Analysis of Hidden Markov Model Learning Algorithms for the Detection and Prediction of Multi-Stage Network Attacks

Timothy Chadza^{a,b,*}, Konstantinos G. Kyriakopoulos^a, Sangarapillai Lambotharan^a

^a*Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, Loughborough, UK*

^b*Electrical Engineering Department, University of Malawi-Polytechnic, Blantyre, MW*

Abstract

Hidden Markov Models have been extensively used for determining computer systems under a Multi-Stage Network Attack (MSA), however, acquisition of optimal model training parameters remains a formidable challenge. This paper critically analyses the detection and prediction accuracy of a wide range of training and initialisation algorithms including the expectation-maximisation, spectral, Baum-Welch, differential evolution, K -means, and segmental K -means. The performance of these algorithms has been evaluated, both individually and in a hybrid approach, for detecting all the states and current state, and predicting the next state (NS), and the next observation (NO) of a given alert observation sequence. For generating this alert sequence, the Snort signature-based intrusion detection system was utilised, using either bespoke or default rules, to raise alerts while examining the DARPA 2000 MSA dataset. The investigation also sheds further light on alternative approaches for forecasting the possible NS and NO in an MSA campaign, as well as, the impact of window size on the prediction performance for all analysed techniques. The results and discussion emphasise on the appropriateness of various techniques for the prediction of NS and NO. Furthermore, NO prediction accuracy has indicated a performance increase of up to 44.95% in the proposed hybrid approaches.

Keywords: Hidden Markov model, Multi-stage attacks, Viterbi

List of Abbreviations

AS	All states
BW	Baum-Welch
CS	Current state
DARPA	Defense Advanced Research Projects Agency
DMZ	Demilitarised Zone
DDoS	Distributed Denial of Service
DE	Differential evolution
HMM	Hidden Markov model
IDS	Intrusion Detection System
IP	Internet Protocol
MSA	Multi-stage network attack
NO	Next observation
NS	Next state
SA	Spectral algorithm
SA-AHK	Spectral algorithm by Anandkumar, Hsu and Kakade
SA-HKZ	Spectral algorithm by Hsu, Kakade and Zhang
SKM	Segmental K -means
SnC	Segmentation and Clustering

1. Introduction

The explosion of cloud computing and Internet of Things devices has inevitably increased cyber threats. The proliferation of Internet of Things devices exposes security and privacy

threats as it has attracted spreading of malicious content by intruders [1]. Multi-Stage Network Attacks (MSAs), which are executed in a series of strategic steps, pose a formidable security risk to Intrusion Detection Systems (IDSs) and intrusion prevention systems [2]. This is because, individual stages of MSAs may appear as normal activity when considered in isolation [3]. Furthermore, there is a variation in time between individual attack stages, spanning from hours to months, hence concealed from IDSs.

Attack prediction and forecasting are crucial in cybersecurity as they provide key information for enhancing network security. Authors in [4] survey the four key use cases in cybersecurity, namely, attack projection, attack intent recognition, intrusion prediction, and network security situation forecasting. In practice these use cases are interchangeable since they share a common theoretical background. Nevertheless, attack projection focuses on the next step of an attacker [5]; intention recognition deals with the ultimate attackers motive [6]; intrusion prediction foresees the future cyberattacks [7]; and network security situation forecasting makes quantitative future prediction based on historical and current network security information [8].

Overall, a breakthrough in prediction and forecasting of cyber security is still anticipated, hence the need to revitalise mechanisms that can benefit this discipline [9]. The prediction of future events by investigating previous correlated events has culminated into improved detection and deterrence of internal threats [10]. Prediction and forecasting of MSAs can be implemented using various mechanisms including discrete models,

*Corresponding author

Email addresses: tchadza@poly.ac.mw (Timothy Chadza),
elkk@lboro.ac.uk (Konstantinos G. Kyriakopoulos),
s.lambotharan@lboro.ac.uk (Sangarapillai Lambotharan)

such as attack graphs [11], Bayesian networks [12], Markov models [13, 14] and game theory [15]. Contrary to the other discrete modelling techniques, Markov models perform better for unobservable states and transitions, thereby eliminating the need for complete information prior to attack prediction and detection [4]. Markov chains and Hidden Markov Models (HMM) are the two main examples of Markov models.

In the context of network attacks, the attack surface is modelled following the Markov property, which defines the future state as a function of just the current state. In practice, an HMM requires a dataset for training and the lack of rich datasets has always posed a formidable challenge for evaluating IDSs. For MSA, the DARPA 2000 [16] dataset is undoubtedly the most commonly used dataset [17]. This dataset has been used in this paper since the focus was primarily on analysing HMM learning algorithms.

Three fundamental problems of HMM that exist are: learning, decoding, and evaluation [18]. Firstly, learning involves the determination of HMM parameters, i.e. state transition and observation probability matrices, given an observation sequence. The parameters are adjusted to maximise the observation probability. Secondly, decoding focuses on determining the most likely sequence of states corresponding to an observation sequence given a specific HMM. Finally, evaluation obtains the probability that a given observation sequence has been generated by an HMM. Baum-Welch (BW) is the de-facto method deployed to obtain the HMM training parameters; however, it often falls into local optima solutions. Consequently, several alternative approaches have been proposed to enhance or replace this algorithm.

In this paper, Differential Evolution (DE), a genetic algorithm, and *K*-means were each leveraged together with using BW and Segmental *K*-means (SKM) algorithm, also called Viterbi training. Both BW and SKM algorithms require initialisation of the three HMM parameters (state transition, initial probability, and observation probability matrices). In addition to DE and *K*-means, Expectation-Maximisation (EM), and two Spectral Algorithms (SAs) are also deployed for HMM training. EM uses random initialisation by default; however, SA does not require any initialisation and uses singular value decomposition [19]. All the training and initialisation techniques were assessed with regards to determining the current and prior states of a given observation sequence, and in terms of predicting the forthcoming observation and the corresponding state.

To train HMMs, observation sequences were extracted from alerts obtained after feeding the DARPA 2000 dataset pcap files into the Snort [20] IDS. The dataset comprises a five-step Distributed Denial of Service (DDoS) MSA launched in two scenarios, with scenario two stealthier than scenario one. In this paper, two different datasets from the same scenario one, the first from inside the network and the second from the Demilitarised Zone (DMZ), were used for HMM training and evaluation, respectively. Snort IDS was opted for as it is a well maintained and a popular open source solution or generating intrusion alerts [21]. The contributions of this paper are:

- A comprehensive study and analysis of HMM state-of-the-art learning techniques as applied to network MSA detection and prediction based on IDS alerts. Evaluated techniques include the EM, two SA implementations, DE, *K*-means, BW and SKM. These techniques are both independently and jointly evaluated in a hybrid approach for detecting all prior attack states, current state, and predicting next attack observation and state.
- Analysis of the uniform, random, and count-based, initialisation techniques, the EM, SA, *K*-means and DE techniques. Furthermore, these techniques have been leveraged as initialisation methods to feed initial HMM parameters to BW and SKM algorithms.
- Assessment of the prediction of the next attack state and next observation, using two methodologies. The first method requires both the HMM state and emission probabilities, whilst the second approach incorporates either the state or the emission probabilities for predicting the next state or next observation, respectively. Furthermore, the next state and next observation prediction analysis includes performance with either one (most likely) or two levels (two most likely).
- Evaluation of the optimum alert observation window size and analysis of the effects of the window size on the HMM performance for all evaluated techniques.

The rest of the paper is organised as follows: Section 2 reviews closely related work on HMM parameter training and initialisation. HMM types, parameter definitions, and challenges are described in Section 3. In Section 4, the selected parameter estimation techniques are outlined. Training and evaluation data, including an illustration of customised Snort IDS rules used to obtain the observation sequence, HMM types, and experimental setup are provided in Section 5. Results and discussions are outlined in Section 6 and the paper concludes in Section 7.

2. Related Work

Several researchers have deployed HMM for cyber security threats detection using diverse techniques. The BW algorithm has been extensively described in literature and widely applied for training HMM as it is a special case of EM [22]. BW re-estimates HMM parameters given initial model parameters thereby improving an observation sequence probability of occurrence. Three typical initialisation techniques for BW are: random parameters [23], uniform parameters [24], and partial estimation based on training data. Examples of explicit techniques that use part of the training dataset to obtain HMM parameters are: modified *K*-means clustering [25], SKM [26], time-based and histogram-based [27], and segmentation and clustering [28].

Numerous techniques were proposed to address slow convergence and local optima issues of BW. A DE technique is proposed in [29, 30] based on original work by [31]. The technique

is evolutionary and designed to primarily solve global optimisation problems over continuous domains. It estimates the best initial parameters given a cost function. The performance of DE relies on the optimal selection of the scaling factor, crossover rate and population size. The DE parameter selection has been automated by authors in [32].

