Universal Source Coding for Classification and Anomaly Detection

Fathima Zarin Faizal
180070018

1 Introduction

This report briefly summarizes how universal source coding methods have been used for classification and anomaly detection.

In section 2, we introduce the topic of universal source coding and discuss the Lempel-Ziv algorithm. In section 3 we see how LZ parsing has been used for classification. In section 4 we see how the LZ parsing algorithm has been used for Anomaly Detection in the context of detecting Botnet attacks in communication networks.

2 Universal Source Coding

The sequence generated by a source usually contains some redundancy and the point of source coding is to remove it and represent the information contained in the sequence in as few symbols as possible. We first start with a brief review of various information theoretic terms and results from [1] that would be used throughout the report.

Definition 2.1. A *source code* C for a random variable X is a mapping from $\mathcal{X}$, the range of X to D^*, the set of finite-length strings of symbols from a D-ary alphabet. $C(x)$ denotes the codeword corresponding to x and $l(x)$ denotes the length of $C(x)$.

Definition 2.2. The *expected length* of a $L(C)$ of a source code $C(x)$ for a random variable X with probability mass function $p(x)$ is given by:

$$L(C) = \sum_{x \in \mathcal{X}} p(x) l(x).$$

Theorem 1. *The expected length L of any instantaneous D-ary code for a random variable X is greater than or equal to the entropy $H_D(X)$, i.e.,*

$$L \geq H_D(X).$$

There are various coding schemes like Huffman coding that, when given $p(x)$ compresses an i.i.d. source to its entropy limit $H(X)$, but if the code was designed assuming an incorrect distribution, say $q(x)$, then a penalty of $D(p\|q)$, i.e., the KL divergence of $p(x)$ and $q(x)$ is incurred. In many practical applications, the probability distribution underlying the source is unknown and one would have to resort to universal coding techniques like Arithmetic Coding and the Lempel-Ziv algorithm to encode the source information.

2.1 Lempel-Ziv Algorithm

There are many variations of the LZ algorithm but we will consider the tree-structured version where the input sequence from the source is parsed into phrases, where each phrase is the shortest continuous string that has not been seen so far in the sequence. For instance, the sequence AABABBAAABA will be parsed as A, AB, ABB, AA, ABA. Let $c(n)$ be the number of phrases thus created, where n is the number of elements in the input sequence. One can represent these phrases using a rooted tree where each phrase is a path from the root node to some other node.
The compressed sequence consists of a list of $c(n)$ pair of pointers, one pointing to the previous occurrence of the prefix of each phrase and the other pointing to the last bit of the phrase. Each pointer requires $\log c(n)$ bits and hence the total length of the compressed sequence is approximately $c(n) \log c(n)$.

Theorem 2. *Let $\{X_i\}_{-\infty}^{\infty}$ be a binary stationary ergodic stochastic process. Let $l(X_1, X_2, \ldots, X_n)$ be the LZ codeword length associated with $X_1, X_2, \ldots, X_n$. Then*

$$\limsup_{n\to\infty} \frac{1}{n} l(X_1, X_2, \ldots, X_n) \leq H(\mathcal{X})$$

with probability 1 where $H(\mathcal{X})$ is the entropy rate of the process.

Thus the Lempel-Ziv compression algorithm achieves the entropy rate of the source without knowing the underlying source distribution.

3 Universal Classification using LZ compression

3.1 Introduction

This section surveys the work done in [2].
Let an unknown ℓ^{th}-order Markov process $p(\cdot)$ over a finite alphabet $\mathcal{D}$ with $D =$

$|\mathcal{D}|$ letters emit a sequence $\boldsymbol{x} = (x_1, \ldots, x_n)$. To estimate the n^{th} order entropy, i.e. $\frac{1}{n}\log p(\cdot)$ when ℓ is unknown, one can use Theorem 2, i.e. the LZ codeword length for $\boldsymbol{x}$ divided by n is a computationally feasible estimate of the entropy. Let $c(\boldsymbol{x})$ denote the number of phrases resulting from the LZ compression algorithm outlined earlier. Then

$$\lim_{n\to\infty}\left[-\frac{1}{n}\log p(\boldsymbol{x}) - \frac{1}{n}c(\boldsymbol{x}\log c(\boldsymbol{x})\right] = 0, \text{ almost surely.} \tag{1}$$

The paper [2] generalizes this result to the case when there are two Markov probability measures $p(\cdot)$ and $q(\cdot)$.

