Hidden Markov Models for Human/Computer Interface Modeling

Terran Lane
School of Electrical and Computer Engineering and
CERIAS
Purdue University, West Lafayette, IN 47907-1287
email: terran@ecn.purdue.edu

Abstract

Automated modeling of human behaviors is useful in the computer security domain of anomaly detection. In the user modeling facet of the anomaly detection domain, the task is to develop a model or *profile* of the normal working state of a computer system user and to detect anomalous conditions as deviations from expected behavior patterns. In this paper, we examine the use of hidden Markov models (HMMs) as user profiles for the anomaly detection task. We formulate a user identity classification system based on the posterior likelihood of the model parameters and present an approximation that allows this quantity to be quickly estimated to a high degree of accuracy for subsequences of the total sequence of observed data. We give an empirical analysis of the HMM anomaly detection sensor. We examine performance across a range of model sizes (i.e. number of hidden states). We demonstrate that, for most of our user population, a single-state model is inferior to the multi-state models, and that, within multi-state models, those with more states tend to model the profiled user more effectively but imposters less effectively than do smaller models. These observations are consistent with the interpretation that larger models are necessary to capture high degrees of user behavioral complexity. We describe extensions of these techniques to other tasks and domains.

1 Introduction

Automated modeling of human behaviors is useful in the computer security domain of anomaly detection [Anderson, 1980; Denning, 1987]. In the user modeling facet of the anomaly detection domain, the task is to develop a model or *profile* of the normal working state of a computer system user and to detect anomalous conditions as deviations from expected behavior patterns. A subset of hostile activities can then be detected through their anomalous behaviors. For example, recursively searching system directory hierarchies by hand or browsing through another user's files are unusual behaviors for many users and the presence of such activities may be indicative of an intruder who has penetrated the account. Alternatively, the *manner* in which similar tasks are carried out may be a critical indicator. For example, the sudden presence of complex tools such as `awk`, `perl`, or `dd` in the data of a user accustomed to making repetitive file modifications by hand may be a tip-off that a more system-literate user is employing the account. The flood of data generated by a user through sources such as command line and GUI events requires automated modeling and detection to uncover such events. We take a "personal assistant" view of this domain, in which the task of the anomaly detection sensor is to augment the security of a private individual's computer system or account by monitoring usage activity for "suspicious" incidents that do not conform to known behavior patterns of the account owner (denoted the *valid* or *profiled* user). Under this view, behavioral data are assumed to be private and available only to the valid user's assistant. Thus, training data are single class — representing only the behaviors of the profiled user.

We present an anomaly detection sensor that employs hidden Markov models (HMMs) as user models. Although the focus of this paper is on the application of HMMs as user models for a security domain, their potential as user models is more general. In Section 5, we discuss other possible applications of the types of models developed here.

2 Hidden Markov Models as User Behavioral Models

In this section we describe a framework for employing HMMs as user behavioral models for anomaly detection. We discuss different formulations of HMMs as sequence-data labelers, and describe the classification strategy we have adopted.

2.1 Notation

We employ a variant of Rabiner's HMM notation, [Rabiner, 1989], in which q_t and O_t are variables denoting

the HMM state and observed output at time t, respectively, $\bar{q}$ and $\bar{O}$ denote the complete sequence of states and outputs for the whole period of observation, π_i is the prior probability of state i, a_{ij} is the probability of transitioning from state i to state j, and $b_i(o)$ is the probability of generating output symbol v_o when the HMM is in state i. The matrix forms of the probabilities are Π, A, and B, respectively, and the set of all HMM parameters, $\{\Pi, \mathsf{A}, \mathsf{B}\}$ is θ. The number of hidden states in the model is K and the size of the alphabet of observable symbols is $|\Sigma|$. While the particular state that the model is in at time t is denoted q_t, the states themselves are labeled $S_1, S_2 \ldots S_K$. Similarly, the output symbols are labeled $v_1, v_2 \ldots v_{|\Sigma|}$ and the particular symbol observed at time t is O_t.

2.2 Domain Definition

The anomaly detection task can be regarded as a binary classification problem of self/non-self identification, [Forrest *et al.*, 1996; Lane and Brodley, 1997; 1998]. The problem is to label the incidences of anomalous behavior (for this work, behaviors originating with a party other than the profiled user) within a temporal stream of observations. Because the anomalous behaviors can, in principle, occur at any point in time, each observation must be assigned a label.

Because we cannot guarantee coverage of the space of anomalous activities, and for privacy reasons, we assume availability of data only from the profiled user. Our data are UNIX shell history traces and are described in detail in Section 3.2. Individual alphabet symbols are whitespace-separated "words" (or *tokens*) and the total HMM alphabet is the set of unique symbols occurring in all available user data.

2.3 HMMs as Sequence Data Classifiers

The task of employing hidden Markov models as temporal classification systems can be framed in at least three different manners. One popular method for multiclass problems is to identify the class labels with the hidden states of a single model. The state sequence inferred from observed data via the Viterbi algorithm, [Rabiner, 1989], then constitutes the classification of the temporal sequence data. Such an approach has been employed in, for example, speech recognition [Rabiner and Juang, 1993], positional tracking and prediction in user modeling [Orwant, 1995], and fault monitoring [Smyth, 1994a; 1994b]. Smyth describes this approach as *discriminative*, viewing the classification problem as one of estimating the probability of class labels given the data and model parameters, $p(\bar{q}|\bar{O}, \theta)$. He notes that this approach makes the assumptions that the class labels (states) are mutually exclusive and exhaustive. While the first condition certainly holds for the anomaly detection domain — any given input token can be generated by only a single user — the latter poses a considerable difficulty. In the anomaly detection domain we clearly have examples of the valid user's behavioral characteristics, but we lack examples of the behaviors of hostile users or intruders.[1] Even given examples of hostile behaviors, however, the problem of demonstrating that our training set is exhaustive may be difficult at best.

