An Online Learning Network for Biometric Scores Fusion

Youngsung Kim\textsuperscript{a}, Kar-Ann Toh\textsuperscript{a,}\textsuperscript{*}, Andrew Beng Jin Teoh\textsuperscript{a}, How-Lung Eng\textsuperscript{b}, Wei-Yun Yau\textsuperscript{b}

\textsuperscript{a}School of Electrical & Electronic Engineering, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul 120-749, Republic of Korea.
\textsuperscript{b}Institute for Infocomm Research, Singapore 119613.

Abstract

In design of a multibiometric system, a major concern is the learning cost in terms of computation complexity and memory usage due to large size data. In this paper, we propose an online learning network to circumvent the computational problem. Although conventional online learning algorithms can be adopted, their optimization of the fitting residuals does not meet the actual classification error requirement. A direct optimization to the classification performance is thus desired. Since the proposed classification-based formulation involves a class-specific weight which varies according to the total number of genuine-users and imposters, an online learning formulation becomes non-trivial. Extensive empirical evaluations on publicly available data sets show promising potential of the proposed method in terms of fusion verification accuracy and computational cost.

Keywords: Multibiometrics, Score level fusion, Error rate, Online learning, Pattern classification, Extreme learning machine (ELM).

1. Introduction

1.1. Background

Biometrics is a technology which uses either inherited behavioral characteristics or biological traits from the user himself/herself for authentication. Recently, biometrics has been widely adopted to substitute conventional means such as token-based (keys, cards, or passwords) techniques in authentication system \cite{1}, \cite{2}, \cite{3}. However, a biometric system using a single trait may
be unreliable due to its limitation in terms of imperfect accuracy, non-universality, and vulnerability to attacks in practice.

To alleviate these problems, a multibiometric system [4] which combines multiple number of biometrics from the same identity has been attempted. According to [4], the biometrics data can be acquired from multi-sensors, multi-algorithms, multi-instances, or multi-modals. Fusion of multibiometrics can be performed at sensor level, feature level, abstract level, rank level, or match score level. Among these levels, fusion at match score level is often preferred since the match score contains the richest information related to the input pattern and it is relatively easy to access [5].

Multibiometric fusion at score level can be treated as a classification problem since the decision inference is to decide whether a query identity belongs to one of the two class labels—genuine-user or imposter. In this paper, we shall formulate an online learning algorithm for multibiometric fusion at the score level.

1.2. Motivation

In design of a multibiometric system, a major concern is the computational requirement. Firstly, due to a combination of multiple biometric data, the system has to deal with a larger set of data comparing with a unimodal biometric system. Very often, such data size demands for a large storage requirement as well as high computing power in order to achieve a near real-time performance. Secondly, since each new enrollee arrives sequentially or intermittently, the corresponding biometric template is generated adds on to the database sequentially. In other words, biometric data arrives either one by one or chunk by chunk for processing. Under such circumstance, the conventional batch-based learning methods become inefficient since a re-training process needs to be activated for each newly arrived data. Thirdly, the training score data grows drastically with respect to the number of enrollees and the number of samples per enrollee. This is because the match scores are obtained by comparing two templates from either the same or different users which are called intra-matching and inter-matching respectively. Particularly, when there is a large pool of users, the number of inter-matching grows exponentially while the number of intra-matching grows moderately. This creates an additional problem of imbalance data among the two
classes. To circumvent those computational problems, an online or incremental learning can be formulated. Since an online method uses only the newly arrived data to update the system, a low computing cost is anticipated during each updating cycle.

Many online learning algorithms such as Least Mean Square (LMS), Perceptron, Recursive Least Squares (RLS), and their variants have been developed during the past few decades. Generally, these learning algorithms aimed to optimize the residuals (distance errors) between the actual output and the predicted output. For example, the commonly adopted RLS aimed to minimize the sum of least squares of residuals (distance errors). The RLS has been widely used since it has a good convergence property [6]. However, this residual minimization leads to certain discrepancies between the actual classification error and the least residual fit.

1.3. Contribution and organization

In this paper, we propose an online learning algorithm to fuse multiple biometrics. The main contributions of this paper are enumerated as follows:

1) proposal of a novel recursive verification error rate minimization network which processes incoming samples one at time. The proposed online TER network algorithm can be operated on large scale multibiometrics scores fusion scenarios when the batch-based TER network algorithm fails.

2) Due to the approximation nature of the recursive error rate formulation, we analyze the convergence of the proposed online network solution with respect to the original batch-based classification-error-rate optimization network. The proposed online algorithm is shown to converge to the batch-based classification-error-rate solution but not to the batch-based least-squares-error minimization based solution (ELM).

3) provision of extensive empirical evaluation of several online algorithms on publicly available data sets. The proposed method shows either comparable or better performance in terms of verification accuracy and learning cost with respect to these state-of-the-art methods.