The Segmentation and Clustering (SnC) technique suggested by authors in [28] is used for initialisation of HMM parameters in industrial applications. Unlike BW algorithm which has a predefined topology, i.e the number of states, SnC technique estimates the the number of states prior to initial parameter determination by using segmentation and clustering. The HMM parameters are then computed by counting the occurrence of the observations and hidden states. The generated parameters are ultimately used to initialise BW. The SnC technique has been validated on both synthetic and real industrial data; however, like most clustering techniques, the state significance of training data is ignored as it computes optimal number of states. Therefore, this is not ideal when there is a predetermined number of states as in the case of MSA, i.e the five-step model of the DARPA 2000 dataset.

Authors in [33] suggested HMM parameters estimation with inequality constraints. They proposed a four-step procedure for HMM parameters estimation and re-estimation namely: initialisation, parameter estimation, parameter re-estimation when inequality constraint is not satisfied, and termination. The third step uses an active set-based Lagrange multiplier method and EM algorithm.

K -means algorithm has also been extensively used to train HMMs. A classical K -means clustering technique aims at partitioning the observation sequence into clusters where each observation is allocated to the cluster with the nearest mean. Modifications to this technique have been suggested by several researchers. To begin with, SKM has been applied for handwriting recognition [26]. In this case, K -means is used to initialise observation probabilities by an iterative process of dividing the observation sequence into clusters and selecting the one with a higher mean square error for further division. In contrast to BW, which simply updates all possible paths, the key advantage of SKM is that, only the most likely path is updated, which reduces the computation time.

Furthermore, authors in [25] proposed a modified K -means for initialising BW. Unlike in most classical K -mean implementations, the authors reduced a large number of states by performing a K -means algorithm on the states, rather than on the observation sequence. Initially, they assumed a classical set of models having a discrete state space of hidden variables and observation probabilities following a Gaussian distribution. A model with a very large number of states describing training sequences was then constructed. Finally, the maximum likelihood that an observation sequence is generated by an HMM, was computed. A higher likelihood value with the modified K -means algorithm was obtained than when using the classical BW initialisation techniques.

Authors in [27] proposed two methods for one-off initialisation of HMM for trading off between accuracy and training time. They used a time-based and histogram-based tech-

nique. The former partitions the training data using timestamps, whereas the latter considers feature proximity of the data.

Recently, spectral methods, which use method of moments, have been used for HMM parameter estimation. Two spectral learning algorithms have been proposed in [19] and [34]. The key advantages of this technique are less computational time and sample complexity. The implementation of the proposed algorithms, was attempted by authors in [35], who applied HMM using real-valued emissions.

It can be noted that there are numerous proposals for both training HMM and initialisation of training algorithms. Despite various techniques being proposed for HMM parameter estimation, deployment of such techniques for intrusion anomaly detection and prediction suffers in practical evaluation [36]. To address the need for deploying attestable training and initialisation techniques, the work in this paper consolidates the aforementioned techniques and evaluates their performance.

3. Hidden Markov Model

3.1. HMM Parameter Definitions

HMM is a two level stochastic process, where the first level represents the states of a modelled system, which are unobserved. The second level represents the observations/emissions obtained from the system, which in a network security application may be IDS alerts indicating the state of the system. The HMM is commonly represented as a 3-tuple (A, B, π) , where A is the state transition matrix, B is the observation probability matrix, and π is the initial probability vector.

Authors in [37, 24] outline a 6-tuple (Q, V, A, B, π, O) representation of the HMM parameters. The three additional parameters are a set of system states, S , a set of distinct observation symbols, V , and an observation sequence, O .

The following is a brief description of the six HMM parameters:

1. *Set of system states.* The finite set, S having N states represented as:

$$S = \{s_1, s_2, \dots, s_N\} \quad (1)$$

The state at time t is designated as q_t . Examples of these states are the five phases of DARPA 2000 dataset previously discussed e.g IP sweep, and Sadmin probe.

2. *Set of distinct observations.* The set, V , comprising M distinct number of observation symbols given by:

$$V = \{v_1, v_2, \dots, v_M\} \quad (2)$$

Examples of the observations are Snort IDS alerts such as ICMP ping, flood DDoS attempt and RPC Sadmin UDP ping.

3. *Observation sequence, O .* This has a variable length, T , and represents a series of observations whose elements belong to the set of distinct observation symbols V . The set is denoted as:

$$O = \{o_1, o_2, \dots, o_T\} \quad (3)$$

4. *State transition matrix, A*. This is an $N \times N$ matrix whose rows sum is unity i.e $\sum_j a_{ij} = 1, \forall i, j \in [1, N]$

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix} \quad (4)$$

Each element of A , a_{ij} denotes the probability of transitioning from state, i to j i.e. $\forall t \in [1, T], a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$.

5. *Observation probability matrix, B*. This is an $N \times M$ matrix, $\sum_k b_j(v_k) = 1, \forall j \in [1, N]$ and $k \in [1, M]$.

$$B = \begin{bmatrix} b_1(1) & b_1(2) & \dots & b_1(M) \\ b_2(1) & b_2(2) & \dots & b_2(M) \\ \dots & \dots & \dots & \dots \\ b_N(1) & b_N(2) & \dots & b_N(M) \end{bmatrix} \quad (5)$$

Each element of B , $b_j(v_k)$ denotes the probability of observing symbol, v_k while in state j i.e. $\forall t, b_j(v_k) = P(o_t = v_k | q_t = s_j)$.

6. *Initial state probability vector, π* . This is represented as a row matrix of introductory state probabilities at time, $t = 1$, i.e. $\pi_i = P(q_1 = s_i)$. Similarly to A and B , $\sum_i \pi_i = 1, \forall i \in [1, N]$.

$$\pi = [\pi_1, \pi_2, \dots, \pi_N] \quad (6)$$

3.2. HMM Types

There are generally two HMM types: ergodic and left-right. Authors in [22] define the ergodic HMM as a fully connected HMM where each state can traverse to any possible state as shown in Fig. 1 (a). This type is specifically not applied to MSAs since it is rarely anticipated that an attacker would compromise a target by regressing to previous state(s), rather than proceeding towards the goal. The left-right HMM type, shown in Fig. 1 (b), does not permit transitions to previous states, which is typical in MSAs. A challenge with this type is that $a_{ij} = 0, \forall j < i$, and zero values are a constraint to BW and Viterbi decoding implementations since the state transition is expected to be $0 < a_{ij} \leq 1$. This challenge can be addressed by assigning very small values instead of zeros and subsequently scaling each row so that it sums up to unity.

Figure 1: State diagrams for HMM types: (a) ergodic, and (b) left-right.

3.3. Three Fundamental Problems for HMMs

Training (learning), decoding, and evaluation are the three fundamental HMM problems, as originally described in [18]. The former is the most challenging problem as it plays a pivotal

role in defining the HMM. The following is a brief description of these problems and how they are commonly tackled:

1. *Training (Learning)*: This adjusts all HMM parameters, λ , to maximise the observation probability, $P(O|\lambda)$. This problem is the focus of this paper and the proposed training algorithms are briefly described in Section 4. The BW algorithm is most frequently used to alleviate this problem.
2. *Decoding*: The states of an HMM are non observable; however, the observations that are generated in a state are evident. The decoding problem attempts to find the most likely state path (Viterbi path) that can be transitioned given an observation sequence, O , and HMM parameters, λ . The Viterbi decoding is usually used to solve this problem.
3. *Evaluation*: Given an HMM, it is essential to determine the probability of an observed sequence being generated by that model. The forward and backward algorithms are normally used for this task.

4. HMM Parameter Estimation Techniques

4.1. Expectation-Maximisation Algorithm

The Expectation-Maximisation (EM) algorithm is often used for estimating unobserved latent variables in a model. The algorithm simplifies the complex challenges of analytically applying maximum likelihood estimation [38]. Authors in [39] present the EM's two step processes as: (1) Expectation step or E-step, which obtains the likelihood using the current parameters, and (2) Maximisation step or M-step which computes new parameters optimising the anticipated likelihood. Algorithm 1 demonstrates how the estimated state transition, $\hat{a}_{ij}$, and estimated observation probability matrices, $\hat{b}_j(k)$, are obtained using EM.

Algorithm 1 Expectation-Maximisation pseudocode

Input: Observation sequence O

Output: Re-estimated state transition matrix $\hat{a}_{ij}$,
Observation probability matrix $\hat{b}_j(v_k)$

- 1: Initialise number of observations M , number of states N , threshold Th
 - 2: Randomly initialise $a_{ij}, b_j(v_k)$ to obtain model, λ
 - 3: **repeat**
 - 4: *E-step*: given current parameters λ , and O , determine the Q function (i.e expectation)
 - 5: *M-step*: find $\hat{\lambda}$ with $\hat{a}_{ij}$ and $\hat{b}_j(v_k)$ that maximises the expected likelihood, $P(O|\lambda)$
 - 6: $\epsilon = \lambda - \hat{\lambda}$
 - 7: $\lambda = \hat{\lambda}$
 - 8: **until** $\epsilon \leq Th$
 - 9: **return** $\hat{a}_{ij}, \hat{b}_j(v_k)$
-

4.2. Baum-Welch Algorithm

The Baum-Welch (BW) algorithm is a recursive EM algorithm. Authors in [40] describe the algorithm as a forwards-backwards and iterative hill-climbing algorithm to re-estimate HMM parameters. It uses initialised HMM parameters which are then iteratively adjusted based on the likelihood of the parameters and observation sequence. The maximum number of iterations, G , and tolerance or threshold, Th , have to be pre-determined.