In the fault detection work, Smyth addresses the question of unobserved classes by adding an extra, "catch-all", state to the model and augmenting the discriminative model with a *generative* model. A generative model views the HMM as a data generator and estimates observation likelihoods, $p(\bar{O}|\theta)$ via the forward step of the forward-backward algorithm, [Rabiner, 1989]. Class probabilities can be derived from instantaneous observation probabilities, $p(O_t|q_i)$, via Bayes's rule. The hybrid of discriminative and generative approaches allows estimation of class probabilities for an auxiliary state modeling unobserved data. The combination of the two classes of models involves prior distribution assumptions about the likelihood of the data under the unknown class.

In this work, we take a different approach to the classification problem. Similar to the generative approach, we employ estimations of data probabilities via the forward-backward algorithm, but rather than associating class labels with model states, we associate class labels with individual *models*. Model probabilities can be evaluated from posterior observation probabilities via Bayes' rule:

$$p(\theta|\bar{O}) = \frac{p(\bar{O}|\theta)p(\theta)}{p(\bar{O})} \quad ,$$

where $p(\bar{O})$ is a normalizing factor that is identical for all classes. The model prior probability, $p(\theta)$, can be selected by domain knowledge, but we take it here to be a non-informative prior (i.e. a uniform probability distribution for a finite set of models). For an N class problem, an observational sequence is assigned the class label corresponding to the maximum likelihood model,

$$\text{class}(\bar{O}) = \underset{i \in 1 \ldots N}{\text{argmax}}\{p(\theta_i|\bar{O})\} \quad .$$

Effectively, we are assessing the likelihood that each model generated the sequence in question and selecting the model with the highest likelihood. This framework allows us to assign only a single label to an entire observational sequence, but gives us the freedom to assign "unknown" class labels. Any sequence judged insufficiently likely with respect to all known models can be labeled "unknown". Similar, "model-class", approaches have been widely applied in the speech recognition community, [Rabiner, 1989]. Orwant used related a framework to determine a user's current behavioral state (e.g. "idle", "writing", or "hacking"), [Orwant, 1995], but employed manually constructed models for each class and interconnected the class models into a "meta-HMM"

[1] Such data has proved to be difficult to come by. Examples (usually simulated) of machine-level attack logs (such as network packet logs or system call traces) are available, but traces of *real attacks* at the *human command* level are considerably rarer. A recent call for examples of such data by the CERIAS security research center has, to date, yielded no instances of such data.

from which classes were predicted via the Viterbi algorithm.

The choice of K is important, as it effects the potential descriptiveness of the HMM. In the discriminative and generative forms of HMM classification, the domain provides us with an appropriate value of K (the number of classes present in the data, and possibly one or more "unknown" states). In the model-class framework, however, the classes are not directly associated with model states so we must seek either domain-specific knowledge to help choose K (e.g. some estimate of the natural number of distinct behavioral classes present in the data) or employ an empirical search. We examine the latter method in the experimental section of this paper.

2.4 Sequence Labeling for the Anomaly Detection Domain

Under the model-class framework outlined above, we construct a *single* HMM, θ_v, to model the observed behavioral patterns of the valid user. The likelihoods of incoming data sequences are evaluated with respect to θ_v and those judged insufficiently likely via a threshold test are labeled as anomalous. The value of this "minimum acceptable likelihood" is denoted $t_{\min}$. A feature of the anomaly detection domain is the threat of "replay attacks"[2]. To avoid such attacks, we introduce an upper threshold, $t_{\max}$, which is used to flag data which are too similar to historical behaviors. The thresholds, $t_{\min}$ and $t_{\max}$ are chosen from the upper and lower $r/2$ quantiles of the non-parametric distribution of observation likelihoods on an independent, "parameter-selection", subset of the training data. The parameter r corresponds to an "acceptable" false-alarm rate[3] and its selection is a site-specific issue related to security policy.

2.5 Sequence Alignment

As noted above, the model-class framework assigns class labels only to entire sequences, yet we wish to be able to label arbitrary subsequences of the observed data stream. We can, of course, run the forward-backward likelihood estimation algorithm between every possible pair of subsequence start, s, and termination, t, time steps. This turns out to be computationally expensive, as the complexity of the F-B algorithm is $O(K^2 l)$ for a time sequence of length l. Merely to consider all fixed-length subsequences ($t - s = l$ for some fixed l for all t) within a total data sequence of length T requires $O(K^2 l(T - l))$ time. This becomes prohibitive for the subsequence lengths of interest in this domain ($l > 50$).

[2]A replay attack is one in which an attacker monitors a system and records information such as user commands. These commands can then later be "replayed" back to the system literally (or with the inclusion of a very few hostile actions). Because the vast majority of the data was, in fact, originally generated by the valid user, it will appear perfectly normal to the detection sensors unless some check is made for occurrences which are *too* similar to past behavior.

[3]Rate of incorrectly identifying the valid user as anomalous.

Instead we employ the approximation algorithm obtained by considering the endpoint state transitions (those at time steps s and t) to be statistically uncoupled from their adjacent states (those at time steps $s-1$ and $t-1$). That is,

$$p(O_s, O_{s+1} \dots O_t) \approx \frac{p(O_1 \dots O_T)}{p(O_1 \dots O_{s-1}) p(O_{t+1} \dots O_T)}$$

for $1 < s < t < T$. Because of the exponential decay of state influence in the Markov formulation, this approximation is reasonably good for large l. For example, a comparison of the approximated sequence log-likelihood to the exact value for one of our tested users at $l = 100$ (the value used in our empirical investigations, Sections 3 and 4) revealed that the approximated value had a mean deviation of only 0.8% and a median deviation of only 0.46% from the true value (indicating that the deviations are skewed towards 0). Thus, this approximation allows us to consider all fixed-length subsequences from a global temporal sequence of length T in time $O(K^2 T + (T - l))$, with a marginal loss in precision.

