Comparison of Supervised Learning Techniques for Non-Technical Loss Detection in Power Utility

Keem Siah Yap1,*, Sieh Kiong Tiong2, Jawad Nagi3, Johnny S. P. Koh4, Farrukh Nagi5

Abstract – Non technical losses (NTLs) originating from electricity theft and other customer malfeasances are a problem in the electricity supply industry. In recent times, electricity consumer dishonesty has become a universal problem faced by all power utilities. Previous work carried out for NTL detection resulted in a Support Vector Machine (SVM) based detection framework. The present study performs a comparative study for NTL detection using supervised machine learning techniques such as the: Back-Propagation Neural Network (BPNN) and Online-sequential Extreme Learning Machine (OS-ELM). Model testing is performed using historical customer consumption data for three towns within peninsular Malaysia. The detection hit-rate of all compared models is obtained from TNB Distribution (TNBD) Sdn. Bhd. for onsite customer inspection. Experimental results obtained indicate that the BPNN detection model achieves the lowest average detection hit-rate of 36.07%, while the OS-ELM model obtains a slightly higher average detection hit-rate of 51.38%. The previously proposed SVM-based NTL detection model outperforms the BPNN and OS-ELM by far with the highest average detection hit-rate of 60.75%. This indicates that the use of a SVM-based soft-margin approach results in a better generalization performance for the application of NTL detection as compared to the BPNN and OS-ELM schemes.

Keywords: Non-Technical Loss, Electricity Theft, Back-Propagation Neural Network, Extreme Learning Machine, Support Vector Machine

I. Introduction

Non-technical losses (NTLs) originating from electricity theft and other customer malfeasances comprise one of the most important concerns of any power utility [1]. Such losses occur due to meter tampering, meter malfunction, illegal connections, billing irregularities and unpaid bills [2].

The problem of NTLs is not only faced by the least developed countries in the Asian and African regions, but also by developed countries such as the United States of America and the United Kingdom [3]. Specifically, high rates of NTL activities have been reported in the majority of developing countries in the Association of South East Asian Nations (ASEAN) group, which include Malaysia, Indonesia, Thailand and Vietnam [4]. As an example, in the United States NTLs have been estimated to account between 0.5% to 3.5% of the total annual revenue [2], which is relatively low when compared to the NTLs faced by utilities in developing countries such as Bangladesh [3], India [4], Pakistan [5] and Lebanon [6], where an average of between 20% to 30% of NTLs have been observed. Nonetheless, in 1998, the revenue loss by power utilities in United States was estimated between USD 1 billion to USD 10 billion, given that all utility companies in the United States had an annual gross revenue of USD 280 billion [2].

In deregulated markets, the knowledge of electricity customers provides an understanding of their consumption behaviour, which has recently become important in the electricity supply industry. With this knowledge, electricity providers are able to develop new marketing strategies and offer services based on customer demand. One of the most common methods used in acquiring knowledge of customers’ behaviour is load profiling [7], which is defined as “the pattern of electricity consumption of a customer or group of customers over a period”.

Load profiling has been used for many years by power utilities for tariff formulation, system planning, and devising marketing strategies [8].

As a common practice, power utility companies record historical customer data, such as contractual details, billing procedures, and consumption records in various customer databases to support their billing activity [9]. However, such information as it presently exists is often too complex to allow the human mind to formulate efficient and strategic decisions or draw effective conclusions. In addition, this information is often inaccessible and extremely time consuming to retrieve, due to the problems associated with archived data in complex database systems [9], [10]. Due to the problem associated with NTLs in electric utilities, various
methods for efficient management of NTLs [11] and protecting revenue in the electricity distribution industry [12] have been proposed. The most effective method currently to reduce NTLs and commercial losses up to date is by using smart and intelligent electronic meters [13], which make fraudulent activities more difficult, and easy to detect. However, the cost of such meters comes at an expensive price. Therefore, these types of meters are not currently feasible to be used by power utilities throughout the entire low voltage distribution network, i.e., in residential and commercial sectors.

In recent years, several data mining studies on fraud identification and detection in the electricity distribution industry have been researched, including: Rough Sets [15], Statistical methods [16], [17], Decision Trees [18], [19], Artificial Neural Networks (ANNs) [20], Support Vector Machines (SVM) [21]–[23], Extreme Learning Machine (ELM) [1], Statistical-based Outlier detection [24], Knowledge Discovery in Databases (KDDs) [10], [25] and Wavelet-based feature extraction with multiple classifiers [26]. In addition to this, data mining techniques have also been used in the other types of businesses including: telecommunication [27], insurance [28], risk management [29], and credit card transaction [30]. Most of these studies have used data mining techniques for means of detection and prediction of fraud activities.

The present study performs a comparative study for NTL detection using supervised machine learning techniques such as the: Back-Propagation Neural Network (BPNN) and Online-sequential Extreme Learning Machine (OS-ELM). This study is performed in comparison to the SVM-based NTL detection models previously proposed by the authors in [21]. Model testing is performed using historical customer consumption data from for three towns within peninsular Malaysia. The detection hit-rate of all compared models is obtained from the pilot testing results obtained for all three detection techniques. In Section IV, the development of the fraud detection model (FDM) for all three machine learning techniques were used to implement NTL detection models for the purpose of a comparative study. The following machine learning algorithms were presented in Section V.