A detailed description of the BW algorithm is presented in [18]. Similar to EM, the BW follows general steps to initialise, update, and re-estimate parameters until convergence. To estimate the HMM parameters using BW, the following terms are useful:

1. *Forward variable, $\alpha_t(i)$* : This is computed using forward algorithm and is defined as the probability of seeing partial observations, $o_1, o_2, \dots, o_t$ at time t , while in state i . It is obtained as shown in Eq. (7) and Eq. (8), for initialisation and induction, respectively.

$$\alpha_1(i) = \pi_i b_j(o_1) \quad (7)$$

$$\alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^N \alpha_t(i) a_{ij} \quad (8)$$

2. *Backward variable, $\beta_t(i)$* : This is defined as the probability of the remaining observation sequence, $o_{t+1}, o_{t+2}, \dots, o_T$, while being in state i at time T , given the HMM parameters, λ [41]. It is computed as in Eq. (9) and Eq. (10), for initialisation and induction, respectively.

$$\beta_T(i) = 1 \quad (9)$$

$$\beta_t(i) = \sum_{j=1}^N \beta_{t+1}(j) b_j(o_{t+1}) a_{ij} \quad (10)$$

3. *State probabilities, $\gamma_t(i)$* : This is a temporary variable obtained from the forward and backward variables. It is defined as the probability of being in state i at time t , given the observed sequence, O , and HMM parameters, λ , and is defined as in Eq. (11).

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^N \alpha_t(j) \beta_t(j)} \quad (11)$$

4. *Optimal state sequence, $\xi_t(i, j)$* : This is defined as the probability that at time t the state is s_i , whereas at time $t + 1$, the state is s_j , given the observed sequence, O and HMM parameters, λ . Eq. (12) shows the corresponding computation:

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} \beta_{t+1}(j) b_j(o_{t+1})}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} \beta_{t+1}(j) b_j(o_{t+1})} \quad (12)$$

The following equations are then used to update the HMM

parameters [18, 42, 41]:

$$\hat{\pi}_i = \gamma_1(i), \forall i \in [1, N] \quad (13)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \forall i, j \in [1, N] \quad (14)$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1}^T \gamma_t(j), \text{ if } O_t = v_k}{\sum_{t=1}^T \gamma_t(j)}, \forall j \in [1, N], \forall k \in [1, M] \quad (15)$$

Firstly, the initial state probability as shown in Eq. (13) is re-estimated as the expected frequency in state s_i at time $t = 1$. Secondly, Eq. (14) re-estimates the state transition matrix, which is the ratio of the expected number of transitions from state i to j , to the expected number of state transitions from state i to any other state. Finally, Eq. (15) re-estimates the observation matrix as the ratio of expected number of times that observation symbol v_k is observed in state i , to the total expected number of times in state i . The pseudocode of BW is presented in Algorithm 2.

Algorithm 2 Baum-Welch pseudocode

Input: Observation sequence O

Output: Re-estimated state transition matrix $\hat{a}_{ij}$,
Observation probability matrix $\hat{b}_j(v_k)$

- 1: Initialise number of observations M , number of states N , number of iterations G , threshold Th
 - 2: Obtain a_{ij} , $b_j(v_k)$ using initialisation techniques e.g random, uniform, and count-based
 - 3: Obtain expected likelihood, $P(O|\lambda)$
 - 4: **repeat**
 - 5: Calculate forward variable, $\alpha_t(i)$ from Eq. (8)
 - 6: Calculate backward variable $\beta_t(i)$ from Eq. (10)
 - 7: Estimate the state probability, $\gamma_t(i)$ from Eq. (11)
 - 8: Compute the optimal state sequence, $\xi_t(i, j)$ from Eq. (12)
 - 9: Estimate $\hat{a}_{ij}$ and $\hat{b}_j(v_k)$ using Eq. (14) and Eq. (15)
 - 10: Obtain $P(O|\hat{\lambda})$ using estimated parameters
 - 11: $\epsilon = P(O|\hat{\lambda}) - P(O|\lambda)$
 - 12: $P(O|\lambda) = P(O|\hat{\lambda})$
 - 13: $a_{ij} = \hat{a}_{ij}$
 - 14: $b_j(v_k) = \hat{b}_j(v_k)$
 - 15: **until** $\epsilon < Th$ or G iterations reached
 - 16: **return** $\hat{a}_{ij}$, $\hat{b}_j(v_k)$
-

Essentially, there are three well-known techniques for initialising HMM parameters in BW, namely, uniformly, randomly, and count based. A brief description of the three techniques is as follows:

1. *Uniform parameters*: Each element π_i and a_{ij} , is approximated as $1/N$, whereas $b_j(v_k)$ is assigned $1/M$.
2. *Random parameters*: Every element π_i , a_{ij} , and $b_j(v_k)$ are drawn randomly from a uniform distribution between

0 and 1, thereafter each row is scaled to unity by normalising each element with the sum of the corresponding row.

3. *Count-based parameters*: Authors in [43], implement this technique for left-right HMM by using the following:

- $a_{ij} = 0.8$, $j = i + 1$, and evenly distribute the remaining 0.2 for the rest of the elements, where $j \geq i$, and $j \neq i + 1$.
- $b_j(v_k)$ is obtained by counting the number of v_k observations in state j . Each row is normalised so that it sums to unity.

4.3. Segmental K-means Algorithm

The Segmental K-means (SKM) algorithm uses state-optimised joint likelihood for observation data and the underlying Markovian state sequence as the objective function for estimation [44]. Unlike BW algorithm which updates all potential paths for all states in HMM, the SKM updates only the Viterbi (most likely) path reducing the computation time. Authors in [26] describe the SKM learning using the following steps:

1. *Selection of topology*: This involves determining the number of states, number of observations per state, where applicable, and state transitions.
2. *Initialisation of HMM parameters*:

- A and π are estimated randomly, where $\sum_j a_{ij} = 1$, $\sum_i \pi_i = 1$
- B is obtained from the training data and using K -means. The state significance of the observations is ignored and $K = M \times N$ clusters are created. Initially the observation sequence is split into two clusters, and the cluster with a higher mean square error is selected for further division. The process continues until K clusters are created. The centroid of each cluster with lower mean square error is stored until there are K centroids. K -means clustering is performed on the training data using the obtained centroids, $C = \{c_{11}, c_{12}, \dots, c_{1M}, c_{21}, \dots, c_{NM}\}$. Each element of B is then obtained using Eq. 16

$$b_j(v_k) = \frac{n_{jk}}{\sum_k n_{jk}} \quad (16)$$

where n_{jk} is the number of observations $b_j(v_k)$ in Cluster _{jk} , $j \in (1 \dots N)$, $k \in (1 \dots M)$.

3. *Calculation of forward parameter, $\alpha_t(i)$, and backward parameter, $\beta_t(i)$* as in Eqs. 7,8,9,10.
4. *Determining optimal state sequence, $\xi_t(i, j)$* as per Eq. (12).
5. *Estimating the state probability*: The parameter $\gamma_t(i)$, which is the probability of being in state i at time t is computed using Eq. (17)

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j) \quad (17)$$

6. *Updating model parameters*

Steps 3 to 6 are recursive when there is reassignment of new state to an observation. For small amounts of data, the required number of clusters may be difficult to achieve, henceforth, a random or any other initialisation can be used and compute the HMM parameters using the SKM as shown in Algorithm 3.

Algorithm 3 Segmental K-means pseudocode

Input: Observation sequence O

Output: Re-estimated state transition matrix $\hat{a}_{ij}$,
Observation probability matrix $\hat{b}_j(v_k)$

- 1: Initialise $M, N, Th, a_{ij}, b_j(v_k)$
 - 2: Estimate model, λ , using expected likelihood, $P(O|\lambda)$, from $a_{ij}, b_j(v_k)$ and O
 - 3: **repeat**
 - 4: Split O into N states using Viterbi decoding
 - 5: $\hat{a}_{ij} = \frac{\# \text{of state transitions } i \rightarrow j}{\text{Total \# of state transitions from } i}$ as in Eq. 14
 - 6: $\hat{b}_j(v_k) = \frac{\# \text{of occurrences of observation } k \text{ in state } j}{\text{Total \# of observations in state } j}$ as in Eq. 15
 - 7: Normalise row sums of $\hat{a}_{ij}$ and $\hat{b}_j(v_k)$ to unity so that all elements $\in [0, 1]$
 - 8: Estimate $\hat{\lambda}$ from $\hat{a}_{ij}, \hat{b}_j(v_k)$ and O
 - 9: $\lambda = \hat{\lambda}, a_{ij} = \hat{a}_{ij}, b_j(v_k) = \hat{b}_j(v_k)$
 - 10: **until** $|\hat{\lambda} - \lambda| > Th$
 - 11: **return** $\hat{a}_{ij}, \hat{b}_j(v_k)$
-

4.4. Spectral Algorithm

The Spectral Algorithm (SA) uses a method of moments approach to determine HMM parameters. It samples empirical moments from experimental data and consequently determines model parameters that yield expected values equivalent to sampled quantities [35]. Two SAs, one proposed by Hsu, Kakade and Zhang (SA-HKZ) [19], and the other one suggested by Anandkumar, Hsu and Kakade (SA-AHK) [34], were implemented. The former is for HMM learning, whereas the latter outlines a method of moments for mixture models.

