IEEE Computer*, pp. 57-63, 1999.

[10] A. Kokkinaki, On atypical database transactions: Identification of probable frauds using machine learning for user profiling, *Proceedings of the Knowledge and Data Engineering Exchange Workshop*, pp. 107-113, 1997.

[11] T. Lane, Hidden Markov models for human/computer interface modeling, *Proceedings of the IJCAI-99 Workshop on Learning about Users*, pp. 35-44, 1999.

[12] J. Lee, R. McCartney and E. Santos, Learning and predicting user behavior for particular resource use, *Proceedings of the Fourteenth International FLAIRS Conference*, pp. 177-181, 2001.

[13] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE*, pp. 257-285, 1989.

[14] P. Smyth, Hidden Markov monitoring for fault detection in dynamic systems, *Pattern Recognition*, pp. 149-164, 1994.

[15] P. Smyth, Markov monitoring in unkown states, *IEEE Journal on Selected Areas in Communications*, pp. 1600-1612, 1994.

[16] C. Warrender, S. Forrest and B. Pearlmutter, Detecting intrusions using system calls: Alternative data models, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 133-145, 1999.

[17] B. Yi, N. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos and A. Biliris, Online data mining for co-evolving time sequence, *Proceedings of the IEEE International Conference on Data Engineering*, pp. 13-22, 2000.