3.2 Empirical Divergence based on the LZ Algorithm

Consider the following variation of the LZ parsing algorithm of a vector $\boldsymbol{z}$ with respect to another vector $\boldsymbol{x}$. The first phrase of $\boldsymbol{z}$ with respect to $\boldsymbol{x}$ is defined to be the longest prefix of $\boldsymbol{z}$ that appears in $\boldsymbol{x}$, i.e. the largest integer m s.t. $(z_1, \ldots, x_m) = (x_i, \ldots, x_{i+m-1})$ for some i. If $m = 0$, then the first phrase of $\boldsymbol{z}$ with respect to $\boldsymbol{x}$ is z_1. Now start from z_{m+1} and find the longest prefix that is there in $\boldsymbol{x}$. Continue this till the entire vector $\boldsymbol{z}$ has been parsed. Let $c(\boldsymbol{z}|\boldsymbol{x})$ denote the number of phrases in $\boldsymbol{z}$ with respect to $\boldsymbol{x}$.
For two vectors $\boldsymbol{z}$ and $\boldsymbol{x}$ of length n, define

$$\begin{aligned} Q(\boldsymbol{z}\|\boldsymbol{x}) &= \frac{1}{n}c(\boldsymbol{z}|\boldsymbol{x})\log n \\ \Delta(\boldsymbol{z}\|\boldsymbol{x}) &= \frac{1}{n}[c(\boldsymbol{z}|\boldsymbol{x})\log n - c(\boldsymbol{z})\log c(\boldsymbol{z})] \end{aligned}$$

Theorem 3. *Let $p(\cdot)$ be a stationary Markov source of order ℓ and let μ be an arbitrarily small number. Let $\boldsymbol{x}$ be a realization of $p(\cdot)$. Then,*

(a) For every fixed $\boldsymbol{z} \in \mathcal{D}^n$ for which $p(\boldsymbol{z}) > 0$,

$$\begin{aligned} &p\left[\boldsymbol{x} : -\frac{1}{n}\log p(\boldsymbol{z}) - Q(\boldsymbol{z}\|\boldsymbol{x}) < -2\mu\log\frac{1}{\delta}\right] \\ &\leq e^{-(n^\mu/\log n)K(\delta,\ell)+o(n^\mu/\log n)} \end{aligned}$$

where

(a) $\delta = \min_{a^{\ell+1}\in\mathcal{D}^{\ell+1}:p\left(a^{\ell+1}\right)>0} p\left(a^{\ell+1}\right)$

(b) $K(\delta,\ell) = \delta^{\ell+1}\log\frac{1}{1-\delta}$,

(b) For every $\boldsymbol{z} \in \mathcal{D}^n$ *for which* $p(\boldsymbol{z}) > 0$,

$$p\left[\boldsymbol{x} : -\frac{1}{n}\log p(\boldsymbol{z}) - Q(\boldsymbol{z}\|\boldsymbol{x}) > 2\mu \log\frac{1}{\delta}\right] \leq n^{-\mu/2} + o\left(n^{-\mu/2}\right). \qquad (2)$$