The theory about these algorithms has not been presented in this paper but detailed explanation about SA implementation of the two algorithms is provided in [35]. Algorithm 4 shows a generic pseudocode for SA. The N triples (x_1, x_2, x_3) of an observation sequence $O = O_1, O_2 \dots O_T$, where T is a multiple of three is obtained as in Eq. (18):

$$(x_1, x_2, x_3) = \begin{bmatrix} O_1 & O_{T/3+1} & O_{2T/3+1} \\ O_2 & O_{T/3+2} & O_{2T/3+2} \\ \vdots & \vdots & \vdots \\ O_{T/3} & O_{2T/3} & O_T \end{bmatrix} \quad (18)$$

As can be observed from Eq. (18), an observation sequence of length T is arranged into three consecutive columns. Both SAs are advantageous in that the computational and sample complexity is polynomial and that they are not prone to local optima [35]. The HMM parameters may be complex, negative or

$\notin [0, 1]$, hence, absolute values can be used for complex results, and normalisation is applied where appropriate.

Algorithm 4 Spectral algorithm pseudocode

Input: Observation sequence O , number of observations M , number of states N

Output: Re-estimated state transition matrix $\hat{a}_{ij}$,
Observation probability matrix $\hat{b}_j(v_k)$

- 1: Split O into N triples of observations
 - 2: Create empirical estimates of marginal probabilities of singletons, pairs and triples of observations using N triples of observations
 - 3: Compute singular vector decomposition of N on marginal probabilities of pairs to obtain a matrix of left singular vectors
 - 4: Compute $\hat{a}_{ij}$ and $\hat{b}_j(v_k)$ from empirical estimates and matrix of left singular vectors
 - 5: **return** $\hat{a}_{ij}$, $\hat{b}_j(v_k)$
-

4.5. K-means Algorithm

The K-means algorithm is a popular clustering method, faster and easier to implement for a large dataset [45]. The algorithm divides data into K clusters where elements of a cluster are assigned based on the shortest distance to the cluster centre. For MSA, K-means can be implemented by either ignoring the phase significance or by considering the phases, if available.

Algorithm 5 is a pseudocode of K-means without considering the attack phases. If phases of observations are predetermined, K-means is performed on each phase to obtain M clusters.

Algorithm 5 K-Means algorithm pseudocode

Input: Observation sequence O

Output: Re-estimated observation probability matrix $\hat{b}_j(v_k)$

- 1: Initialise number of distinct observations M and number of states N
 - 2: $K = N \times M$
 - 3: Obtain K clusters, $C_{jk} = \{C_{11}, C_{12}, \dots, C_{1M} \dots C_{NM}\}$ using K-means
 - 4: Compute $\hat{b}_j(v_k) = \frac{n_{jk}}{\sum_k n_{jk}}, \forall k, 1 \leq k \leq M, \forall j, 1 \leq j \leq N$ as in Eq. 16
 - 5: Replace zeroes with a very small value and normalise so that sum of rows is unity
 - 6: **return** $\hat{b}_j(v_k)$
-

4.6. Differential Evolution

The Differential Evolution (DE) is an optimisation algorithm that is characterised by genetic mutation, crossover, and selection [29]. The differential mutation and differential crossover are regarded as two operations of generation of offspring [30]. Firstly, the initialised HMM parameters, A and B are transformed into one dimensional vectors and appended to produce

an $(N + M)$ length vector denoted by ab . Secondly, the differential mutation computes new individuals (mutants) using three randomly selected vectors. Thirdly, differential crossover recombines mutant and parent vector to create a trial vector based on predetermined crossover rate between 0.4 and 1 [30]. Finally, new vectors replace their parent vectors if they are closer to the global optimum [29]. Maximum likelihood can be used as the objective function for selection of best member of the population. Algorithm 6 details the pseudocode used to obtain the calibrated HMM parameters, A and B .

Algorithm 6 Differential evolution pseudocode

Input: Observation sequence O

Output: Re-estimated state transition $\hat{a}_{ij}$, observation probability $\hat{b}_j(v_k)$

- 1: Initialise number of distinct observations M , number of states N , number of iterations G , number of D dimensional vectors NP , crossover rate CR
 - 2: Initialise a_{ij} and $b_j(v_k)$ randomly
 - 3: Compute fitness function, F_1 , using expected likelihood from $a_{ij}, b_j(v_k), O$
 - 4: **for** $m \leftarrow 1$ to G **do**
 - 5: $A_{Target} = [a_{11}, a_{12} \dots a_{1N}, a_{21} \dots a_{NN}]$
 - 6: $B_{Target} = [b_1(1), b_1(2) \dots b_1(M), b_2(1) \dots b_N(M)]$
 - 7: Append B_{Target} to A_{Target} creating a row vector AB_{Target} of length $N + M$
 - 8: $Population = NP \times D$
 - 9: **for** $i \leftarrow 1$ to length of A_{Target} **do**
 - 10: Obtain three random individuals A_1, A_2 and A_3
 - 11: $A_{Mutant}(i) = A_1 + F_1 * (A_2 - A_3)$
 - 12: **for** $j \leftarrow 1$ to length of B_{Target} **do**
 - 13: Obtain random individuals B_1, B_2 and B_3
 - 14: $B_{Mutant}(i) = B_1 + F_1 * (B_2 - B_3)$
 - 15: Merge A_{Mutant} and B_{Mutant} to AB_{Mutant}
 - 16: Define vector k_i randomly between 1 and $Population$
 - 17: **for** $i \leftarrow 1$ to $Population$ **do**
 - 18: Generate random number between 0 and 1, $Rand$
 - 19: **if** $Rand \leq CR$ OR $i = k_i$ **then**
 - 20: $AB_{Trial}(i) = AB_{Mutant}(i)$
 - 21: **else**
 - 22: $AB_{Trial}(i) = AB_{Target}(i)$
 - 23: Extract A_{Trial} and B_{Trial} from AB_{Trial}
 - 24: Normalise row sums of A_{Trial} and B_{Trial} to unity so that all values $\in [0, 1]$
 - 25: Compute fitness function, F_2 using A_{Trial} and B_{Trial}
 - 26: **if** $F_2 > F_1$ **then**
 - 27: $a_{ij} = A_{Trial}$
 - 28: $b_j(v_k) = B_{Trial}$
 - 29: Recompute fitness function, F_1
 - 30: **return** $\hat{a}_{ij}, \hat{b}_j(v_k)$
-

5. Experimental Methodology

An HMM, constructed from the DARPA 2000 dataset, having five states and ten observations is illustrated in Fig. 2. The

five states are the phases of the dataset, namely Internet Protocol (IP) sweep, Sadmin probe, Sadmin exploit, DDoS install, and DDoS launch. Apart from the last state, all states navigate to the next state or remain in the same state on occurrence of an observation. The ten observations are arbitrarily selected from Snort IDS alerts and each observation has been assigned a symbol as shown in Table 1.

Figure 2: HMM two-level illustration for DARPA 2000 dataset alerts using five states and ten observations.

Table 1: Selected alerts form DARPA 2000 Scenario one with assigned symbols.

Alert Description	Symbols
DDoS mstream agent to handler	O ₁
ICMP Destination Unreachable Port Unreachable	O ₂
ICMP ping	O ₃
ICMP ping BSDtype	O ₄
Protocol-services rsh root	O ₅
RPC portmap Sadmin request UDP attempt	O ₆
RPC sadmin UDP Ping	O ₇
SDF combination alert	O ₈
SNMP request UDP	O ₉
Telnet login failed	O ₁₀

As observed from Fig. 2, the state-observation mapping is not one to one. Different observations may occur within one state. On the other hand, the same observation may manifest within different states, hence the need for a heuristic approach in correlating observations to states.

5.1. Training and Evaluation Data

DARPA 2000 [16] datasets were used for training and evaluation. This dataset contains two Distributed Denial of Service (DDoS) scenarios. Both the attackers in first and second scenarios were novices; however, the attacker in the second scenario was more stealthy. Each scenario had two nodes for collecting packets: inside the network and within the Demilitarised Zone (DMZ). Furthermore, the dataset provides both a complete dump file capturing the actual full attack scenario, including normal background traffic, and a set of five dump files capturing the individual five phases of the attack traffic. The former was opted for, as it also includes background traffic, as in a typical real life instance; however, the timestamps of the latter were used to identify the phases. Authors in [22] summarise

both these attacks using five stages, namely, IP sweeping, Sadmin probing, Sadmin exploit, DDoS software installation and launching.