II. Supervised Machine Learning Techniques

Classification techniques, also known as supervised machine learning techniques were used to implement NTL detection models for the purpose of a comparative study. The following machine learning algorithms were implemented as separate classification engines for NTL detection.

II.1. Back-propagation Neural Network (BPNN)

Back-propagation, which is also known as the generalized delta rule [31], [32] is one of the most popular and widely investigated methods for training ANNs. The most common network topology is multiple layers with connections only between nodes in neighbouring layers. There are no connections between nodes located in a common layer. Information is passed in one direction through the network; starting at one side of the network and moving through successive layers [33].

There are three distinct types of layers: the input layer, the hidden layer(s) and the output layer. The connections between the neurons of adjacent layers relay the output signals from one layer to the next. The input layer receives the input information and distributes the information to the next processing layer (first hidden layer). The number of the neurons in the input layer equals to the dimension of the input vector (number of features). The hidden and output layers process the incoming signals by amplifying, attenuating or inhibiting the signals through weighting factors. Except for the input layer neurons, the network input to each neuron is the sum of the weighted outputs of the neurons in the previous layer. The number of neurons in the output layer is determined by the number of classes under investigation. The number of hidden layers and the number of neurons in each hidden layer depend on specific application.

All nodes in the network, except the input nodes, perform the same two functions; collecting the activation of nodes in the previous layer and setting output activation. The input nodes activations are determined by the input data. The collection function [32] used in this study is:

\[ \text{net}_{pi} = \sum_j w_{ij} a_{pj} + \text{bias}_i \]  

(1)

where variable \( w_{ij} \) represents the connection strength for the current node \( i \) to a node \( j \) in the previous layer, \( a_{pj} \) is the activation of node \( j \) for pattern \( p \) plus a node bias, \( \text{bias}_i \), which can be considered a connection to a node which is always at full activation. The result of the collection function, \( \text{net}_{pi} \), is passed to the output section which determines the node’s output activation, \( a_{pi} \). The output function is a nonlinear function which allows a network to solve problems that a linear network cannot solve [33], [34]. In this study the sigmoid function [33] given in (2) is used to determine the output state as following.

\[ a_{pj} = \frac{1}{1 + e^{-\text{net}_{pi}}} \]  

(2)

A BPNN is trained by example. A set of representative input and output patterns is selected. As each input pattern is presented, the connections of the
network are adjusted so that the activation of the output nodes more closely matches the desired output pattern. All the patterns are repeatedly presented to the network until the network “learns” the patterns. The foundation of the back-propagation learning algorithm is the nonlinear optimization technique of gradient descent [35] on the sum of the squared differences between the activation, \( O_{pi} \), of the nodes in the output layer and the desired output \( t_{pi} \). The objective is to minimize the function \( E \), i.e.:

\[
E = \sum_{p} \sum_{i} (t_{pi} - O_{pi})^2
\]

where \( p \) indexes the training patterns and \( i \) indexes the output nodes of the network. By adjusting the network connection strengths, \( w_{ij} \), the above function is minimized and the network learns the patterns. Application of the gradient descent method [34] yields the following iterative weight update rule:

\[
\Delta w_{ij}(n+1) = \varepsilon (y_{pi}a_{pi}) + \alpha \Delta w_{ij}(n)
\]

where \( w_{ij} \) is the connection strength from node \( j \) to node \( i \), \( y_{pi} \) is the node \( i \) error for pattern \( p \), \( a_{pi} \) is the activation of node \( j \) for pattern \( p \), and \( \varepsilon \) is a parameter known as the learning rate and the parameter, \( \alpha \) controls the momentum term. The node error, \( y_{pi} \), for an output node is then given as:

\[
y_{pi} = (t_{pi} - a_{pi})a_{pi}(1 - a_{pi})
\]

where the first term is the error between an output node’s activation and the target pattern. The other terms in (5) are the result of the derivative of the activation function. The error at an arbitrary hidden node \( i \) is:

\[
y_{pi} = a_{pi}(1 - a_{pi}) \sum_{k} y_{pk}w_{ki}
\]

The summation in (6) collects the errors from the layer below and the other terms are the derivative of the activation function. For details of the back-propagation learning algorithm including derivation of the equations see [32]. The BPNN has been selected as one of the classification engines used in this NTL analysis comparative study.

II.2. Extreme Learning Machine (ELM)

The Extreme Learning Machine (ELM) is one of the most recent machine learning algorithm proposed by Huang in 2006 [36] for Single Hidden-layer Feedforward Neural Networks (SLFNs). The ELM was devised in order to produce superior performance [37]–[39]. The learning speed of feed forward neural networks has been claimed to be slower due to the fact that a slow gradient-based iterative learning algorithm is used extensively to train neural networks [40]. Unlike many other popular learning algorithms, little human involvement is required in ELM, except the number of the hidden neurons (which is insensitive to ELM), no other parameters need to be tuned manually by users.