2.6 Alternate Approaches to Sequence Learning for User Modeling

Many traditional approaches to learning from temporal sequence data are not applicable to user modeling, where the base data consists of discrete, unordered (i.e. nominal-valued) elements such as command strings. For time series of *numeric* values, techniques such as spectral analysis [Oppenheim and Schafer, 1989], principle component analysis [Fukunaga, 1990], linear regression [Casella and Berger, 1990], linear predictive coding [Rabiner and Juang, 1993], nearest neighbor matching, and neural networks [Chenoweth and Obradovic, 1996] have proven fruitful. Such techniques typically employ a Euclidean distance or a related distance measure defined for real-valued vectors.

There are a number of learning algorithms that are amenable to learning on spaces with nominal-valued attributes, but they typically employ a feature-vector representation that may not be well suited to temporal data. For example, decision trees [Quinlan, 1993] are effective at discovering decision boundaries on discrete spaces. The bias used to search for such structures generally employs a greedy search that examines each feature independent of all others. Such a bias ignores the natural order relations present in temporal data (such as causality or correlation chains).

One method of circumventing this difficulty is to convert the data to an atemporal representation in which the causal structures are represented explicitly. Norton, (1994), and Salzberg, (1995), each independently used such a technique for the domain of learning to recognize coding regions in DNA fragments. DNA coding, while not temporal, does exhibit interrelations between positions that are difficult for conventional learning systems to acquire directly. The features extracted from the DNA sequences were selected by domain experts,

and thus cannot be generalized to other sequential domains. Although such an approach could be applied to the anomaly detection domain, it would require considerable effort on the part of a domain expert, and the developed features would apply only to that data source. We are interested in developing techniques that can be applied across different data sources and tasks.

There also exist learning methods explicitly developed to model sequence data. Algorithms for discovering temporal rule relations, for example, have been examined by Srikant and Agrawal (1996). Methods for learning the structure of deterministic finite-state automata have been widely studied [Angulin, 1987; Rivest and Schapire, 1989]. DFA's, however, are not well suited to modeling highly noisy domains such as human-generated computer interface data. If the data can be observed below the shell level, then many syntactic and semantic errors will have been removed and the data will be cleaner. Yoshida and Motoda employ I/O relations at this level to develop finite-state graph models of user behaviors (1996). The simplest extension of DFA models to noisy domains are Markov chain models, [Davison and Hirsh, 1998], which allow stochastic state transitions. These models have the advantage that, unlike HMMs, the Maximum-Likelihood estimate for transition probabilities has a closed form. Markov chain models typically emit symbols deterministically (each state or arc emitting only a single symbol), requiring a state for each symbol of the alphabet, or $|\Sigma|^2$ total transition probabilities to be learned for an alphabet of size $|\Sigma|$. When the alphabet is large (in our empirical analyses, we have observed alphabets of over 2500 unique symbols), the dimensionality of the parameter space is high and the amount of training data required to accurately estimate low probability transitions is very large. Finally, deterministic output Markov models with unique states (i.e. each symbol is emitted by only one state) can only represent a single context for any given symbol. In the anomaly detection domain symbols can have multiple contexts. The command `vi`, for example, can be employed for editing both source code and conference papers.

3 Empirical Analysis

Here we present the structure of our experimental evaluation of HMMs as user models for the anomaly detection domain. We examine performance measures for this task, discussing an alternative to the classic performance accuracy. We describe the sources and formatting of the data employed in our evaluations, and finally the structure of the experimental procedure.

3.1 Performance Criteria

We employ two methods for evaluating the performance of anomaly detection systems. In addition to the traditional *accuracy* measurements, we argue that the *mean time to generation of an alarm* is a useful quantity to consider.

The goal in the anomaly detection task is to identify potentially malicious occurrences while falsely flagging innocuous actions as rarely as possible. We shall denote the rate of incorrectly flagging normal behaviors as the *false alarm* rate and the rate of failing to identify abnormal or malicious behaviors as the *false acceptance* rate. Under the null hypothesis that all behavior is normal, these correspond to Type I and Type II errors, respectively. The converse accuracy rates are referred to as the true accept (ability to correctly accept the profiled user as normal) rate and the true detect (ability to correctly detect an anomalous user) rate. For the detector to be practical, it is important that the false alarm rate be low. Users and security officers will quickly learn to ignore the "security system that cried wolf," if it flags innocuous behavior too often.

Detection accuracy does not, however, reveal the complete story. A second issue of importance is *time to alarm* (TTA), which is a measure of how quickly an anomalous or hostile situation can be detected. In the case of false alarms, the time to alarm represents the expected time until a false alarm occurs. Thus, we wish the time to alarm to be short for hostile users so that they can be dealt with quickly and before doing much harm, but long for the valid user so that normal work is interrupted by false alarms as seldom as possible. To this end, we measure the mean run-length of "normal" classifications. Time is measured in token counts rather than wall clock time because the number of tokens emitted is more closely correlated with the activity of the user than is physical duration of the shell session.

3.2 Data

Because non-simulated human-level attack data has proven difficult to obtain, we have profiled our techniques on the user-differentiation problem. In this formulation, data are gathered from valid system users under normal working conditions and a user profile is constructed for each. The performance of the anomaly detection sensor is evaluated with respect to its ability to correctly recognize the profiled user and discriminate the other users as anomalous. This framework simulates only a subset of the possible misuse scenarios — that of a naive intruder gaining access to an unauthorized account — but it allows us to evaluate the approach.