5.2. Alerts from DARPA Dataset using Snort IDS

Snort IDS Version 2.9.11.1 [20] was used in Ubuntu 18.04.1 OS to obtain alerts from the DARPA 2000 scenario one. IDS alerts are important as they provide the observations that are fed to the HMM. Initially, Snort IDS default configuration, with rules aggregated by PulledPork 0.7.4 [46] was used; however, customised rules as guided by [22] were added to enhance the attack detection. Table 2 depicts the DARPA 2000 phases and the corresponding customised rules, which were extracted from [47] and modified where applicable. Most of the customised rules are absent from default Snort IDS rules aggregated by PulledPork. Overall, 2306 alerts were obtained from DARPA 2000 scenario one inside dataset, whereas 2949 alerts were obtained from the DARPA 2000 scenario one DMZ dataset. Unlike several authors who use part of the same dataset for training and evaluation, the work in this paper opted to use the entire inside network and DMZ datasets for training and testing, respectively. This is considered ideal as the HMM is trained with the complete attack scenario.

Table 2: Customised Snort IDS rules for DARPA 2000 scenario one.

Phase	Customised Rules
IP Sweep	ICMP-INFO Rules [47]
Sadmin probe	PROTOCOL-RPC rules, ICMP-INFO Rules
Sadmin exploit	PROTOCOL-RPC Rules
DDoS install	PROTOCOL-SERVICES RULES, TEL-NET Rules
DDoS launch	SNMP Rules, DDoS RULES

Table 3 depicts the actual attack packets for individual phases and for the complete DARPA 2000 scenario one inside dataset. Furthermore, the table illustrates the number of alerts obtained from Snort IDS. It can be observed that no alerts were triggered using default rules when considering the individual attack dump files. When the rules were applied on the complete dataset, which includes benign background, 1790 alerts were triggered. Since there were no alerts triggered on individual stages, the 1790 alerts from the complete dataset were deemed to be false positives raised from background traffic.

Table 3: Alerts triggered by customised and default Snort IDS rules for DARPA 2000 scenario one: inside network.

Dataset	Attack packets	# of alerts raised using		
		Default rules	Customised rules	Combined rules
Phase 1	40	0	40	40
Phase 2	158	0	227	227
Phase 3	225	0	74	74
Phase 4	520	0	15	15
Phase 5	73925	0	18	18
Complete	649787	1790	516	2306

In contrast, using customised rules, alerts were triggered during each phase. An improved detection rate of 100% (phases 1 and 2), 33% (phase 3), and 3% (phase 4) was achieved. A slight improved detection rate was observed for phase 5. This phase has the highest number of attack packets due to DDoS flood attack.

An equivalent detection rate pattern is observed in Table 4 for DMZ. In this case, default Snort IDS rules were able to trigger only 35 alerts in phase 5. This was the only phase where the default rules yielded tangible results. Even though both the internal and DMZ datasets were collected during the same attack scenario, they were extracted from different observation nodes across the network. Therefore, Snort IDS generates different alerts between the two networks. For both tested datasets, the customised rules improved the detection rate for each phase except the fifth phase of DMZ. However, the Snort IDS detection performance might be improved further with better suited rules. Nevertheless, since the work in this paper focuses on evaluating accuracy of HMM parameter estimation techniques, the triggered alerts sufficed for the purpose of HMM training and evaluation.

Table 4: Alerts triggered by customised and default Snort IDS rules for DARPA 2000 scenario one: DMZ.

Dataset	Attack packets	# of alerts raised using		
		Default rules	Customised rules	Combined rules
Phase 1	785	0	785	785
Phase 2	148	0	263	263
Phase 3	530	0	173	173
Phase 4	526	0	12	12
Phase 5	34214	35	13	48
Complete	392089	1601	1348	2949

5.3. HMM Modelling

The DDoS attack in DARPA 2000 traversed from phase one to five without retracting, hence a left-right HMM was adopted. The timestamps provided in the individual phase dump files were used to assign phases to the alerts. All alerts prior to the start time of phase 2 dump file were placed in phase 1. Similarly, alerts prior to the start time of phase 3 dump file were considered under phase 2 (excluding those in phase 1). The same process was applied to the subsequent files. There was no one-to-one phase and alert type mapping, hence, the same type of alert would appear in different phases. The alert type was the only parameter used to represent a distinct observation.

5.4. Experimental Setup

The experiments were performed using MATLAB R2018b. A script was written to extract essential fields (timestamp, alert ID, alert description, alert category, source and Destination port and IP addresses) from Snort IDS alerts. All mentioned fields were used to remove duplicate alerts.

The training dataset comprises 30 distinct observation types. On the other hand, the evaluation dataset had 28 distinct observations. One challenge faced by HMM is due to the different observation types between the training and evaluation datasets.

Solutions to address this include: (1) removing the different alert types, and (2) modifying the HMM to include new observations present in evaluation dataset. Since the goal of the work in this paper was primarily to apply various training algorithms and analyse their performance, the first solution was adopted.

The inside network dataset was selected for training as it had more alert types than in DMZ network dataset. Three alert types were exclusive to DMZ, whereas five alert types were exclusive to inside network. Ideally, these distinct types were removed from the dataset resulting into 2278 and 2941 alerts for inside network and DMZ, respectively. Table 5 summarises the initialisation approaches used for EM, BW, SA, and SKM.

Table 5: Combination of initialisation techniques used for training HMM.

Initialisation Technique		Training Technique
A	B	
Random	Random	EM, BW, and SKM
Uniform	Uniform	BW, and SKM
Random	Count-based	BW, and SKM
N/A	N/A	SA-AHK & SA-HKZ
Random	DE	BW, and SKM
Random	SKM	BW, and SKM
Random	K-means without using phases	BW, and SKM
Random	K-means using phases	BW, and SKM

5.5. Determination of Detection and Prediction Accuracy

In order to assess the performance of the learning algorithms, a confusion matrix is usually used to further compute other metrics i.e precision, recall, and F-measure [48]. However, the implemented HMM in this paper, was designed following the DARPA 2000 five-stage MSA and all fed alerts are considered part of this MSA manifestation. Thus, without classified benign alerts, the performance of the deployed training and initialisation techniques in this paper is instead determined by the detection and prediction accuracy. The prediction accuracy is computed as follows:

$$Accuracy (\%) = \frac{Number\ of\ correct\ predictions}{Total\ predictions} \times 100\% \quad (19)$$

The performance evaluation focuses on detection of all states (AS) of an observation sequence, the current state (CS), and the prediction of next state (NS) and next observation (NO). In either case two levels of accuracy are used, where in the first level, the most probable symbol is considered. For the second level accuracy, two most probable symbols are selected and compared with the expected symbol.

Regarding AS and CS detection, the Viterbi decoding has been used to determine the most probable state sequence. The CS is the last state of this sequence. However, prediction of NO is crucial as there are 25 possible observations. Unlike detection of AS and CS, where a hard decision of the most probable state is considered using Viterbi decoding, for NO and NS, a soft decision was opted for. In particular, for both NO and NS, two soft decision methods have been used and these are illustrated in Table 6.

Table 6: Illustration of prediction methods for next state and observation.

Method	Next State	Next Observation
1	Emission and state transition probabilities	
2	Transition probabilities	Emission probabilities

In the first method, method 1, the forward algorithm has been used by considering the current forward variable, $\alpha_t(i)$, and multiplying by its corresponding state transition, a_{ij} , and observation probability distribution, $b_j(v_k)$. The principle behind this algorithm can be observed in Eq. (8) with a detailed example in [49]. Since there are five states and 25 observations, a total of 125 possible state-observation symbol combinations is anticipated. The most likely state-observation symbol combination deduces both the NS and NO.

The second method, method 2, requires computation of the NS and NO separately. NS is computed by multiplying the current forward variable, $\alpha_t(i)$, by the corresponding state transition, a_{ij} . Instead, NO is obtained by multiplying the current forward variable, $\alpha_t(i)$ by the respective observation probability, $b_j(v_k)$. In each case, the possible symbols to be predicted are significantly reduced in comparison to method 1.

6. Results and Discussion

This section analyses the results obtained by implementing and evaluating the corpus of training and initialisation techniques for detecting and predicting MSAs. For the detection and prediction accuracy of all the training techniques, it was necessary to first decide an optimal alert sample window size for evaluation purposes. Ideally, the window size is not supposed to be very long as it would signify that the prediction system would have to wait for a long period of time before making a decision, which is not practical. Window sizes of 50-900 samples were tested to determine the overall optimal window size for all techniques. The corresponding results are shown in Section 6.1. In Section 6.2, results for the selected optimum window size, applied individually on EM, SA, *K*-means, DE and on the standard initialisation techniques of BW and SKM, are presented. Section 6.3 discusses the results obtained from hybrid approaches after leveraging EM, SA, *K*-means and DE, and feeding their HMM output parameters to BW and SKM.

6.1. Detection and Prediction Accuracy for Varying Alert Sample Window Size

To begin with, Fig. 3 shows the effect of varying the alert sample window size from 50 to 900 with respect to AS detection. As anticipated, while increasing the window size, the accuracy varies non-consistently for different techniques. With the exception of EM, Uniform BW and SKM, and SA-HKZ, the rest of the techniques does not demonstrate a considerable variation with an increase in window size.