(c) Let $\boldsymbol{z}$ *be a realization of a stationary Markov source of order* ℓ' *with an underlying probability measure* $q(\cdot)$. *Then* (2) *can be replaced by a tighter bound for almost every* $\boldsymbol{z} \in D^n$ *(relative to the probability measure* $q(\cdot)$ *) as follows:*

$$p\left[\boldsymbol{x} : -\frac{1}{n}\log p(\boldsymbol{z}) - Q(\boldsymbol{z}\|\boldsymbol{x}) > 2\mu \log\frac{1}{\delta}\right] \leq e^{-\frac{1}{2}\left(\frac{\mu^2}{\log^2 n}\right)n^{K'(\delta,\delta')}\left[1-o\left(n^{-\mu^2}\right)\right]}$$

for every $\boldsymbol{z} \in D^n - B$ *for which* $p(\boldsymbol{z}) > 0$, *where* B *is a subset with the property:*

$$q(\boldsymbol{z} \in B) \leq e^{-K''(\delta,D,\mu)\frac{n^{\mu'}}{\log n}+o\left(\frac{n^{\mu'}}{\log n}\right)}$$

and

(a) $\delta' = \min_{a^{\ell+1}\in\mathcal{D}^{\ell+1}:q\left(a^{\ell+1}\right)>0} q\left(a^{\ell+1}\right)$,

(b) $K' = \frac{1}{4}\frac{\log\left(\frac{1}{1-\delta'}\right)}{\log\frac{1}{\delta}}$

(c) $K''(\delta,\delta',\mu) = \frac{\mu}{2(1+\mu)}\frac{\log\frac{1}{1-\delta}}{\log\frac{1}{\delta}}\log\frac{1}{1-\delta'}$,

(d) $\mu' = 1 - \frac{1}{4}\log\left(\frac{1}{1-\delta'}\right)/\log\frac{1}{\delta}$.

Theorem 3 says that the probability that $Q(\boldsymbol{z}\|\boldsymbol{x})$ exceeds $-\frac{1}{n}\log p(\boldsymbol{z})$ by a large value decays almost exponentially, but the probability that $Q(\boldsymbol{z}\|\boldsymbol{x})$ is smaller than $-\frac{1}{n}\log p(\boldsymbol{z})$ by a large value decays polynomially unless $\boldsymbol{z}$ belongs to some subset of high probability under $q(\cdot)$.

Corollary 3.1.

$$\lim_{n\to\infty}\left[-\frac{1}{n}\log p(\boldsymbol{z}) - Q(\boldsymbol{z}\|\boldsymbol{x})\right] = 0$$

a.s. for every sequence of pairs $(\boldsymbol{x},\boldsymbol{z})$, *with respect to a probability measure for pairs given by*

$$f(\boldsymbol{x},\boldsymbol{z}) \triangleq p(\boldsymbol{x})q(\boldsymbol{z})$$

Hence, one can consider this result to be analogous to (1) and essentially split $Q(\boldsymbol{z}\|\boldsymbol{x})$ into an estimate of the entropy, i.e. $\frac{1}{n}c(\boldsymbol{z}\log c(\boldsymbol{z})$ and $\Delta(\boldsymbol{z}\|\boldsymbol{x})$ as $-\log p(\boldsymbol{z}) = nH_{\boldsymbol{z}} + nD\left(q_{\boldsymbol{z}}\|p\right)$ and obtain $\lim_{n\to\infty}\left[\Delta(\boldsymbol{z}\|\boldsymbol{x}) - D\left(q_{\boldsymbol{z}}\|p\right)\right] = 0$ almost surely (with conditions similar to Corollary 3.1). The theorem enabling this result is provided below.

Theorem 4. *(a) Let $\boldsymbol{x}$ be a realization of a stationary ℓ th-order Markov source $p(\cdot)$. Then, for every $\boldsymbol{z} \in \mathcal{D}^n$ such that $p(\boldsymbol{z}) > 0$*

$$p\left(\boldsymbol{x} : D\left(q_{\boldsymbol{z}}\|p\right) - \Delta(\boldsymbol{z}\|\boldsymbol{x}) < -3\mu\log\frac{1}{\delta}\right) \leq e^{-(n^{\mu}/\log n)K(\delta,\ell)-o(n^{\mu}/\log n)}$$

where δ and $K(\delta,\ell)$ are same as defined earlier.