ELM is a general learning algorithm for SLFNs that works effectively for function approximations, classifications, and online prediction problems. Moreover, it can generally work well for a variety of types of applications. Usually, a SLFN has three kinds of input parameters: (i) the input weight \( w_{i} \), (ii) the hidden neuron biases \( b_{i} \), and (iii) the output weight \( \beta_{i} \). While conventional learning algorithms of SLFNs have to tune these three parameters, ELM randomly generates the input weight \( w_{i} \), the hidden neuron biases \( b_{i} \) and then analytically calculates the output weight \( \beta_{i} \). No further learning is required for SLFNs trained using ELM.

Phase-1. Approximation Problem of SLFNs: Given \( N \) arbitrary distinct samples \( (x_{1}, t_{1}), (x_{2}, t_{2}), \ldots, (x_{N}, t_{N}) \in \mathbb{R}^{n} \) and \( t_{i} = [t_{i1}, t_{i2}, \ldots, t_{iM}] \in \mathbb{R}^{m} \), a SLFN with \( N \) hidden neurons and activation function \( g(x) \) can be mathematically modelled as:

\[
\sum_{i=1}^{N} \beta_{i}g(w_{i} \cdot x_{j} + b_{i}) = o_{j}, \quad j = 1,2,\ldots,N
\]

where \( w_{i} \) is the weight vector connecting the input neurons and the \( i^{th} \) hidden neuron \( \beta_{i} \) is the weight vector connecting the \( i^{th} \) hidden neuron and the output neurons and \( b_{i} \) is the threshold of the \( i^{th} \) hidden neuron.

Here \( w_{i} \cdot x_{j} \) denotes the inner product of \( w_{i} \) and \( x_{j} \). If the SLFN can approximate these \( N \) samples with zero error, then \( \sum_{i=1}^{N} |o_{j} - t_{ij}| = 0 \), follows; i.e., there exists \( \beta_{i}, w_{i}, b_{i} \) such that \( \sum_{i=1}^{N} |\beta_{i}g(w_{i} \cdot x_{j} + b_{i}) - t_{ij}| = 0, \quad j = 1,2,\ldots,N \). The above \( N \) equations can be written compactly as \( H\beta = T \), where:

\[
H = \begin{bmatrix}
g(w_{1} \cdot x_{1} + b_{1}) & \cdots & g(w_{1} \cdot x_{N} + b_{1}) \\
g(w_{2} \cdot x_{1} + b_{2}) & \cdots & g(w_{2} \cdot x_{N} + b_{2}) \\
\cdots & \cdots & \cdots \\
g(w_{N} \cdot x_{1} + b_{N}) & \cdots & g(w_{N} \cdot x_{N} + b_{N})
\end{bmatrix}_{N \times N}
\]

\[
T = \begin{bmatrix}
t_{11} \\
t_{12} \\
\vdots \\
t_{1M}
\end{bmatrix}_{N \times M}
\]

As specified by Huang and Babri in [41], \( H \) is called the hidden layer output matrix of the neural network, with the \( i^{th} \) column of \( H \) being the \( i^{th} \) hidden neuron output with respect to inputs \( x_{1}, x_{2}, \ldots, x_{N} \).

Phase 2. ELM Learning Algorithm for SLFNs: Based on the previous work of Huang in [42], matrix \( H \) is square and invertible only if the number of hidden neurons is equal to the number of distinct training samples \( \bar{N} = N \), indicating that SLFNs can approximate these training samples with zero error. In most cases, the
number of hidden neurons is much lower than the number of distinct training samples, \( N \ll N \). \( H \) is a non-square matrix and there may not exist \( w_i, b_i b_i(i = 1, ..., N) \) such that \( H \beta = T \). Thus one specific set of \( \tilde{w}_i, \tilde{b}_i (i = 1, ..., \tilde{N}) \) needs to be found such that:

\[
\|H(\tilde{w}_j, ..., \tilde{w}_\tilde{N}, \tilde{b}_j, ..., \tilde{b}_\tilde{N})\beta - T\| = \min_{w_i, b_i \beta} \|H(w_j, ..., w_N, b_j, ..., b_N)\beta - T\|
\]  

(10)

which is equivalent to minimizing the following cost function:

\[
E = \sum_{j=1}^{N} \left( \sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i) - t_j \right)^2
\]  

(11)

Huang in [39],[42] indicated that the hidden neuron parameters do not need to be tuned, as the matrix \( H \) indeed converts the data from non-linear separable cases to high dimensional linear separable cases.

However, Huang in [40] showed that the input weights and hidden neurons or kernel parameters do not necessarily need to be tuned and can be randomly selected and then fixed. In fact, the parameters of the hidden neurons are not only independent of each other but also independent of the training data. Thus, for fixed input weights and the hidden layer biases or kernel parameters training a SLFN is equivalent to finding a least squares solution \( \beta \) for the linear system, \( H \beta = T \). The unique smallest norm least squares solution of the above linear system is \( \hat{\beta} = H^\dagger T \), where \( H^\dagger = (H^T H)^{-1}H^T \) is the Moore-Penrose generalized inverse of the hidden layer output matrix \( H \). As analyzed by Huang [42], this method may tend to reach a good generalization performance. The ELM algorithm consists of only three steps and can be summarized as follows:

**ELM Algorithm [40]:**

Given a training set \( \mathbf{X} = \{(x_i, t_i) | x_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m, i = 1, ..., N\} \), activation function \( g(x) \), and the hidden neuron number \( N \).