There is similar performance for uniform BW and SKM, seen overlapping in Fig. 3, with accuracy values steadily decreasing from 38.6% to 33.3%. The detection accuracy of SA-HKZ decreases from 24.0% to 17.2% with an increase in win-

dow size. EM has an increasing exponential detection accuracy between 60.0% and 86.8% as the window size increases and stabilises around a window size of 500. Despite the considerable increase in accuracy for EM as the window size increases, it is not in tandem with the other techniques. Furthermore, the window size is not anticipated to be very large as this would cause the IDS to wait longer prior to making a decision. Thus, to account for the diverse variation in accuracy for all the techniques, a window size of 300 samples is selected for predicting AS.

Figure 3: Comparison of HMM parameter estimation techniques in detecting all states while varying alert sample window size.

The results for detecting the CS are depicted in Fig. 4. The performance pattern for EM is similar to that for AS detection. However, there is a considerable drop in accuracy values for count-based BW (56.0% to 38.8%), count-based SKM (50.1 to 29.3%), and uniform SKM and BW (37.4% to 11.3%) whose results overlap. For *K*-means without using phases (KmeansP in Fig. 4), there is a sharp increase in accuracy from 38.8% to 42.6% between 50 and 100 window sizes, thereafter, the performance deteriorates to 24.1%. For DE, random BW, and count-based BW there is no significant change in accuracy values. Roughly, for CS, a window size of 150 is observed to be an ideal trade-off between performance and reaction time.

Figure 4: Comparison of HMM parameter estimation techniques in detecting the current state while varying alert sample window size.

Fig. 5 presents the results for predicting NS using method 1. From the results it is observed that, just like for AS and CS detection, results for uniform BW and SKM overlap. Furthermore, there is significant performance drop for all techniques,

with increasing window size, except for random SKM, SA-AHK, K -means without using phases (KmeansP in Fig. 5), and DE. Thus, a large window size is not ideal and a window size of 150 is chosen to be an appropriate balance for all techniques, overall.

Figure 5: Comparison of HMM parameter estimation techniques in predicting next state using transition and emission probabilities while varying alert sample window size.

The results for predicting NS using method 2, are depicted in Fig. 6. There is a substantial increase in prediction accuracy for EM from 50-200 window size. There is a steady performance for default K -means up to 200 samples, thereafter the accuracy decreases linearly from 39% to 19%. Both uniform BW and SKM, also have a linear decrease and their results overlap. Similarly, count-based BW and SKM also have significant degradation of accuracy with increasing sample size. On the other hand, both SAs, random BW and SKM, and K -means using phases, all have slight increase in detection accuracy as the window size increases. From the various performance of all techniques, a window size of 200 is considered suitable for all techniques.

Figure 6: Comparison of HMM parameter estimation techniques in predicting next state using transition probabilities while varying alert sample window size.

Fig. 7 portrays the results for predicting NO using method 1. Generally, all techniques exhibits a gradual increase and decrease in prediction accuracy sporadically. However, uniform-SKM has a steady linear increase of accuracy from 200 samples onwards. Bearing in mind that a large window size would mean a long delay in decision making, a window size of 300 is selected.

Figure 7: Comparison of HMM parameter estimation techniques in predicting next observation using transition and emission probabilities while varying alert sample window size.

The results for predicting NO using method 2, are shown in Fig. 8. The prediction accuracy for SA-AHK and DE is mainly constant regardless of window size increase. Results for uniform BW and SKM overlap with prediction accuracy ranging from 23% to 30% as window size increases. On the other hand, for all approaches of K -means, there is a steady performance up to 200 samples, thereafter, the performance significantly drops from 50% to 35%. With the exception of DE, SA-AHK, and K -means techniques, there is a constant performance up to about 200 samples, after which, the accuracy values significantly increase. A crossover between all techniques can thus be observed to be at 200 samples.

Figure 8: Comparison of HMM parameter estimation techniques in predicting next state using emission probabilities while varying alert sample window size.

For all detection and prediction scenarios, it is difficult to determine a break-even point as there are generally varying responses for all techniques. An average of all the suggested window sizes yielded a selected window size of 200. This value is adopted and utilised in the analysis within the following subsections.

6.2. Performance Analysis of HMM Training Techniques using a Fixed Alert Sample Window Size

In this Section, the results obtained using a fixed alert sample window size of 200 as determined in Section 6.1, are discussed. Initially, the results for AS and CS are discussed fol-

Table 7: Percentage detection accuracy using EM, BW, SKM, SA, *K*-means, and DE with an alert sample window size of 200.

Detection approach	Percentage Accuracy (%) for individual training techniques											DE
	EM	BW			SKM			SA		K-means		
		Uniform	Rand.	Count	Uniform	Rand.	Count	AHK	HKZ	No Phase	Phases	
AS	72.79	37.68	13.63	58.58	37.60	15.16	51.35	11.12	46.47	39.94	20.79	42.72
CS	71.69	33.97	13.83	54.51	33.97	15.91	47.72	14.67	46.52	39.55	19.96	41.74

lowed by all approaches for NS and NO using two levels of accuracy.

Table 7 depicts the AS and CS results and it can be observed that in either case, the EM is a better technique followed by count-based BW and count-based SKM. The least performance for AS is noted to be the SA-AHK, whereas for CS, it is the random BW. The count-based techniques are expected to perform better than random and uniform initialisation of BW as they use phase-observation information. However, *K*-means without using phases performs better than its counterpart which uses phases.

Fig. 9 shows a bar graph comparing NS determination using both methods 1 and 2 for level 1 accuracy. With the exception of count-based BW, SA-HKZ and *K*-means using phases, it can be deduced that method 2 improves or maintains the prediction accuracy. EM has the highest prediction accuracy of 81.0% for method 2, whereas, SA-HKZ has the best performance when using method 1 at 46%. Regarding BW and SKM, it can be observed that the count-based and random SKM techniques perform better than the rest of BW and SKM techniques.

Figure 9: Next state level 1 percentage accuracy for using EM, BW, SKM, SA, *K*-means, and DE with an alert sample window size of 200.

Results in Fig. 10 depict a bar graph that compares NS level 2 prediction using method 1 against method 2. A significant improvement is achieved using method 2 except with the SA-HKZ technique. A plausible reason is that method 2 reduces the number of probable state forecasts to five. Instead, using method 1, there are 25 possible state forecasts for each phase. For method 2, EM has the highest prediction accuracy of 90% followed by DE, which achieves an accuracy of 61%. Random SKM using method 2, had better performance compared to both BW and SKM.

The results for NO level 1 prediction using both methods

Figure 10: Next state level 2 percentage accuracy for using EM, BW, SKM, SA, *K*-means, and DE with an alert sample window size of 200.

1 and 2 are illustrated in Fig. 11. It is noted that method 2 has significant prediction accuracy improvement over method 1 with the exception of SA-AHK and DE. *K*-means without using phases performs better than all the other techniques while using method 2.

Figure 11: Next observation level 1 percentage accuracy for using EM, BW, SKM, SA, *K*-means, and DE with an alert sample window size of 200.

The results for NO level 2 prediction using method 1 and 2, are presented in Fig. 12. Overall, level 2 NO prediction performs better than level 1 accuracy with an improvement of at least two times. The best performance is obtained when using random SKM followed by both count-based SKM and *K*-means without using phases. Also, with the exception of SA-AHK and DE, using method 2 significantly improves the prediction accuracy.

It can be deduced that for either NS or NO prediction, the performance of method 2 is better than that of method 1. It is

Figure 12: Next observation level 2 percentage accuracy for using EM, BW, SKM, SA, *K*-means, and DE with an alert sample window size of 200.

also observed that no single technique performs equally well for detection of AS and CS, and for prediction of NS and NO. Hence, it would be essential to consider different techniques, for practical implementation, depending on the application.

6.3. Performance Analysis of HMM Hybrid Techniques

In this section, EM, SA, *K*-means, and DE techniques are used as initialisation techniques to BW and SKM training algorithms. The results for detection of AS and CS, and prediction approaches for NS and NO using the two levels, have been presented in Table 8.

To begin with, for AS prediction, EM initialisation with BW degrades the accuracy by 6.54%, whereas with SKM the accuracy drops by 0.6%. Performance is also reduced by 4.02% for SA-AHK when trained with BW, however, an insignificant 0.55% improvement in performance is obtained with SKM. SA-HKZ has 46.47% prediction accuracy. It is, thus, noted that there is a performance drop in accuracy when leveraged with BW. On the other hand, a 1.11% increase in accuracy is achieved when combined with SKM. *K*-means without using phases, *K*-means using phases and DE, have detection accuracy of 39.94%, 20.79% and 42.72%, respectively. Among all these three initialisation techniques, a slight increase in accuracy of 1.59% is observed when leveraging DE with BW, while for the rest, there is performance degradation. Overall, the hybrid technique of leveraging initialisation techniques with BW and SKM are not beneficial for AS prediction as there are very few cases having small improvements.