(b) Let $\boldsymbol{x}$ be a realization of a stationary ℓ th-order Markov source $p(\cdot)$ and let $\boldsymbol{z}$ be a realization of a stationary ℓ' th-order Markov source $q_{\boldsymbol{z}}(\cdot)$. Then

$$p\left(\boldsymbol{x} : D\left(q_{\boldsymbol{z}}\|p\right) - \Delta(\boldsymbol{z}\|\boldsymbol{x}) > 3\mu\log\frac{1}{\delta}\right) \leq e^{-\frac{1}{2}\left(\frac{\mu^2}{\log^2 n}\right)n^{K'(\delta,\delta')}\left[1-o\left(n^{-\mu^2}\right)\right]}$$

for every $\boldsymbol{z} \in \mathcal{A}^n - \mathbb{B}$ such that $p(\boldsymbol{z}) > 0$, where B is a subset with the property:

$$q(\boldsymbol{z} \in B) \leq e^{-K''(\delta,D,\mu)\frac{n^{\mu'}}{\log n}+o\left(\frac{n^{\mu'}}{\log n}\right)}$$

and $K'\left(\delta,\delta'\right), \mu'$ and $K''(\delta, A, \mu)$ are same as defined earlier.

Thus, theorem 4 says that $\Delta(\boldsymbol{z}\|\boldsymbol{x})$ is a good estimate of $D\left(q_{\boldsymbol{z}}\|p\right)$ and this empirical divergence is used in the classification scheme given below.

3.3 Universal Classification of Individual Sequences

The main idea is that $\Delta(\boldsymbol{z}\|\boldsymbol{x})$ is a good estimate of $D\left(q_{\boldsymbol{z}}\|p\right)$ and hence can be used as a discriminating function. The classical hypothesis testing theory assumes the knowledge of the underlying source distribution, which is not the case in practice usually. One assumption that is made here is that the class of permissible classification rules consists only of those which can be implemented by Finite State Machines (FSMs). A two-class problem is considered and the result can be extended to the more general case too.
Let $\phi_i = \{\boldsymbol{x}_1^i, \ldots, \boldsymbol{x}_k^i\}$ denote two disjoint collection of vectors in $\mathcal{D}^n$ that are the

training sets for each source σ_i. The classification problem is the following: Given a test vector $\boldsymbol{y} \in \mathcal{D}^n$, decide whether $\boldsymbol{y} \in \sigma_i$, $i = 1, 2$ or whether $\boldsymbol{y} \in \sigma_1^{\complement} \cap \sigma_2^{\complement}$. A *classification rule* M partitions $\mathcal{D}^n$ into three disjoint sets: M_1, M_2 corresponding to the decision regions for σ_1, σ_2 and M_0 is the rejection region. The objective is to develop a classification rule M that is *consistent*, i.e. $\sigma_i \subseteq M_i$, $i = 1, 2$ and $M_1 \cap \sigma_2 = M_2 \cap \sigma_1 = \emptyset$. Note that the first goal instructs M_i to be as large as possible while the second restricts the size of M_i.
The permissible family $\mathcal{M}$ of classification rules M consists of classifiers that can be realized by FSM's followed by modulo-n counters. An $\mathcal{S}$ state classifier is a triple $C = (\mathcal{S}, g, \Omega)$, where $\mathcal{S}$ is a finite set of states with S elements, $g : \mathcal{D} \times \mathcal{S} \to \mathcal{S}$ is a next state function, and Ω is a partition of the space of empirical probability measures on $\mathcal{D} \times \mathcal{S}$, into three disjoint regions Ω_i, $i = 0, 1, 2$, depending on the training sets, where Ω_0 is the rejection region and Ω_1 and Ω_2 are acceptance regions of σ_1 and σ_2, respectively. When a test sequence $\boldsymbol{y} = (y_1, y_2 \cdots y_n)$ is fed into C, which in turn is initialized with $s_0 \in \mathcal{S}$, a state sequence $\boldsymbol{s} = (s_1, s_2 \cdots s_n), s_i \in \mathcal{S}$, is generated by