We gathered command traces from eight UNIX users via the `tcsh` history mechanism over the course of more than two years. The command traces were parsed with a recognizer for the `tcsh` command language to convert them into a format suitable for scanning with the HMM classification sensor and to do feature selection. Each whitespace-delimited "word" in the input stream is considered to be a separate symbol, or token. The feature selection step removes filenames, replacing them with the count of the number of file names occurring in the command line. Removal of filenames reduces the alphabet from over 35,000 unique tokens to slightly more than 2,500 unique tokens and dramatically improves recognition accuracy in empirical tests.

3.3 Experiment Structure

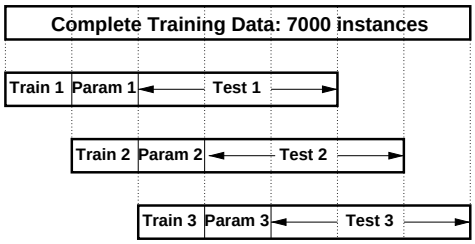


Figure 1: Division of training data into train, parameter selection, and test data sets.

Because user behaviors change over time, the effective lifetime of a *static* user profile, as is employed in the work described here, is limited. Thus, we have constructed experiments to evaluate the detector's performance over a limited range of future activities. The separation of the 7,000 token training data into three groups (or *folds*) of train, parameter selection, and test data is shown in Figure 1. The initial 1,000 tokens of each user's data were taken as training (profile construction) data, the following 1,000 tokens were taken for parameter selection data (used to set the decision thresholds $t_{\max}$ and $t_{\min}$), and 3,000 following tokens were taken to test performance for that profile. To guard against isolated data anomalies,[4] three folds of train, parameter selection, and test data were produced for each user. Within each fold, five random restarts were run to reduce the chance of the Baum-Welch training algorithm locating a spuriously poor local maximum ML parameter estimate. All tests were repeated for each fold and restart, and results were averaged across restarts.

From each test set, a profile HMM was constructed with $l = 100$ (the window length for sequence alignment, Section 2.5). To examine the impact of K on sensor performance, we constructed models with $K \in \{1, 2, 15, 30, 50\}$. The resulting profile was tested against the corresponding test set for each user (a total of 8^2 test pairings). A "self" test pairing — testing the profiled user's data against his or her own profile — allows us to examine false alarm rates while a "non-self" pairing allows us to examine false accept rates.

The acceptable false alarm rate, r, determines how the classification thresholds, $t_{\max}$ and $t_{\min}$, are set and has a substantial impact on the tradeoff between false alarm and false accept errors. Because the notion of "acceptable" false alarm rate is a site-dependent parameter, we wish to characterize the performance of the system across a spectrum of rates. We took $r \in \{0.5, 1, 2, 5, 10\}\%$ which yields a performance curve for each profile/test set pair. This curve, which expresses the tradeoff between false alarm and false accept errors with respect to r, is known as a Receiver Operating Characteristic (ROC) curve, [Provost and Fawcett, 1998]. An ROC curve allows the user to evaluate the performance of a system under different operating conditions or to select the optimal operating point for a given cost tradeoff in classification errors.

[4] E.g. we have found that our users tend to experience large behavioral changes at the beginning of academic semesters. The batch mode detection system presented here is highly sensitive to such changes.

4 Experimental Results

In this section we present the results of our empirical evaluations of HMMs as user models. Our first experiment explores the performance of 50 state ($K = 50$) models. We compare the performance values of these models to those with other values of K in Section 4.2.

4.1 Base System Performance

Figure 2 displays an example of accuracy, **(a)**, and time-to-alarm (TTA), **(b)** results for one test fold of a single profile (that of USER0). Each column in these plots displays the performance results for a single test set when tested against the profile. For accuracy results, when the test set originates with the profiled user (i.e. USER0 tested against Profile 0), the results indicate the ability to correctly identify the valid user (true accept rate). This condition is denoted with an "o" symbol on the plot. When the test set originates with a different user (e.g. USER3 tested against Profile 0), the results indicate the ability to correctly flag an anomalous condition (true detect rate). This condition is denoted with a "+" symbol on the plot. For both classes of tests in Figure 2, accuracy is increasing in the positive direction on the Y axis. The spectrum of results in each column is generated by testing at different values of r, the acceptable false accept rate, as described in Sections 2.4 and 3.3. Because r encodes the size of the acceptance region, it yields a tradeoff in detect versus accept accuracies. The smallest value of r tested ($r = 0.5\%$) yields the widest acceptance region and corresponds to the highest (most accurate) point on the true accept column (USER0). But because the acceptance region is wide, more anomalous points fall into it and are accepted falsely. Thus, this value of r corresponds to the *lowest* accuracy in each of the true detect columns (USER[1–7]).

Time-to-alarm results are displayed analogously but are not limited to a 0–1 range. Note that the vertical (time) axis is logarithmic, and that the times-to-alarm for the profiled user are nearly an order of magnitude larger than those for the opponents. This is the desired result because we wish times to be long for the profiled user so that false alarms are generated rarely, but short for the opponent users so that a hostile imposter can be detected quickly.