With respect to CS, prior to leveraging with BW and SKM, EM, SA-AHK, SA-HKZ, *K*-means without using phases, *K*-means using phases, and DE have detection accuracy of 71.69%, 14.67%, 46.52%, 39.55%, 19.96%, and 41.74%, respectively. Comparing these results with the results in Table 7 after leveraging with BW and SKM, it can be noted that there is a slight detection accuracy improvement of 0.63% (EM + SKM), 2.66% (SA-HKZ + SKM), 0.29% (*K*-means using phases + BW), and 0.69% (*K*-means using phases + SKM). Despite these results of minor improvements, in general, there is a performance drop in CS detection accuracy when leveraging BW and SKM.

Considering NS level 1 prediction using method 1, it should be noted that prior to leveraging with BW and SKM the prediction accuracy was 36.88% (EM), 11.2% (SA-AHK), 46.78% (SA-HKZ), 36.35% (*K*-means without using phases), 16.85% (*K*-means using phases), and 18.75% (DE). Contrasting these results with those presented in Table 7, it is noted that there is an increase in accuracy by 16.53% (EM + BW), 13.5% (EM + SKM), 0.06% (SA-HKZ + SKM), 1.12% (*K*-means without using phases + BW), 5.7% (*K*-means using phases + SKM), 5.7% (*K*-means using phases + SKM). Therefore, leveraging EM with BW and SKM, demonstrates a significant improvement for most techniques and would be proposed for inclusion in NS level 1 prediction.

Regarding NS level 1 prediction using method 2, the prediction accuracy of individual techniques was 81.0% (EM), 13.44% (SA-AHK), 22.91% (SA-HKZ), 36.84% (*K*-means without using phases), 15.68% (*K*-means using phases), and 31.73% (DE). From the results depicted in Table 7, it is observed that there is an increase in accuracy of 23.86% (SA-HKZ + SKM), 2.45% (*K*-means without using phases + BW), 6.62% (*K*-means using phases + SKM), 4.24% (DE + BW), and 5.99% (DE + SKM).

A similar analysis of NS level 2 prediction using method 1 is made and it is observed that there is a significant increase of 18.79% and 13.72% for EM feeding BW and SKM, respectively. The same analysis is also made for NS level 2 prediction using method 2 and it is noted that there is a significant prediction accuracy enhancement of 21.9% (SA-AHK + SKM), 24.33% (SA-HKZ + SKM), 17.3% (*K*-means without using phases + BW), and 14.71% (*K*-means without using phases + SKM).

Lastly, an analysis of NO prediction is made using level 1 and 2 for both methods 1 and 2. When using method 1, there is no significant improvement for NO level 1 prediction with hybrid approaches. The highest improvement is 1.62% for SA-HKZ + BW. However, there is significant improvement when leveraging BW and SKM using method 2. Specifically, EM achieves 33.41% improvement when fed into BW, whereas DE had 20.88% and 24.12% accuracy enhancement when using BW and SKM, respectively.

Regarding level 2 NO prediction using method 2, there is significant improvement for all techniques, when leveraged with BW and SKM, with the exception of *K*-means without using phases combined with BW. Particularly, the highest improvement appears for DE which has 44.95% increase when leveraged with SKM and 35.02% when leveraged with BW. EM leveraged with BW had the second highest improvement by 37.89%, reaching 80.37% in prediction accuracy.

7. Conclusion

This paper has analysed six machine learning methods, specifically EM, SA (with two sub-methods), *K*-means, and DE, alongside the BW and SKM algorithms, for detecting network MSA states, and predicting the next state and observation. Snort IDS version 2.9.11.1 was used in Ubuntu 18.04.1 operating system to trigger alerts, which were mapped to observations for HMM. Both default and customised Snort IDS rules have been used to

Table 8: Percentage accuracy for detecting all states, the current state, and predicting the next state leveraged with BW and SKM using alert sample window size of 200.

Prediction approach	Training Technique	Percentage accuracy (%) for initialisation techniques					
		EM	SA		K-means		DE
			AHK	HKZ	No Phase	Phases	
All States	BW	66.25	7.10	41.84	37.58	19.08	44.31
	SKM	72.71	11.67	47.78	39.63	19.45	41.02
Current State	BW	68.62	6.93	44.55	37.03	20.25	40.72
	SKM	72.31	10.54	49.18	39.29	20.65	38.64
NS Level 1 using A and B	BW	53.41	7.19	36.05	37.47	9.45	14.08
	SKM	50.38	10.07	46.84	40.97	14.67	11.27
NS Level 1 using A	BW	68.59	8.94	43.49	6.52	21.38	35.97
	SKM	76.21	13.06	46.77	39.29	21.67	37.72
NS Level 2 using A and B	BW	55.67	7.52	39.33	40.46	13.10	17.58
	SKM	50.60	10.91	48.63	45.64	20.90	17.04
NS Level 2 using A	BW	84.75	14.81	61.73	60.05	39.18	49.14
	SKM	87.30	49.51	62.53	44.84	61.62	53.45
NO Level 1 using A and B	BW	7.26	7.52	4.89	6.68	7.84	5.69
	SKM	6.71	4.85	6.79	5.18	4.85	4.74
NO Level 1 using A	BW	56.66	9.12	14.01	25.87	13.90	21.63
	SKM	25.98	9.27	11.27	51.51	12.84	24.88
NO Level 2 using A and B	BW	9.16	10.11	7.99	9.67	9.78	8.72
	SKM	9.05	8.79	8.90	8.43	8.50	8.17
NO Level 2 using A	BW	80.37	12.95	36.48	50.64	36.34	37.03
	SKM	46.73	25.79	44.87	68.70	33.16	46.95

generate alerts. The experiments were performed using MATLAB R2018b scripts and this paper opted to use the entire inside network and DMZ DARPA 2000 datasets for training and testing respectively.

The EM, SA, *K*-means, and DE were used both individually, and synergistically with BW and SKM in a hybrid approach. The three popular initialisation techniques for BW and SKM, i.e uniform, random, and count-based, have also been analysed.

Furthermore, the evaluation of the techniques has been based on the following criteria: detection of all states (AS) and current state (CS), and predicting next state (NS) and next observation (NO). For prediction of NS and NO, two methods have been compared. The first method, method 1, is using the forward parameter with both the state transition and emission probabilities, whilst the second method, method 2, is using the forward parameter and incorporates either the state transition or the emission probabilities for predicting NS and NO, respectively.

The results show that no single technique outperforms the others for all the evaluation criteria mentioned above. However, EM performs better with a detection accuracy of 73% for AS and 72% for CS, and a prediction accuracy of 90% for NS level 2 using method 2. Instead, for NO prediction, SKM-random using method 2 achieves a slightly higher prediction accuracy of 10% than EM.

The prediction of NO and NS using method 2 performs better than using method 1. For instance, using method 2, *K*-means without using phases had an increase of about 40% prediction accuracy for level 1 and 2, when compared to method 1. Furthermore, when using method 2, EM achieves a prediction accuracy increase of 53% and 44% for level 1 and 2, respectively.

Finally, using the hybrid approach, leveraging BW and SKM

with the various HMM parameter estimating techniques, enhances the prediction performance. Specifically, for NS level 2 prediction accuracy using method 2, the EM technique together with SKM reaches 87.30%. In addition, a significant improvement is evident in DE together with SKM, which achieves an NO level 2 prediction accuracy increase by 44.95%. Similarly, the NO level 2 prediction accuracy reaches 80.37% using the EM together with BW.

References

- [1] J. Arshad, M. A. Azad, M. M. Abdellatif, M. H. U. Rehman, K. Salah, Colide: a collaborative intrusion detection framework for internet of things, *IET Networks* 8 (1) (2019) 3–14.
- [2] J. Svoboda, I. Ghafir, V. Přenosil, Network Monitoring Approaches: An Overview, *International Journal of Advances in Computer Networks and Its Security* 5 (2) (2015) 88–93.
- [3] I. Ghafir, V. Přenosil, Advanced Persistent Threat and Spear Phishing Emails, in: *International Conference on Distance Learning, Simulation and Communication*, Brno, Czech Republic, 2015, pp. 34–41.
- [4] M. Husák, J. Komárková, E. Bou-Harb, P. Čeleda, Survey of Attack Projection, Prediction, and Forecasting in Cyber Security, *IEEE Communications Surveys Tutorials* 21 (1) (2018) 640–660.
- [5] S. J. Yang, H. Du, J. Holsopple, M. Sudit, Attack Projection, in: *Cyber Defense and Situational Awareness*, Springer, Cham, Switzerland, 2014, pp. 239–261.
- [6] A. A. Ahmed, N. Ahlami, K. Zaman, Attack Intention Recognition: A Review, *International Journal of Network Security* 19 (2) (2017) 244–250.
- [7] M. Abdhamed, K. Kifayat, Q. Shi, W. Hurst, Intrusion Prediction Systems, in: *Information Fusion for Cyber-Security Analytics*, Springer International Publishing, Cham, Switzerland, 2017, pp. 155–174.
- [8] Y.-B. Leau, S. Manickam, Network Security Situation Prediction: A Review and Discussion, in: *Communications in Computer and Information Science*, Springer, Berlin, Heidelberg, 2015, pp. 424–435.
- [9] A. Kott, *Towards Fundamental Science of Cyber Security*, Springer, New York, USA, 2014, Ch. 1, pp. 1–13.