$$s_t = g\left(y_t, s_{t-1}\right), \quad t = 1, 2 \cdots n.$$

Let $n_y^g(a, s), a \in \mathcal{D}, s \in \mathcal{S}$, denote the joint count of $y_t = a$ and $s_{t-1} = s$ along the pair sequence (y, s), and let $q_y^g(a, s) = n_y^g(a, s)/n$ denote the empirical joint probability of a and s with respect to g. The empirical joint probability distribution $Q_y^g = \left\{q_y^g(a, s), a \in A, s \in S\right\}$ serves as test statistics for classifying y, that is, the classification rule $M = \{M_i\}_{i=0}^2$, associated with a FS classifier C is given by

$$M_i = \left\{y \in \mathcal{D}^n : Q_y^g \in \Omega_i\right\}, \quad i = 0, 1, 2$$

where the partition Ω (and hence also M) in turn depends on the training sequences via the empirical distributions $Q_{x_j^i}^g$ $j = 1, 2 \cdots k, i = 1, 2$. These empirical distributions are precomputed in the training phase.
Two sequences $\boldsymbol{y}$ and $\boldsymbol{z}$ are said to be of the same type with respect to g, or, of the same g-type if $Q_{\boldsymbol{y}}^g = Q_{\boldsymbol{z}}^g$. The g-type of $\boldsymbol{z}$ is defined as

$$T_g(\boldsymbol{z}) \triangleq \left\{\boldsymbol{y} \in \mathcal{D}^n : Q_{\boldsymbol{y}}^g = Q_{\boldsymbol{z}}^g\right\}$$

For a given next state function g, the smallest acceptance regions associated with a consistent FS classifier must include the entire g-type of each training sequence and hence the acceptance regions are

$$M_i = \bigcup_{j=1}^{k} T_g\left(x_j^i\right), \quad i = 1, 2$$

If these are intersecting sets, then these two sources cannot be distinguished by any S-state classifier. This motivates the following definition.

Definition 3.1. A pair of sources (σ_1, σ_2) is said to be (ϵ, S) separable relative to ϕ_1 and ϕ_2 if there exists a next-state function g of an S-state FSM such that for every $y \in \sigma_1$, every $z \in \sigma_2$, and some $x \in \phi_1 \cup \phi_2$,

$$\left| D\left(Q_{\boldsymbol{z}}^g \| Q_{\boldsymbol{x}}^g\right) - D\left(Q_{\boldsymbol{y}}^g \| Q_{\boldsymbol{x}}^g\right) \right| > \epsilon$$

where

$$D\left(Q_{\boldsymbol{z}}^g \| Q_{\boldsymbol{x}}^g\right) = \sum_{a \in \mathcal{D}, s \in \mathcal{S}} q_{\boldsymbol{z}}^g(a, s) \log \frac{q_{\boldsymbol{z}}^g(a \mid s)}{q_{\boldsymbol{x}}^g(a \mid s)}$$

Definition 3.2. A classification rule M that employs a next state function g is called *robust* w.r.t. a given training set (ϕ_1, ϕ_2), if for every $\nu > 0$, every $\boldsymbol{y} \in M_i, (i = 1, 2)$, and every n-sequence $\boldsymbol{y}'$ whose Hamming distance $d_H(\boldsymbol{y}, \boldsymbol{z})$ from $\boldsymbol{y}$ does not exceed $n \cdot \nu$, we have

$$\frac{1}{n} \left| \log Q_{\boldsymbol{x}}^g(\boldsymbol{y}) - \log Q_{\boldsymbol{x}}^g\left(\boldsymbol{y}'\right) \right| \leq \delta_n(\nu)$$

where $\delta_n(\nu) \to 0$ as $\nu \to 0$ uniformly for every $\boldsymbol{x} \in \phi_1 \cup \phi_2$.