USER0 was chosen for display here to highlight a number of features of the HMM user profiling sensor. First is that accuracy is highly sensitive to the particular opponent. For example, testing with respect to USER5 yields far different detection accuracies than does testing on USER7. Second, although the acceptable false alarm

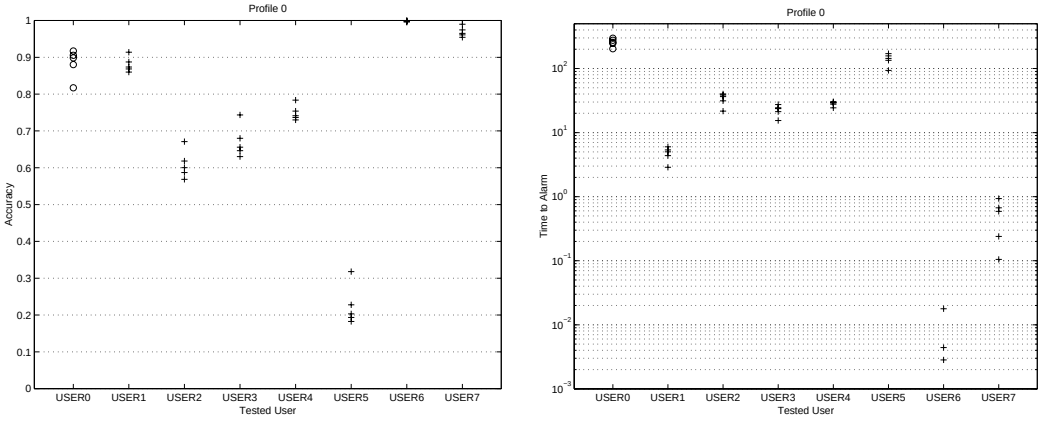


Figure 2: Accuracies, **(a)**, and mean times-to-alarm, **(b)**, for an HMM model ($K = 50$) of USER0's behaviors.

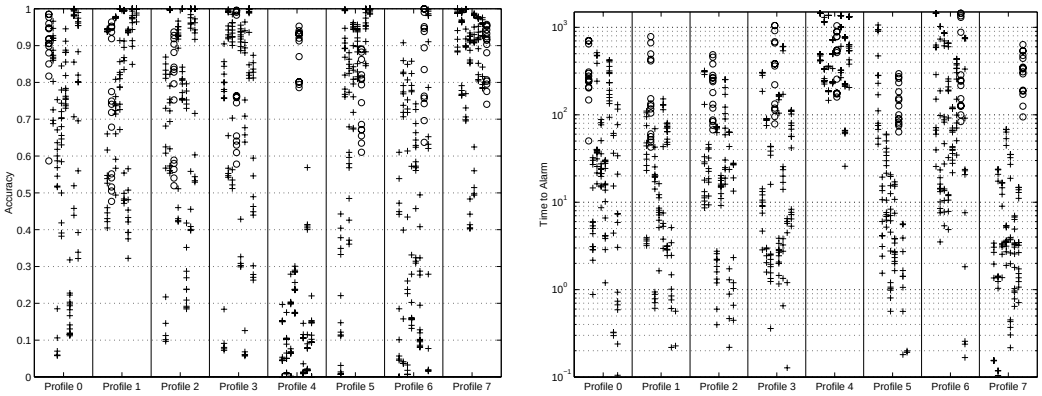
rate parameter, r, was tested across the range 0.5%–10%, all of the observed false alarm rates are greater than this (8.3%–18.3%). This is a result of the training and parameterization data failing to fully reflect the behavioral distribution present in the testing data. Because the user has changed behaviors or tasks over the interval between the generation of training and testing data, the profile does not include all of the behaviors present in the test data. This phenomenon is actually exacerbated by the batch-mode experimental setup used here. We have investigated online techniques for this domain which employ an instance based learning technique (IBL), [Lane and Brodley, 1998], and have found that they do perform better than the corresponding batch-mode IBL sensors. In future work, we will be investigating techniques for online versions of the HMM user modeling sensor.

The complete set of results for all profiles and folds for the HMM user model with $K = 50$ is shown in Figure 3. These plots are intended not as a reference for individual accuracy or time-to-alarm (TTA) values, but to convey a sense of the general performance of the anomaly detection sensor under different operating conditions and to highlight some behavioral characteristics of the detection system. In these plots, each column displays the results for a single user's profile (the same data as are displayed for USER0 in Figure 2). Now, however, all three folds are given for each profile.

The primary point of interest in these plots is that true acceptance abilities (ability to correctly identify the profiled user as him or herself) is generally good as evidenced by high accuracies and long times to generation of false alarms (i.e. the "o" symbols are clustered toward the top of each Y axis). In addition, the true detection abilities (ability to correctly identify that an imposter is not the profiled user) are generally fair to good as evidenced by reasonable accuracies and short times to generation of true alarms. Note that mediocre true detection abilities may be acceptable because each intruder need be caught only once.

The obvious and notable exception to the general performance trends is USER4. While the sensor displays strong true accept abilities with respect to this user, it provides only poor true detection abilities. This is an example of the decision thresholds ($t_{\max}$ and $t_{\min}$, as described in Section 2.4) being set to artificially extreme values, resulting in a spuriously large acceptance region. Thus, the system has effectively decided that "everything is USER4", and no real differentiation is being done — it is simply accepting most behaviors as normal. Examination of USER4's training data reveals that this user appears to devote entire shell sessions to single tasks (such as the compile-debug cycle) which appear as rather repetitious and monotonous patterns. Because this user is working in the X-Windows environment, tasks can be assigned to single shell sessions, and those shell sessions may be long-lived (some were over 2,000 commands). Thus, the training data may display only one or two sessions and a very small number of behaviors, while the parameter selection data displays a different (but also small) set of behaviors. Because there may be little overlap between training and parameter selection data, the observed similarity-to-profile frequency distribution may be distorted and the selected decision thresholds would then be poorly chosen.

A converse behavior occurs with Profile 1 on fold 2 (the set of circles at the lowest end of Profile 1 in Figure 3). This profile displays relatively low true accept rates in comparison to other profiles and folds, but very high true detect rates (often 100%). This is an example of the user model deciding that "nothing is USER1" because the acceptance region has been set too narrowly. As with USER4, this arises because different behaviors are displayed in the training and testing data. In this case, the parameter selection data reflects the training data well, but the test data is different from both of them. As a result, the acceptance range is narrowly focused to high-similarity behaviors, but the behaviors encountered in the testing data have lower similarity.