Step 1) Assign random input weight \( w_i \) and bias \( b_i, i = 1, ..., N \).

Step 2) Calculate the hidden layer output matrix \( H \).

Step 3) Calculate the output weight \( \beta \), \( \beta = H^\dagger T \) where \( H, \beta \) and \( T \) are defined in eq. (8) and (9).

The ELM algorithm can accommodate nonlinear activation functions and kernel functions. Furthermore, it can avoid difficulties like the: stopping criteria, learning rate, learning epochs and local minima very well unlike other tuning or adjustment methods [42]. ELM provides faster learning capability compared with other alternatives. In [40], the performance of the ELM algorithm was compared with other algorithms, including SVM and conventional back-propagation. The results showed that ELM has an outstanding performance in relative terms.

**II.3. Online Sequential Extreme Learning Machine (OS-ELM)**

In order to handle online applications, the online variant of ELM referred to as Online-sequential Extreme Learning Machine (OS-ELM) was introduced by Liang et al. in [43]. The OS-ELM was originally developed for SLFNs with the additive or radial basis function (RBF) hidden nodes in a unified framework. The OS-ELM can learn the training data that accommodate its real application in the industry, chunk by chunk. According to [43] OS-ELM also works for other types of SLFNs which may not be neural networks.

In the derivation of sequential ELM, only the specific matrix \( H \) is considered, where the rank of \( H \) is equal to the number of hidden neurons: \( \text{rank}(H) = N \). Under this condition, the following implementation of the pseudo inverse of \( H \) is easily derived and given by \( H^\dagger = (H^T H)^{-1}H^T \), which is often called the left pseudo inverse of \( H \), given from the fact that \( H^T H = I_N \). The corresponding estimation of \( \beta \) is given as:

\[
\hat{\beta} = (H^T H)^{-1}H^T T
\]  

(12)

which is called the least-squares solution to \( H \beta = T \). The sequential implementation of (12) can be derived and referred as the recursive least-squares (RLS) algorithm.

In [44], Liang et al. reviewed the ELM and OS-ELM approaches. The OS-ELM consists of two main phases, specifically: (1) the boosting phase or initialization phase, and (2) the sequential learning phase. The boosting phase trains the SLFNs using the primitive ELM method with some batch of training data in the initialization stage. This data is discarded once the process is complete. After that, the OS-ELM will learn the training data chunk by chunk. Subsequently, all the training data will be discarded once the learning process is complete. The OS-ELM algorithm can be summarized as follows:

**OS-ELM Algorithm [43]:**

Given an activation function \( g \) or RBF kernel \( \phi \) and hidden neuron or RBF kernel number \( N \) for a specific application, the two following undertakings are:

Step 1) **Boosting Phase:** Given a small initial training set, \( \mathbf{X} = \{(x_i, t_i) | x_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m, i = 1,2, \ldots, N\} \), the intention is to boost the learning algorithm by means of the following procedure:

(a) Assign random input weight \( w_i \) and bias \( b_i \) or centre \( \mu_i \) and impact width \( \sigma_i, i = 1,2, \ldots, N \).

(b) Calculate the initial hidden-layer output matrix, \( H_0 = [h_1, ..., h_N]^T \), where \( h_i = [g(w_i \cdot x_1 + b_1), ..., g(w_i \cdot x_N + b_N)]^T, i = 1,2, \ldots, N \).

(c) Estimate the initial output weight, \( \beta^{(0)} = M_0 H_0^T T_0 \), where \( M_0 = (H_0^T H_0)^{-1} \) and \( T_0 = [t_1, \ldots, t_N]^T \).

(d) Set \( k = 0 \).
Step 2) Sequential-Learning Phase: For each further incoming observation \((x_i, t_i)\), where \(x_i \in \mathbb{R}^n\), \(t_i \in \mathbb{R}^m\) and \(i = 1, \ldots, n\), do the followings. 
(a) Calculate the hidden-layer output vector, using
\[
h_{(k+1)} = g(w_1 \cdot x_i + b_1), \ldots, g(w_R \cdot x_i + b_R) \quad (13)
\]
(b) Calculate the latest output weight \(\beta^{(k+1)}\) based on a recursive least-square (RLS) algorithm, i.e.:
\[
M_{k+1} = M_k - \frac{M_k h_{(k+1)} h_{(k+1)}^T M_k}{1 + M_k h_{(k+1)} h_{(k+1)}^T M_k} \quad (14)
\]
\[
\beta^{(k+1)} = \beta^{(k)} + M_{k+1} h_{(k+1)} (t_i - h_{(k+1)} \beta^{(k)}) \quad (15)
\]
(c) Set \(k = k + 1\).