The paper is organized as follows: some preliminaries of a linear estimation-based network model are provided for immediate reference in section 2. In section 3, we present our proposed
online learning algorithm based on an approximated solution to minimize the verification error rate. This is followed by experiments based on two sets of publicly available databases in section 4. Finally, we provide some concluding remarks in section 5.

2. Preliminaries

2.1. Multibiometric decision and error rates

The basic task of biometric decision is to predict whether a given feature data belongs to a genuine-user or an imposter. If the output labels are known for a given set of input features, then the input and output pairs can be used for model inference. This type of learning is called supervised learning.

Given \( n \) number of learning observation pairs \( \{x_i, y_i\}_{i=1}^n \) where \( x_i = [x_{1i} \ x_{2i} \ \cdots \ x_{di}]^T \) is the feature vector which consists of \( d \) number of individual biometrics scores and \( y_i \in \{-1, +1\} \) is the learning target label where ‘−1’ denotes an imposter and ‘1’ denotes a genuine-user. Under the supervised learning framework, a biometric classifier can be considered to map a certain given input sample \( x \) to a certain desired output target label \( y \). Mathematically, a classifier can be represented as a continuous mapping function, \( f : \mathbb{R}^d \rightarrow \mathbb{R} \), with a labeling procedure (1), \( cls(f(\bullet)) : \mathbb{R} \rightarrow \mathbb{Z} \). Here, the classifier output label is determined with inference to \( f \) based on a decision threshold \( \tau \) given in (1). For convenience, we shall call the two class labels negative class (imposter) and positive class (genuine-user) in binary classification.

\[
\hat{y} = cls(f(x)) = \begin{cases} 
-1 \ (= \text{imposter}) & \text{if } f(x) < \tau, \\
+1 \ (= \text{genuine-user}) & \text{if } f(x) \geq \tau. 
\end{cases} 
\]  

(1)

By comparing a predicted label \( \hat{y}_i \) with the corresponding true label \( y_i \) for each operational setting of \( \tau \), correctly classified samples such as True Negatives (TN) and True Positives (TP), and incorrectly classified samples such as False Positives (FP) and False Negatives (FN) are obtained. Suppose there are \( n^- \) number of negative-class samples and \( n^+ \) number of positive-class samples \( (n = n^- + n^+) \) in the training set. Then these training TN, TP, FP, and FN can be normalized using their corresponding number of samples as \( \text{TNR} = \frac{\text{TN}}{n^-}, \text{TPR} = \frac{\text{TP}}{n^+}, \text{FPR} = \frac{\text{FP}}{n^-}, \)
and FNR = FN/n⁺ where the R in suffix indicates rate [7]. This normalization process can also be applied to the test set using its positive and negative sample sizes.

2.2. A SLFN

Due to its universal approximation capability, single hidden layer feedforward neural network (SLFN) has been widely adopted for supervised learning in decision making applications [8], [9].

Consider a set of training data \( \{(x_i, y_i) \mid x_i: \text{multibiometric scores as input; } y_i: \text{target label}, \ i = 1, 2, \ldots, n\} \). A standard SLFN involves \( m \) hidden nodes which are connected to the output nodes via a set of output weights denoted by \( \theta \). Each hidden node is first represented with a weighted summation of \( d \)-dimensional input \( [x_{1i} \ x_{2i} \ \cdots \ x_{di}]^T = x_i \) and then it passes through a nonlinear activation function at the hidden layer. Each hidden node is denoted by \( h_j = \phi(\alpha_j \cdot x_i + b_j) \in \mathbb{R} \) where \( j = 1, 2, \ldots, m \). \( \phi(\cdot) \) is an activation function, and \( \{\alpha_j, b_j\} \) are hidden layer weights for the \( j \)-th hidden node where \( \alpha_j = [\alpha_{1j} \ \alpha_{2j} \ \cdots \ \alpha_{dj}]^T \in \mathbb{R}^d \) and \( b_j \in \mathbb{R} \). Here, \( \alpha_j \cdot x_i \) denotes the inner product of \( \alpha_j \) and \( x_i \). The network output can be computed as: \( f(x_i) = h(x_i)^T \theta \) where \( h = [h_1 \ h_2 \ \cdots \ h_m]^T \in \mathbb{R}^m \) and \( \theta \in \mathbb{R}^m \).

In a compact vector-matrix notation, the output of the SLFN can be written as: \( f = H\theta \) where

\[
H = \begin{bmatrix}
    h(x_1) & \cdots & h(x_n)
\end{bmatrix}^T = \\
\begin{bmatrix}
    \phi(\alpha_1 \cdot x_1 + b_1) & \cdots & \phi(\alpha_m \cdot x_1 + b_m) \\
    \vdots & \ddots & \vdots \\
    \phi(\alpha_1 \cdot x_n + b_1) & \cdots & \phi(\alpha_m \cdot x_n + b_m)
\end{bmatrix}_{n \times m}
\]

and \( f = [f_1, f_2, \ldots, f_n]^T \in \mathbb{R}^n \). Here, \( H \) is called the hidden layer output matrix of the neural network. The \( j \)-th column of \( H \) is the \( j \)-th hidden node output with respect to inputs \( \{x_i\}_{i=1}^n \).