(a) (b)

Figure 3: Results for all user profiles and folds. Each column now displays a single profile tested against all test sets (i.e. each column is the equivalent of Figure 2).

4.2 Number of Hidden States

An open question in the use of HMMs for modeling is the choice of K, the number of hidden states. When the states have a clear domain interpretation, as for example in fault monitoring, the value of K may be naturally dictated by the domain. When K is not so conveniently available, however, we can employ an empirical analysis to discover an appropriate value. To examine the impact of K on sensor performance, we constructed models with $K \in \{1, 2, 15, 30\}$ and tested them under the same conditions used for $K = 50$. The case $K = 1$ is a degenerate form of an HMM equivalent to frequency estimation of the alphabet symbols with all time steps of the sequence data considered to be statistically independent. Effectively, the data is considered to have been generated by a multinomial process with $|\Sigma|$ elements drawn according to the distribution B (the output symbol generation distribution). Because the $K = 1$ case has different qualitative behaviors than the other cases, we discuss it separately.

Results for the $K = 1$ case are displayed in Figure 4. These figures are comparative, plotting the results for the $K = 1$ mode on the vertical versus results for the $K = 50$ mode on the horizontal. The diagonal line is the iso-performance surface, and points falling above it indicate higher performance by the $K = 1$ sensor, while points falling to the right of it indicate higher performance by the $K = 50$ sensor.

The general result of Figure 4 is that the 50 state HMM has much stronger true detection accuracies and TTAs. And though the true accept points are scattered more uniformly across the iso-performance surface (61 of the 120 true accept accuracy measurements fall on the $K = 50$ side of the line), the $K = 50$ system appears to have a slight margin, at an average of 1% higher true accept accuracy[5] than that reported by the $K = 1$ system. At first appearance, these results, while slight, seem to indicate at least that the $K = 50$ sensor is performing *no worse* than is the $K = 1$ sensor in terms of true accept accuracy and better in terms of true detection. The situation becomes somewhat more confused, however, when mean time-to-alarm is considered. In this dimension, the $K = 50$ model has superior time to false alarm, at an average of 15.6 tokens longer than $K = 1$, but inferior time to true alarm at 36.9 tokens longer. It turns out that this is skewed by USER4 (note that the logarithmic range of the TTA data allows a single user to significantly skew a simple additive mean). While the $K = 1$ model also suffers from the "everything is USER4" syndrome, it does so to a much lesser degree than does the $K = 50$ model and, thus, appears to be far more effective at separating other users from USER4. When USER4 is removed from the sample, the differences between $K = 50$ and $K = 1$ in the TTA domain favor $K = 50$, for which the mean TTA is 14.6 tokens longer for false alarms and 14.4 tokens shorter for true alarms.

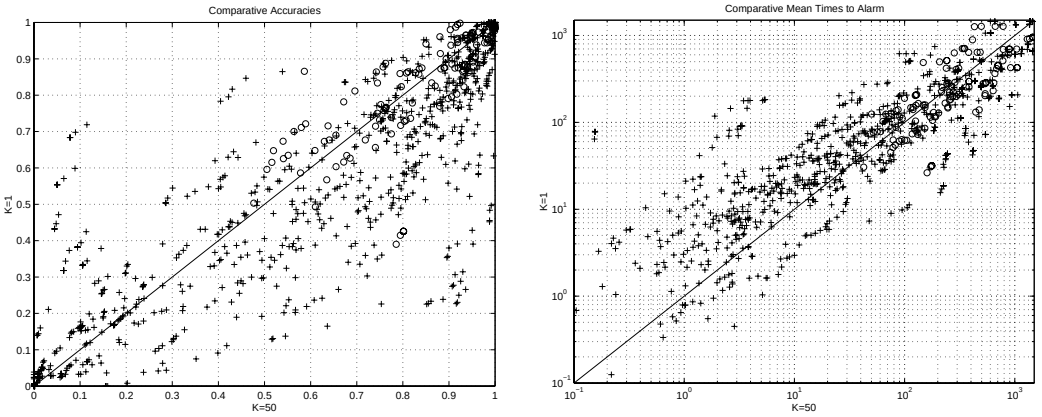
Results comparing the sensor system at $K = 2$, and 30 to $K = 50$ are given in Figure 5 (we omit $K = 15$, as it falls on the spectrum between $K = 2$ and $K = 30$ but is not otherwise unusual). Again, values for $K = 50$ are plotted on the horizontal while values for other settings of K appear on their respective vertical axes.

Figure 5 reflects a trend which is most dramatic in the $K = 2$ plots and which becomes less pronounced as K increases. The general result is that the $K = 50$ sensor has superior or equivalent true accept accuracies, but inferior true detect accuracies (albeit by a narrow margin, on average). The qualitative result in the TTA domain is similar, but the aggregate results are skewed by a few of the tested users — in this case USER0, USER1, and

[5] To measure the relative accuracy performance between two systems, we employ a mean of accuracy value differences. Thus, the difference in true detect rates between method 1 and method 2 is:

$$\frac{1}{N} \sum_{t \in \text{opponent\_test\_sets}} (\text{accuracy}_{method1}(t) - \text{accuracy}_{method2}(t))$$

where N is the number of opponent test sets.



(a) **(b)**

Figure 4: Comparisons of HMM user models with $K = 1$ (vertical axis) to $K = 50$ (horizontal axis). Accuracies appear in **(a)** and TTAs in **(b)**. The "o" symbols denote true accept rates and mean times to false alarms and the "+" symbols denote true detect rates and mean times to true alarms.