This definition embodies the classifier's insensitivity to noise. This can also be associated with a vanishing memory for the classifier in the sense that, if the classifier did not have a vanishing memory, then there is a bigger likelihood of the classifier making errors based on small noises. An ℓ^{th}-order Markov machine has such a vanishing property as only errors from the immediate past ℓ outcomes can affect the current state.
Let $T^\ell(\boldsymbol{z})$ denote the type of the ℓ^{th}-order Markov type of $\boldsymbol{z}$, i.e., g is the next state function of an ℓ^{th}-order Markovian machine. Define the *ν-neighbourhood type* $T_\nu^\ell(\boldsymbol{z})$ as $\{\boldsymbol{y} : d_H(\boldsymbol{y}, \boldsymbol{z}' \leq n\nu$ for some $\boldsymbol{z}' \in T^\ell(\boldsymbol{z}))\}$. For some $\xi > 0$, let

$$U_i = \bigcup_{j=1}^{k} \bigcup_{\boldsymbol{x} \in T_\nu^L\left(\mathbf{x}_j^i\right)} \{\boldsymbol{z} : D\left(Q_{\boldsymbol{z}} \| Q_{\boldsymbol{x}}\right) \leq \xi\}$$

Define M^ℓ as

$$M_1^\ell = U_1 - U_2$$
$$M_2^\ell = U_2 - U_1.$$

The acceptance regions are subtracted from each other to prevent errors. It follows that if (σ_1, σ_2) are $(\epsilon, \mathcal{D}^\ell)$-separable w.r.t. an ℓ^{th}-order Markov machine, then there exists a sufficiently small positive number $\xi(\epsilon, \ell)$ s.t. for any $0 < \xi < \xi(\epsilon, \ell)$,

1. $M_1^\ell \supset \phi_1$ and $M_2^\ell \supset \phi_2$;

2. $M_1^\ell \cap M_2^\ell = \emptyset$.

Thus, if every ν-neighborhood Markov type of σ_i is represented in $\phi_i (i = 1, 2)$, M^ℓ is equivalent to M, in the sense of classifying sequences from σ_1 and σ_2 correctly. Thus it is enough to draw the training sets from M_1^ℓ, M_ℓ^2 and consider the reference sets to be these two instead of σ_1, σ_2. Define

$$\begin{aligned}
\hat{M}_1 =& \bigcup_{i=1}^{k} \left\{ \boldsymbol{z} : \Delta\left(\boldsymbol{z} \| \boldsymbol{x}_i^1\right) \leq \eta \right\} \\
& - \bigcup_{i=1}^{k} \left\{ \boldsymbol{z} : -\frac{1}{n} c(\boldsymbol{z}) \log c(\boldsymbol{z}) < \Delta\left(\boldsymbol{z} \| \boldsymbol{x}_i^2\right) \leq \eta \right\} \\
\hat{M}_2 =& \bigcup_{i=1}^{k} \left\{ \boldsymbol{z} : \Delta\left(\boldsymbol{z} \| \boldsymbol{x}_i^2\right) \leq \eta \right\} \\
& - \bigcup_{i=1}^{k} \left\{ \boldsymbol{z} : -\frac{1}{n} c(\boldsymbol{z}) \log c(\boldsymbol{z}) < \Delta\left(\boldsymbol{z} \| \boldsymbol{x}_i^1\right) \leq \eta \right\}
\end{aligned}$$

Theorem 5. *Let the pair (σ_1, σ_2) be $(\epsilon, \ell)-$ Markov separable relative to the training sets $\bar{\phi}_i = \left\{\bar{x}_j^i, j = 1 \cdots k\right\}$, $i = 1, 2$, which are sampled from M_1^ℓ and M_2^ℓ, respectively. Assume further that every $\bar{\pi}_j^i$ is associated with strictly positive empirical transition probabilities from every state $s \in \mathcal{S}^\ell$ to any letter $a \in \mathcal{D}$. Then, there exists a subset B of test sequences $\boldsymbol{z}$ and a subset B_z of training sequences, for every $\boldsymbol{z} \in \mathcal{D}^n - B$, with the following properties.*

(a) If $\boldsymbol{z} \in \mathcal{D}^n - B$ and $\boldsymbol{x}_j^i \in (\mathcal{D}^n - B_{\boldsymbol{z}}) \cap T^\ell\left(\bar{\boldsymbol{x}}_j^i\right)$ for every $j = 1 \cdots k, i = 1, 2$, then $\hat{M}$ classifies $\boldsymbol{z}$ in a manner identical to that of M^ℓ.