The ELM as OS-ELM have been selected as two of the main classification engines for the NTL analysis comparative study presented in this paper. In Section IV below, their performance will be compared with that of the BPNN and SVM approaches for the purpose of validation.

II.4. Support Vector Machines (SVM)

Support vector machines (SVM) were introduced by Vapnik in the late 1960s. The SVM, based on the foundation of statistical learning theory, is a general classification method and its theoretical foundation is described in [45] and [46]. SVM have recently emerged as one of the most popular and useful techniques for data classification.

The main objective of SVM is to produce a model that predicts the target value of data instances in the testing set in which only attributes are given. The classification goal in SVM in a binary classification problem, is to separate the two classes by means of a function devised from available data and thereby to produce a classifier that will work well on further unseen data. For the sake of completeness, the fundamentals of SVM are reviewed briefly here. More details regarding the SVM implementation for NTL detection can be found from previous work carried out in Section III of [21].

The simplest form of SVM classification is the maximal margin classifier. It is used to solve the most basic classification problem, namely the case of a binary classification with linear separable training data. Consider the training data, i.e.:
\[
\{(x_1, y_1), \ldots, (x_p, y_p)\}, \quad x \in \mathbb{R}^n, \quad y \in \{-1, +1\} \quad (16)
\]
with the assumption that they are linearly separable. That is, there exists a hyperplane, such that:
\[
(w, x_i) + b = 0 \quad (17)
\]
that satisfies the following constraints: for every \((x_i, y_i)\), \(i = 1, \ldots, l\), \(y_i (w, x_i) + b > 0\), where \(w, x\) is the dot product between \(w\) and \(x\).

The aim of the maximal margin classifier is to find the hyperplane with the largest margin, i.e., the maximal hyperplane. This problem can be represented as:
\[
\text{Minimize} \quad \frac{||w||^2}{2} \quad (18)
\]
subject to \(y_i ((w, x) + b) \geq 1, \quad i = 1, \ldots, l \quad (19)

In real-world problems, the training data is not always linearly separable. In order to handle the nonlinearly separable cases some slack variables have been introduced into SVM so as to tolerate some training errors, with the influence of the noise in the training data thereby decreased. The SVM classifier with slack variables as defined in eq. (8) of [21] is referred to as a soft-margin classifier. The radial basis kernel function used in this paper is as following:
\[
K(x, y) = \exp \left( -\frac{||x - y||^2}{2\sigma^2} \right) \quad (20)
\]

This paper employs LIBSVM [47], a library for SVM, as the core of a C-SVM classifier and conducts two-class (binary) classifications using the soft-margin based approach. The C-SVM model parameters \((C, \gamma)\) are optimized using the Grid Search method proposed by Hsu et al. in [48].

III. Implementation

The NTL detection models implemented using the proposed classification techniques in Section II are identical with NTL detection framework in Figure 1 of [21], where the C-SVM is substituted using the BPNN, ELM and OS-ELM. For the purpose of model implementation raw historical consumption data of three cities in peninsular Malaysia is acquired from TNB Distribution (TNBD) Sdn. Bhd. Data pre-processing which includes: customer filtering and selection, feature selection, feature extraction and normalization is performed as indicated in Figure 2 of [21], which corresponds to the previous work carried out by the authors. After data pre-processing, 25 normalized feature attributes as indicated by eq. (21) in [21] are obtained for training and testing/validation using the proposed classification techniques. The BPNN and ELM classification techniques are used to benchmark the performance of the proposed SVM-based NTL detection model previously proposed by the authors in [21].

The BPNN training uses a different approach in calculation as it minimizes the empirical or training error. However, SVM in contrast, minimize the structural risk. Conventional neural network algorithms such as Back-propagation (BP) require tuning of a number of network parameters. However, as ELM and OS-ELM are based on SLFNs, they randomly generate the input weight and the hidden neuron biases of the SLFN and use them to calculate the output weight without requiring further
learning. As neural networks have similar structure to that of SVM, therefore, different variants of neural networks, such as ELM and OS-ELM are in this comparative study for the implementation of a robust NTL detection model.

For the purpose of conducting binary classifications, training is performed using 436 clean data samples, which are identified using manual inspection of customer load and consumption patterns. Amongst the total 436 training samples, 53 are samples acquired from confirmed fraud meters and the remaining 383 samples are of normal meters, as indicted in [21]. The Grid-Search method proposed by Hsu et al. in [48] is used to determine the optimal SVM model parameters \((C, \gamma)\).

For the implementation of the BPNN-based NTL detection model, the key problem is to determine the network structure in the number of hidden layers and the number of hidden neurons in each layer. If the architecture is too small, the network may not have sufficient degrees of freedom to correctly learn the carbonation depth.

On the other hand, if the network is too large, then it may not converge during training, or it may overfitted the data and memorize the depth history rather than generalize it [49]. Thus, using a costly trial-and-error approach a suitable BPNN architecture is determined.