USER5. The structure of this trend is open to multiple interpretations. The "improved true accept coupled with degraded true detect performance" can be viewed as an indication that $K = 50$ is subject to the "everybody is the profiled user" difficulty with respect to smaller values of K. We can, however, take the converse interpretation that the models with smaller values of K are evidencing a "nobody is the profiled user" problem. Thus, the spectrum of values of K represents a spectrum of tradeoffs between user-oriented (at large values of K) and imposter-oriented (at small values of K). This observation is compatible with the interpretation that the models with larger K's are encoding a broader range of user behaviors than are the smaller models, although more investigation is required to verify this hypothesis. In general, the optimal number of hidden states for maximum discriminability is user dependent and seems to be related to the syntactic complexity displayed in the user's data. For example, USER4's data, which is extremely repetitive and employs only simple shell commands, is best modeled by a single state model while USER7's data, which displays some complex shell actions such as multi-stage pipelines, is best modeled by a 15 state HMM.

5 Extensions and Implications

The techniques presented here are not limited solely to the domain of anomaly detection nor to explicitly security-oriented tasks. A number of other possible uses could be realized with straightforward modifications to this framework.

User Identification The most obvious extension to this work is the capacity to identify one particular user from a set of known users solely through behavioral characteristics. This use, however, is also mostly of security interest, as methods such as password or physical tokens can identify users more quickly and accurately *when they can be trusted*. The online monitoring approach to identification serves mainly as a verification of and backup to primary identification techniques.

Group Identification A more visibly useful extension is to identify users as members of groups rather than as individuals. By constructing models of a group's exemplar behaviors, an individual can be automatically assigned to a group and inherit environmental customizations appropriate to that group's needs.

Behavioral Identification At a finer grain, a user's behaviors may be segmented by class (e.g. writing, play, coding, web surfing). Such an approach has been examined with manually constructed HMMs by Orwant, [Orwant, 1995]. By analyzing the substructure of the interconnections in an automatically generated HMM, behavioral classes might be automatically identified and associated with appropriate responses for a user interface.

Behavioral Prediction HMMs can be run not only as observational models but also as generative models. In such a framework, they could be used to predict a user's next actions and provide time-saving shortcuts (such as opening menus or initiating expensive computations early).

The observation that USER4, for example, displays qualitatively different behaviors than do other users (because USER4 is modeled more effectively by the single state model while the other users are modeled more effectively by the multi-state models) indicates that the HMM framework is capable of discerning some types of behavioral groupings. The results on choice of K are also consistent with the interpretation that users fall along a spectrum of behavioral complexities which can be identified by models of differing complexity. Under the privacy-oriented framework employed here, it is difficult to employ some of this knowledge because cross-

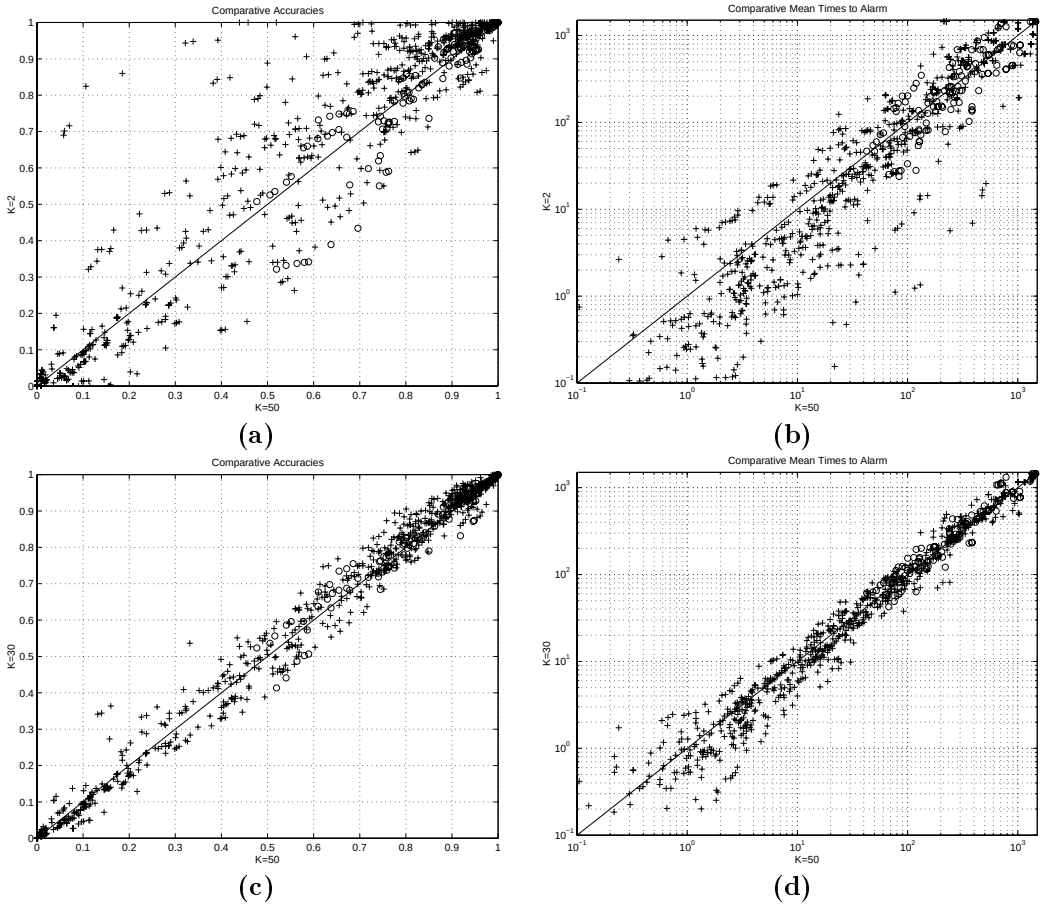


Figure 5: Comparisons of $K = 2$, **(a)** and **(b)** and $K = 30$, **(e)** and **(f)**, (on their respective vertical axes) to $K = 50$ (horizontal axis).

validation testing is impossible, but in a less constrained setting these types of distinctions could be extracted fairly easily and used to assist the user.