(b) The set $\boldsymbol{B}$ satisfies

$$\lim_{n \to \infty} \frac{\left|T^\ell(\boldsymbol{z}) \cap \mathbf{B}\right|}{\left|T^\ell(\boldsymbol{z})\right|} = 0$$

and for every $\boldsymbol{z} \in \mathcal{D}^n - B$,

$$\lim_{n \to \infty} \frac{\left|T^\ell\left(\bar{\boldsymbol{x}}_j^i\right) \cap B_{\boldsymbol{z}}\right|}{\left|T^\ell\left(\bar{\boldsymbol{x}}_j^i\right)\right|} = 0$$

Note that $\hat{M}$ is a classification scheme based on the empirical divergences and this theorem says that it is a good approximation of M^ℓ and is almost as efficient as M^ℓ in the sense that except for a vanishingly small fraction of $(\boldsymbol{z}, \phi_1, \phi_2)$, it achieves the same classification. It should be highlighted that $\hat{M}$ does not assume knowledge of ℓ while $\mathcal{M}^\ell$ does. Another advantage of $\hat{M}$ is that its memory size and computational complexity grows linearly with n [3], [4].

4 Anomaly Detection using LZ compression

This section surveys the work done in [5].

4.1 Introduction

The goal of anomaly detection is to identify instances which do not fit the ones from which the model has learned. If provided the probability distribution of the data $p(\cdot)$, then the optimal decision rule to check whether a test sequence $\boldsymbol{y}$ would be to compare $p(\boldsymbol{y})$ to a threshold, where the threshold is derived using the target false alarm probability. If the underlying distribution is not known, one approach would be to find the empirical distribution $\hat{p}(\cdot)$ and do the same, but this approach will not work always, especially if the original model consists of individual deterministic sequences with no known statistical model. Here, when no statistical model is known for the data, the LZ compression algorithm outlined earlier is used to come up with a universal probability assignment.

4.2 Statistical Model Creation

Using the tree-based framework mentioned earlier, in the beginning, we start with a root node and k leaf nodes where k is the size of the alphabet used in the code. As each new phrase is parsed, one traverses along the tree starting from the root and ends at a leaf node, at which point all the symbols from the alphabet are added as leaf nodes making it an internal node.
To define a statistical model, each node in the tree except the root node maintains a traversal counter, each leaf-node's counter is set to 1 and each internal node's counter is equal to the sum of its immediate children's counters. To assign probabilities, as all leaf-nodes' counters are set to 1, they are assumed uniformly distributed with a probability $\frac{1}{i}$, where i is the total number of leaf-nodes. Each internal node's probability is defined as the sum of its immediate children's probabilities, which also equals the ratio between its counter and current i. The probability of an edge is defined by dividing the nodes' probabilities. The probability of a phrase $P_i \in \mathcal{D}^*$ is defined as the product of the probabilities of the edges along the path defined by the symbols of P_i. If while traversing a sequence S a leaf-node is reached before all the symbols of S are finished then the traversal returns to the root and continues until all the symbols of that sequence are consumed. For a given test sequence, this statistical model is now used to calculate the probability of that test sequence.

4.3 Classification Model

The classification model is divided into a learning phase and a testing phase. During the learning phase, the above statistical model is created using a given training set of discrete sequences over a finite alphabet $\mathcal{S}$ as described earlier. During the testing phase, the probability of each test sequence T_j from a testing set T is estimated based on the statistical model constructed in the learning phase and classified according to a predefined threshold T_r, i.e. if $\hat{p}(T_j) > T_r$, then the sequence is classified as 'normal' and as 'anomalous' otherwise.
The algorithm is evaluated based on the false alarm and hit detection ratios and the ROC curve is generated with respect to a set of thresholds in the range $[\min_j \hat{p}(T_j), \max_j \hat{p}(T_j]$. The authors applied this classification model to Botnet identification in [5].