To model a BPNN, initially a three-layer neural network is chosen, since it is essential to determine the optimal number of hidden layers and hidden layer neurons. After trial and error a 25-15-20-1 neuron network is adopted to develop the BPNN model. The four-layer BPNN architecture consists of a one input layer, two hidden layers and one output layer. The input layer consist of 25 input neurons, which corresponds to 25 feature attributes, and the output layer of the BPNN consist of only one neuron which gives an output of 0 for good customers and 1 for fraud or suspicious customers. The output of the neurons in the hidden layers is calculated using the activation function defined in eq. (2).

Implementation of the ELM/OS-ELM based NTL detection models, utilized the radial basis function (RBF) activation function. Using RBF nodes in the ELM and OS-ELM, the centres and widths of the nodes are randomly generated and fixed, and based on this, the output weights are analytically determined [50].

As the ELM and OS-ELM are based on the architecture of SLFNs, the only network parameter to be determined is the optimal number of hidden layer neurons. The technique to determine the optimal size of the hidden layer neurons for ELM/OS-ELM is suggested by Huang et al. in [40], which indicates that the number of hidden layer neurons varies in the range from 20 to 200. With an initial starting size of 20 neurons initially, the number of neurons is increased with a step of 20, until 200 neurons are reached. Based on the output performance of the network, the optimal number of neurons is determined. By applying this technique, the optimum architecture of the ELM-based NTL detection model is determined with 30 hidden layer neurons, while for the OS-ELM-based NTL model 50 hidden neurons are selected.

## IV. Experimental Results and Discussion

All technique, BPNN, OS-ELM and SVM are evaluated using the same data pre-processing framework outlined in Fig. 1. Experimental results obtained from mentioned models are tabulated in Table I.

As shown in Table I, the highest training accuracy is achieved by the BPNN and the highest inspection hit-rate is obtained by the SVM model. The training accuracy is a measure of the memorization capability of the classification system, however, the inspection hit-rate is a more crucial and important result in this research study, since it reflects the actual detection accuracy of the model for the customers tested/validated. The number of neurons in the hidden layers of the BPNN and OS-ELM are tuned using Cross Validation. In this research study, 10-fold Cross Validation is chosen, since there are many training data present that can be divided into subsets.

The reason for using Cross Validation is to ensure the accuracy of the results do not over fit the training data.

Table II summarizes the comparison results for Table I by computing the average training accuracy and the average inspection hit-rate for the tested customers in the three cities in the state of Kelantan.

As illustrated by Table II, the BPNN achieves the highest training accuracy and the lowest inspection hit-rate. In contrast, the OS-ELM obtains a slightly higher inspection hit-rate as compared to the BPNN. With respect to the training accuracy, the proposed SVM model obtains the lowest training accuracy. However, in terms of the inspection hit-rate, the proposed SVM model outperforms the other two models by far, with an average inspection hit-rate of 40.75%. The increase in the inspection hit-rate of the proposed model is 14.5% and 9.5% as compared to the BPNN and OS-ELM respectively. As indicated from Table II, it is observed that all three models have a relatively good memorization capability, i.e., high learning rates (training accuracies).

However the BPNN obtained the highest training accuracy, followed by the OS-ELM and the proposed SVM. The main reason for the higher training accuracy of the BPNN is, as the BPNN model is a problem of non-linearity optimization using a gradient descent approach, the major peculiarity that impacts its performance is the presence of the local minimum. The main drawback of the BPNN is that, it gets trapped into the local minimum. Figure 2 illustrates the phenomenon of the local minimum, in which case the training is to be optimized to achieve a global optimum solution. The local minimum is determined with respect to the Mean Square Error (MSE), generally referred to as error in BPNNs sometimes alongside the weights, \(w_i\) of the network, as indicated by Fig. 2. The main objective of a neural network training process is to obtain a global optimum solution.
Fig. 1. The flowchart of the proposed SVM for the classification of fraud and normal customers

Fig. 2. The phenomenon of “local minimum” in a BPNN
However, in a BPNN to get the overall minimum answers of the error function, the network extreme corrects itself slowly along the local improved way and eventually ends up in obtaining the local optimization answers.

The reason for this is, as the BP algorithm is based on the gradient optimization method, the network tends to descend slowly with a low learning speed, and when a flat section (roof) appears for a long time the BPNN training ends at that instance, resulting in locally optimized answers. Proof of the local minimum phenomenon is shown in Fig. 2.

It is also observed from the experimental results in Table II that the BPNN achieves the lowest inspection hit-rate of 26.27%. This indicates that the BPNN has a lower generalization capability as compared to the OS-ELM and SVM. The main reason for this is because, as it is difficult to obtain the best network structure of the BPNN using a trial and error procedure, the optimum solution cannot be easily found. In terms of more logical reasoning it can be said that, as the BPNN tries to minimize the training error, it results in over fitting the training data. However, this does not mean that the BPNN is not a good algorithm for classification, but as there are many noisy training data present, the BPNN is not a suitable choice to be used in this application.