- [10] N. S. Safa, C. Maple, S. Furnell, M. A. Azad, C. Perera, M. Dabbagh, M. Sookhak, Deterrence and prevention-based model to mitigate information security insider threats in organisations, *Future Generation Computer Systems* 97 (2019) 587 – 597.
- [11] N. Polatidis, E. Pimenidis, M. Pavlidis, S. Papastergiou, H. Mouratidis, From product recommendation to cyber-attack prediction: generating attack graphs and predicting future attacks, *Evolving Systems* (2018) 1–12.
- [12] A. A. Ramaki, M. Khosravi-Farmad, A. G. Bafghi, Real time alert correlation and prediction using Bayesian networks, in: 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC), IEEE, Rasht, Iran, 2015, pp. 98–103.
- [13] H. A. Kholidy, A. Erradi, S. Abdelwahed, A. Azab, A Finite State Hidden Markov Model for Predicting Multistage Attacks in Cloud Systems, in: 12th International Conference on Dependable, Autonomic and Secure Computing, IEEE, Dalian, China, 2014, pp. 14–19.
- [14] A. Bar, B. Shapira, L. Rokach, M. Unger, Identifying Attack Propagation Patterns in Honeypots Using Markov Chains Modeling and Complex Networks Analysis, in: International Conference on Software Science, Technology and Engineering (SWSTE), IEEE, Beer-Sheva, Israel, 2016, pp. 28–36.
- [15] R. Píbil, V. Lisý, C. Kiekintveld, B. Bošanský, M. Pěchouček, Game Theoretic Model of Strategic Honeypot Selection in Computer Networks, in: *Decision and Game Theory for Security*, Springer, Berlin, Heidelberg, 2012, pp. 201–220.
- [16] MIT, 2000 DARPA Intrusion Detection Scenario Specific Data Sets, (2018). <https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusion-detection-scenario-specific-datasets> (accessed Apr 20, 2019).
- [17] J. Navarro, A. Deruyver, P. Parrend, A systematic survey on multi-step attack detection, *Computers & Security* 76 (2018) 214–249.
- [18] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE* 77 (2) (1989) 257–286.
- [19] D. Hsu, S. M. Kakade, T. Zhang, A Spectral Algorithm for Learning Hidden Markov Models, *Journal of Computer and System Sciences* 78.5 (2012) 1460–1480.
- [20] Cisco, Snort - Network Intrusion Detection and Prevention System, (2018). <https://www.snort.org> (accessed Apr 20, 2019).
- [21] E. Raftopoulos, X. Dimitropoulos, IDS Alert Correlation in the Wild With EDGe, *IEEE Journal on Selected Areas in Communications* 32 (10) (2014) 1933–1946.
- [22] T. A. Shawly, A. A. Elghariani, J. Kobes, A. Ghafoor, Architectures for Detecting Real-time Multiple Multi-stage Network Attacks Using Hidden Markov Model, *CoRR abs/1807.09764* (2018).
- [23] M. Imran, M. T. Afzal, M. A. Qadir, Similarity-Based Malware Classification Using Hidden Markov Model, in: Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec), IEEE, Jakarta, Indonesia, 2015, pp. 129–134.
- [24] U. S. K. P. M. Thanthrige, J. Samarabandu, X. Wang, Intrusion Alert Prediction Using a Hidden Markov Model, *CoRR abs/1610.07276* (2016).
- [25] P. Larue, P. Jallon, B. Rivet, Modified k-mean clustering method of HMM states for initialization of Baum-Welch training algorithm, in: 19th European Signal Processing Conference (EUSIPCO 2011), EUSIPCO proceedings, Barcelona, Spain, 2011, pp. 951–955.
- [26] T. K. Bhowmik, J.-P. van Oosten, L. Schomaker, Segmental k-means learning with mixture distribution for HMM based handwriting recognition, in: Springer, Berlin, Heidelberg, 2011, pp. 432–439.
- [27] Z. Moghaddam, M. Piccardi, Training Initialization of Hidden Markov Models in Human Action Recognition, *IEEE Transactions on Automation Science and Engineering* 11 (2) (2014) 394–408.
- [28] T. Liu, J. Lemeire, L. Yang, Proper initialization of Hidden Markov models for industrial applications, in: IEEE China Summit & International Conference on Signal and Information Processing (ChinaSIP), IEEE, Xi'an, China, 2014, pp. 490–494.
- [29] Á. A. R. Sá, A. O. Andrade, A. B. Soares, S. J. Nasuto, Estimation of Hidden Markov Models Parameters using Differential Evolution, in: AISB 2008 Convention Communication, Interaction and Social Intelligence, AISB, Aberdeen, UK, 2008.
- [30] K. Ghanem, A. Draa, E. Vyumvuhore, A. Simbabawe, Differential Evolution to Optimize Hidden Markov Models Training: Application to Facial Expression Recognition, *Journal of Computing and Information Technology* 23 (2) (2015) 157–170.
- [31] R. Storn, K. Price, Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, *Journal of Global Optimization* 11 (4) (1997) 341–359.
- [32] M. Keshk, H. Singh, H. Abbass, Automatic estimation of differential evolution parameters using Hidden Markov Models, *Evolutionary Intelligence* 10 (3–4) (2018) 77–93.
- [33] L. Xia, H. Fang, Z. Li, HMM parameters estimation with inequality constraints, in: Proceedings of the 32nd Chinese Control Conference, IEEE, Xi'an, China, 2013, pp. 7869–7872.
- [34] A. Anandkumar, D. J. Hsu, S. M. Kakade, A Method of Moments for Mixture Models and Hidden Markov Models, *CoRR abs/1203.0683* (2012).
- [35] C. Mattfeld, Implementing spectral methods for hidden Markov models with real-valued emissions, Master's thesis, ETH Zurich (2014).
- [36] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, in: Proceedings of the 4th International Conference on Information Systems and Security and Privacy (ICISSP 2018), INSTICC, Funchal, Madeira, Portugal, 2018, pp. 108–116.
- [37] G. Brogi, E. Di Bernardino, Hidden Markov models for advanced persistent threats, *International Journal of Security and Networks (IJSN)* 14 (4) (2019) 181–190.
- [38] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society. Series B (Methodological)* 39 (1977) 1–38.
- [39] H. Tan, S. Ma, Learning partially observable Markov decision model with EM algorithm, in: 7th International Conference on Application of Information and Communication Technologies, IEEE, Baku, Azerbaijan, 2013, pp. 1–4.
- [40] C.-M. Chen, H.-W. Hsiao, P.-Y. Yang, Y.-H. Ou, Defending malicious attacks in Cyber Physical Systems, in: IEEE 1st International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA), IEEE, Taipei, Taiwan, 2013, pp. 13–18.
- [41] B. Benyacoub, S. E. Bernoussi, A. Zoglat, Credit scoring model based on Baum-Welch method, in: ACS International Conference on Computer Systems and Applications (AICCSA), IEEE, Ifrane, Morocco, 2013, pp. 1–5.
- [42] N. El Gemayel, J. Schloemann, R. M. Buehrer, F. K. Jondral, Improved Indoor Positioning Using the Baum-Welch Algorithm, in: IEEE Globecom Workshops (GC Wkshps), IEEE, San Diego, CA, USA, 2015, pp. 1–6.
- [43] N. C. Laan, D. F. Pace, H. Shatkay, Initial model selection for the Baum-Welch algorithm as applied to HMMs of DNA sequences, in: First Canadian Student Conference on Biomedical Computing, Queen's University, Kingston, ON, Canada, 2005.
- [44] B.-H. Juang, L. Rabiner, The segmental K-means algorithm for estimating parameters of hidden Markov models, *IEEE Transactions on Acoustics, Speech, and Signal Processing* 38 (9) (1990) 1639–1641.
- [45] X. Wang, H. Wu, Z. Yi, Research on Bank Anti-Fraud Model Based on K-Means and Hidden Markov Model, in: IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), IEEE, Chongqing, China, 2018, pp. 780–784.
- [46] J. Cummings, M. Shirk, PulledPork Team, PulledPork, (2017). <https://github.com/shirkdog/pulledpork> (accessed Apr 20, 2019).
- [47] E. Eldondev, Snort Rules, (2011). <https://github.com/eldondev/Snort> (accessed Apr 20, 2019).
- [48] M. A. Azad, M. Alazab, F. Riaz, J. Arshad, T. Abullah, Socioscope: I know who you are, a robo, human caller or service number, *Future Generation Computer Systems* 105 (2020) 297 – 307.
- [49] I. Ghafir, K. G. Kyriakopoulos, S. Lambotharan, F. J. Aparicio-Navarro, B. Assadhan, H. Binsalleeh, D. M. Diab, Hidden Markov Models and Alert Correlations for the Prediction of Advanced Persistent Threats, *IEEE Access* 7 (2019) 99508–99520.