4.4 Attacking strategies

For an attacker to "fool" the model, it would have to generate the same statistical model that was built in the learning phase and generate sequences according to that. Let the alphabet $\mathcal{D} = \{0, 1\}$ and the length of the sequences be n. Assume the actual sequences are generated i.i.d. according to distribution P while the attacker generates sequences i.i.d. according to distribution Q.
The attacker could try a trial and error strategy and, i.e. randomly generate sequences according to Q. For a given sequence $X = (x_1, \ldots, x_n)$, the type P_X is defined as the empirical distribution and the type class $T(P_X)$ is the set of sequences of length n and type P_X. The probability of type class T(P) under Q^n, i.e. $Q^n(T(P)) \leq 2^{-nD(P\|Q)}$. Thus, as long as the attacker does not know P and uses $Q \neq P$, then the probability of this happening decays exponentially.
Another strategy that the attacker could try would be to obtain a legitimate sequence generated according to P and send it again and again. This method would fail too as as this would result in a stream whose distribution would be far from P and would suffer the same way as the previous strategy.
Another approach would be to generate new sequences based on the above available sequence. Given the above legitimate sequence, considered as a training sequence and denoted by X^m, where m is the length of the sequence, and a string of k purely random bits U^k, which are independent of X^m, the objective is to generate new sequence(s) of the same length or shorter, i.e. of length ($n \leq m$), denoted as Y^n, with the same probability distribution of X^m but with minimum statistical dependency between these sequences. To achieve this, a deterministic function $\phi(\cdot)$, independent of the unknown source P is employed, such that $Y^n = \phi\left(X^m, U^k\right)$, and the mutual information $I\left(X^m; Y^n\right)$ is minimum so that there is weak dependence between the given training sequence and the result output sequences.

However, [6] has shown that for this to happen, the input length m must to be as large as possible, and the number of random bits k needed to guarantee low dependency between X^m and Y^n grows linearly with the output length n. In the main problem setting of this paper, where there is a Botmaster, i.e., the attacker that has corrupted a network of nodes called Bots and controls them, this is computationally infeasible. The Botmaster will have to generate new sequences according to the above model and updates its Bots with these sequences in order to carry out the attack. For the case where $n < m$, the Botmaster must constantly produce and maintain these k random bits (to guarantee low statistical dependency), resulting in high complexity mechanism, and on the other hand, for the case where $n = m$, the Botmaster would have to generate large sequences, which makes it difficult to communicate these sequences to the Bots. It is not possible for the Bots to generate these sequences either as they are usually simple devices that do not have the computational power required to do so and also because of the requirement of a coordinated attack.

5 Conclusion

In this report, a brief summary of techiques based on Lempel-Ziv compression for universal classification and anomaly detection to determine Botnet attacks in communication networks has been discussed.

References

[1] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006, ISBN: 0471241954.

[2] J. Ziv and N. Merhav, "A Measure of Relative Entropy Between Individual Sequences with Application to Universal Classification," en, *IEEE Transactions on Information Theory*, vol. 39, no. 4, p. 10, 1993.

[3] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978. DOI: 10.1109/TIT.1978.1055934.

[4] M. Rodeh, V. R. Pratt, and S. Even, "Linear algorithm for data compression via string matching," *J. ACM*, vol. 28, no. 1, 16–24, 1981, ISSN: 0004-5411. DOI: 10.1145/322234.322237. [Online]. Available: https://doi.org/10.1145/322234.322237.

[5] S. Siboni and A. Cohen, "Universal Anomaly Detection: Algorithms and Applications," en, *arXiv:1508.03687 [cs]*, Aug. 2015, arXiv: 1508.03687. [Online]. Available: http://arxiv.org/abs/1508.03687 (visited on 04/09/2022).

[6] N. Merhav and M. Weinberger, "On universal simulation of information sources using training data," *IEEE Transactions on Information Theory*, vol. 50, no. 1, pp. 5–20, 2004. DOI: 10.1109/TIT.2003.821993.