The OS-ELM used for comparison purposes in this research study overcomes many issues in traditional gradient algorithms like BPNN, such as: stopping criterion, learning rate, number of epochs and local minima. The experimental results obtained in Table II reveal that the training accuracy of the OS-ELM is slightly higher and the inspection hit-rate of the OS-ELM is significantly lower as compared to the proposed SVM model. The reason for the higher training accuracy of the OS-ELM is because, the OS-ELM iteratively fine tunes the network’s input weights and biases using finite samples of the training data, which yields in a higher memorization capability. For the RBF activation function, the OS-ELM randomly initializes hidden neuron parameters (input weight vectors and neuron biases for additive hidden neurons and centres and impact factors for RBF hidden neurons) and iteratively computes the output weight vector.

Furthermore, for the OS-ELM, it was observed that if the order of the training samples is switched or changed, the resulting training accuracy of the OS-ELM also changes. Therefore, in order to cater this situation, during training the training accuracy was computed over an average of 100 trials, where on each trial training samples were ordered randomly. Another noticeable observation concluded from the OS-ELM was that, with the increase of neurons, the OS-ELM achieved a better performance, while remaining stable for a wide range of neuron sizes. However, with an increase in the number of hidden layer neurons, the training time of the OS-ELM decreases.

The inspection hit-rate achieved by the OS-ELM during testing and validation was 31.16%, which is significantly lower as compared to the proposed SVM model, with an inspection hit-rate of 40.75%. There are a few reasons which can be stated in order to contribute to the lower generalization capability of the OS-ELM. The first reason being that, the assignment of the initial weights in the OS-ELM is arbitrary, which effects generalization performance of classifier. As the proper selection of input weights and hidden bias values contributes to the generalization capability of the classifier, the initialization of arbitrary weights may tend to decrease the generalization performance of the OS-ELM.

The last reason contributing to the lower inspection hit-rate of the OS-ELM is that, since OS-ELM only requires one parameter to fine tune, which is the number of hidden neurons in the hidden layer, in reality however, it is relatively difficult to obtain the best network structure using a trial and error procedure, as the optimum solution cannot be easily found. The ELM and OS-ELM, do suffer from a few drawbacks, which are indicated as follows.

For achieving comparable results, the number of neurons in the hidden layer must be chosen larger than in the standard BP algorithms. This is because the neuron weights and biases are not learned from the data.

As there is only one hidden layer in the SLFN, if trained properly, Multi-Layer Perceptron (MLP) networks with more than one hidden layer can possibly achieve similar and even better results as compared to ELM and OS-ELM.

The solution provided by ELM and OS-ELM is not always so smooth, and mostly shows some ripple.

The method of using SVM for fraud detection is very promising, as the SVM achieves the highest inspection hit-rate for fraud customer detection, as indicated in Table II. Firstly, it is noted that, SVM has non-linear dividing hypersurfaces that give it high discrimination. Secondly, SVM provides good generalization ability for

### Table I

**Experimental Results of the Proposed SVM, BPNN and OS-ELM Classification Techniques**

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Accuracy (Memorization)</th>
<th>Hit-rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPNN</td>
<td>Kota Bharu 85.23%</td>
<td>28.51%</td>
</tr>
<tr>
<td></td>
<td>Kuala Krai 81.08%</td>
<td>23.93%</td>
</tr>
<tr>
<td></td>
<td>Guu Musang 83.12%</td>
<td>26.38%</td>
</tr>
<tr>
<td>OS-ELM</td>
<td>Kota Bharu 82.96%</td>
<td>33.06%</td>
</tr>
<tr>
<td></td>
<td>Kuala Krai 75.43%</td>
<td>29.97%</td>
</tr>
<tr>
<td></td>
<td>Guu Musang 81.71%</td>
<td>30.45%</td>
</tr>
<tr>
<td>SVM</td>
<td>Kota Bharu 81.66%</td>
<td>42.56%</td>
</tr>
<tr>
<td></td>
<td>Kuala Krai 73.56%</td>
<td>38.30%</td>
</tr>
<tr>
<td></td>
<td>Guu Musang 79.27%</td>
<td>41.39%</td>
</tr>
</tbody>
</table>

### Table II

**Comparison of the Average Training Accuracy and Inspection Hit-rate of the Proposed SVM, BPNN and OS-ELM**

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Accuracy (Memorization)</th>
<th>Hit-rate (Onsite Inspection)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPNN</td>
<td>83.14%</td>
<td>26.27%</td>
</tr>
<tr>
<td>OS-ELM</td>
<td>80.03%</td>
<td>31.16%</td>
</tr>
<tr>
<td>SVM</td>
<td>78.16%</td>
<td>40.75%</td>
</tr>
</tbody>
</table>
unseen data classification. Lastly, SVM determines the optimal network structure itself, without requiring to fine tune any external parameters, as in the case of the BPNN and the OS-ELM. In contrast to the advantages of SVM over neural networks, there are however some drawbacks of SVM. These drawbacks are restricted due to practical aspects concerning memory limitation and real time training. Some of the major drawbacks of SVM are as follows.

The optimization problem arising in SVM is not easy to solve. Since the number of Lagrange multipliers is equal to the number of training samples, the training process is relatively slow. Even with the use of the SMO algorithm, real time training is not possible for a large set of data.

Another major drawback of SVM is the requirement of storage capacity. The support vectors (SVs) represent the important training samples describing the distinguishing features of the given classes. When the optimization problem has a low separability in the space used, the number of SVs increases. These SVs have to be stored in a model file for future classification. This puts limitations on the use of SVM for pattern recognition or classification in devices with limited storage capacity.