6 Conclusions and Future Work

We have demonstrated the use of hidden Markov models for user profiling in the domain of anomaly detection. The key results of the empirical investigation are:

- HMMs can be used to identify users by their command line behavioral patterns.
- These models suffer from two general classes of errors: overly permissive ("everybody is the profiled user") and overly restrictive ("nobody is the profiled user").
- The number of hidden states in the HMM represents a spectrum of tradeoffs between these two error classes. Larger models were found to be more effective at identifying the valid user, while smaller models were generally better at discerning impostors.
- The optimal number of hidden states is user-dependent and appears to reflect a measure of the syntactic complexity present in the command line data.

We found that single-state HMM models (effectively token frequency estimation models) display qualitatively different behaviors than do multi-state models. For most of the profiled users, the multi-state models were more effective than the single-state model. The exception was USER4, whose data consisted of long sessions each of which encoded a small number of tasks.

An open problem for this user profiling technique is the ability to select appropriate model parameters (such as K) from data or prior knowledge. In supervised learning domains, cross-validation search may be used to select appropriate parameter settings, but we must seek unsupervised techniques for this domain. The observation that optimal choice of K seems to be related to behavioral complexities presents a potential approach to this problem. It is possible that measures of data complexity, such as entropy, could be used to select appropriate model parameters.

Finally, the sensor employed here functions off-line. In other work, [Lane and Brodley, 1998], we have found that on-line extensions to an instance based user mod-

eling sensor displayed heightened performance. We are currently investigating extensions of the HMM anomaly detection sensor to on-line mode, and expect that similar performance improvements will be realized by this change.

Acknowledgments

Portions of this work were supported by contract MDA904-97-C-0176 from the Maryland Procurement Office, and by sponsors of the Center for Education and Research in Information Assurance and Security, Purdue University. We would like to thank Carla Brodley, Craig Codrington, and our reviewers for their helpful comments on this work. We would also like to thank our data donors and, especially, USER4 whose data forced us to examine this domain more closely than we might otherwise have done.

References

[Anderson, 1980] J. P. Anderson. Computer security threat monitoring and surveillance. Technical Report Technical Report, Washington, PA, 1980.

[Angulin, 1987] D. Angulin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[Casella and Berger, 1990] G. Casella and R. L. Berger. *Statistical Inference*. Brooks/Cole, Pacific Grove, CA, 1990.

[Chenoweth and Obradovic, 1996] T. Chenoweth and Z. Obradovic. A multi-component nonlinear prediction system for the S&P 500 index. *Neurocomputing*, 10(3):275–290, 1996.

[Davison and Hirsh, 1998] B. D. Davison and H. Hirsh. Predicting sequences of user actions. In *Proceedings of the AAAI-98/ICML-98 Joint Workshop on AI Approaches to Time-series Analysis*, pages 5–12, 1998.

[Denning, 1987] D. E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232, 1987.

[Forrest *et al.*, 1996] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. In *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy*, 1996.

[Fukunaga, 1990] K. Fukunaga. *Statistical Pattern Recognition (second edition)*. Academic Press, San Diego, CA, 1990.

[Lane and Brodley, 1997] T. Lane and C. E. Brodley. Sequence matching and learning in anomaly detection for computer security. In *Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 43–49, 1997.

[Lane and Brodley, 1998] T. Lane and C. E. Brodley. Approaches to online learning and concept drift for user identification in computer security. In *Fourth International Conference on Knowledge Discovery and Data Mining*, pages 259–263, 1998.

[Norton, 1994] S. W. Norton. Learning to recognize promoter sequences in E. coli by modelling uncertainty in the training data. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 657–663, Seattle, WA, 1994.

[Oppenheim and Schafer, 1989] A. Oppenheim and R. Schafer. *Discrete-Time Signal Processing*. Signal Processing. Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[Orwant, 1995] J. Orwant. Heterogeneous learning in the Doppelgänger user modeling system. *User Modeling and User-Adapted Interaction*, 4(2):107–130, 1995.

[Provost and Fawcett, 1998] F. Provost and T. Fawcett. Robust classification systems for imprecise environments. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Madison, WI, 1998. AAAI Press.

[Quinlan, 1993] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Rabiner and Juang, 1993] L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[Rabiner, 1989] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), February 1989.

[Rivest and Schapire, 1989] R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. In *Proceedings of the Twenty First Annual ACM Symposium on Theoretical Computing*, pages 411–420, 1989.

[Salzberg, 1995] S. Salzberg. Locating protein coding regions in human DNA using a decision tree algorithm. *Journal of Computational Biology*, 2(3):473–485, 1995.

[Smyth, 1994a] P. Smyth. Hidden Markov monitoring for fault detection in dynamic systems. *Pattern Recognition*, 27(1):149–164, 1994.

[Smyth, 1994b] P. Smyth. Markov monitoring with unknown states. *IEEE Journal on Selected Areas in Communications, special issue on intelligent signal processing for communications*, 12(9):1600–1612, 1994.

[Srikant and Agrawal, 1996] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements,. In *Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT)*, Avignon, France, 1996.

[Yoshida and Motoda, 1996] K. Yoshida and H. Motoda. Automated user modeling for intelligent interface. *International Journal of Human-Computer Interaction*, 8(3):237–258, 1996.