In comparing SVM to the OS-ELM, the only advantage of OS-ELM over SVM is its faster training process, with the increase in the chunk size. It is well known that with the RBF as the kernel function, SVM suffers from tedious parameter tuning.

The OS-ELM even with a single parameter to be tuned, its arbitrary assignment of initial weights requires it to search for the optimal size of neurons and execute many times in order to get the average value. Hence, in this case, the OS-ELM loses its edge over SVM. Given all these aspects, the author feels that SVM is a superior technique when the requirement is to solve a classification problem.

V. Conclusion

In this research works, three techniques were used to overcome the problem of detecting non technical losses in power utilities.

As discussed in previous section, the comparative results between the neural networks (BPNN and OS-ELM) and the proposed SVM model indicate that the proposed model has a higher performance, i.e. inspection hit-rate, compared to the BPNN and OS-ELM, as indicated by Table II. This indicates that the generalization performance of the SVM model is significantly better as compared to the BPNN and the OS-ELM.

The performance of a SVM can be however, problem dependent, as it is based on: the collected data set, the selected modeling features and the method in which the data is split between the training and testing/validating sets. In some cases, it’s worth noticing that the SVM has indicated a lower classification performance compared to neural network algorithms, as in the paper presented by Osareh et al. [51].

Further research can be focused on a knowledge extraction, i.e., covert the non-technical losses data to linguistic rules in “If-Then” format as proposed in [52].

References


Authors’ information

1College of Graduate Studyes, and Department of Electronics and Communication Engineering, Universiti Tenaga Nasional, Malaysia. Tel: +603-89273009, Fax: +603-89212065. Address: Jalan IKRAM-UNITEN, 43009 Kajang, Selangor, Malaysia.

2Department of Electronics and Communication Engineering, Universiti Tenaga Nasional, Malaysia.

3Dalle Molle Institute for Artificial Intelligence (IDSIA), CH-6928 Manno-Lugano, Ticino, Switzerland.

4Department of Electronics and Communication Engineering, Universiti Tenaga Nasional, Malaysia.

5Department of Mechanical Engineering, Universiti Tenaga Nasional, Malaysia.
K. S Yap received his B. Eng (Electrical) and M.Sc (Engineering) degree both from Universiti Teknologi Malaysia (UTM) in 1998 and 2000 respectively. In year 2010, he received PhD in Electronics Engineering from Universiti Sains Malaysia (USM). His research interests include theory and applications of artificial intelligence. Currently, he is the Programme Head (Electronics and Communication Engineering) of the College of Graduate Studies, Universiti Tenaga Nasional, Malaysia.

S. K. Tiong received the Bachelor’s and Master’s degrees in electrical, electronic, and system engineering from the Universiti Kebangsaan Malaysia (UKM) in 1997 and 2000, respectively, and the Ph.D. degree in mobile communication from the UKM in 2006. He is currently the Head of Electronic and Information Technology Unit of Power Engineering Centre (PEC) in the Universiti Tenaga Nasional (UNITEN), Selangor, Malaysia. He is also an Associate Professor in the Electronic and Communication Engineering Department of UNITEN. His research interests include digital electronics, microprocessor systems, artificial intelligence, and mobile cellular systems.

J. Nagi was born in Karachi, Pakistan, on March 23, 1985. He received the Bachelor’s and Master’s degrees (Hons.) in electrical and electronics, and electrical engineering from Universiti Tenaga Nasional (UNITEN), Malaysia, in 2007 and 2009, respectively. Currently, he is a with Dalle Molle Institute for Artificial Intelligence (IDSIA), CH-6928 Manno-Lugano, Ticino, Switzerland. His research interests include pattern recognition, image processing, load forecasting, fuzzy logic, neural networks.

J. Koh graduated with B. Eng (1st Class Hons.) in Electronics and Electrical, and M. Sc from University Putra Malaysia in Control and Automation. He received his PhD in Artificial Intelligence and Machine Learning. His research interests include Machine Intelligence and Automation, Lasers, Advanced Mechatronics and Control System. He holds a number of patents and has won a number of local and international engineering invention awards (Inpex, ITEX, Emic, MTE). He involves in a number of research projects, consultancy work and workshops, involving artificial intelligence, microcontroller and automation.

F. Nagi is working as associate professor in Universiti Tenaga Nasional, department of mechanical engineering. Graduated from NED engineering university, Karachi (1982). Worked in Pakistan International Airline for seven years as a project engineer on the development of guidance and control systems and was responsible for integration and testing of flight by wire system. Completed his MS from university of Miami USA (1989) and PhD from university of Nottingham UK (1993). From 1994-2000 worked in Pakistan Navy engineering college –NUST and had undertaken research projepts pertaining to submarine acoustic detection. Since 2001, is working in Universiti Tenaga Nasional and has completed various industrial and research projects. Field of research includes: Mechatronics system, fuzzy control, nonlinear control systems. http://metalab.uniten.edu.my/~farrukh/mywebpage