Unlike the conventional iteration based network learning, a single step solution has been proposed in [10] for SLFN. Essentially, the hidden node weight parameters \( \{(\alpha_j, b_j)\mid j = 1, 2, \ldots, m\} \) in \( H \) are randomly generated based on continuous probability distribution [10]. Then a least squares error fit for the output weight parameters \( \theta \) is obtained as [10]:

\[
\text{ELM: } \hat{\theta} = H^\dagger y = (H^T H)^{-1} H^T y,
\]

where \( H^\dagger \) is the Moore-Penrose generalized inverse of matrix \( H \). This single-step least-squares-error-based learning method was called ELM in [10]. This LSE-based solution of SLFN was
shown to achieve good performance particularly for regression applications. An extensive survey for such applications can be found in [11].

In order to match the classification requirement, in [12], an error rate based formulation for direct classifier optimization was proposed. Consider the total error rate (TER = FPR + FNR). Through adopting a quadratic approximation to the counting step function in the TER formulation, a closed-form solution has been proposed in [12]. The solution which minimizes the approximated TER is obtained as [12]:

\[
\text{TER: } \hat{\theta} = (H^T WH + \lambda I)^{-1} H^T W y,
\]

where

\[
H = \begin{bmatrix} H^- \\ H^+ \end{bmatrix} \in \mathbb{R}^{(n^-+n^+ \times m},
\]

\[
W = \text{diag} \left( \frac{1}{n^-}, \ldots, \frac{1}{n^+}, \frac{1}{n^+}, \ldots, \frac{1}{n^+} \right) \in \mathbb{R}^{(n^-+n^+) \times (n^-+n^+)},
\]

\[
y = \left[ (\tau - \eta), \ldots, (\tau - \eta), (\tau + \eta), \ldots, (\tau + \eta) \right]^T \in \mathbb{R}^{(n^-+n^+)}, H^- \in \mathbb{R}^{n^- \times m}, H^+ \in \mathbb{R}^{n^+ \times m}, \text{ and } \lambda > 0 \text{ controls a weighting of regularization. The variables } (H, n) \text{ are indicated by superscripts } - \text{ and } + \text{ for respective imposters and genuine-users.}
\]

In order to classify an unseen test data \(x_t\), the solutions (3) and (4) can be used to calculate the predicted output \(f(x_t) = h(x_t)^T \hat{\theta}\) where \(h(x_t) = \left[ \phi(\alpha_1 \cdot x_t + b_1) \cdots \phi(\alpha_m \cdot x_t + b_m) \right]^T\) is obtained from the test data. Then a threshold process (1) is applied to this predicted output to determine the class label: \(\hat{y} = \text{cls}(f(x_t))\).

2.3. Recursive least squares (RLS) and online sequential SLFN (OSELM)

The recursive least squares has been widely used in filter design or online learning formulations. This algorithm is operated under the assumption that training data arrives sequentially where the parameter vector is updated based on a previously learned parameter and an inverse of a correlation matrix [13], [6]. Based on RLS framework, an online learning network formulation of LSE-like solution (3) (called OSELM) has been proposed in [14], [15]. An update rule for the output parameter vector can be written as:

\[
\hat{\theta}_n = \hat{\theta}_{n-1} + R_n^{-1} h_n \left( y_n - h_n^T \hat{\theta}_{n-1} \right),
\]

where

\[
R_n = \left( \begin{array}{cc} H_n & \frac{1}{n} h_n \phi(\alpha_1 \cdot x_t + b_1)^T \\ \frac{1}{n} h_n^T \phi(\alpha_1 \cdot x_t + b_1) & \frac{1}{n} \end{array} \right)
\]

\[
h_n = \left[ \phi(\alpha_1 \cdot x_t + b_1) \cdots \phi(\alpha_m \cdot x_t + b_m) \right]^T
\]

\[
\hat{\theta}_0 = 0
\]

\[
y_n = \left[ (\tau - \eta), \ldots, (\tau - \eta), (\tau + \eta), \ldots, (\tau + \eta) \right]^T
\]

\[
R_0 = \frac{1}{n} \left( \begin{array}{cc} n & \frac{1}{n} \sum_{i=1}^{n} x_i \phi(\alpha_i \cdot x_i + b_i)^T \\ \frac{1}{n} \sum_{i=1}^{n} x_i^T \phi(\alpha_i \cdot x_i + b_i) & \frac{1}{n} \end{array} \right)
\]

\[
\hat{\theta}_n \rightarrow \text{optimal parameter vector}
\]

\[
y \rightarrow \text{output of system}
\]

\[
x \rightarrow \text{input of system}
\]

\[
\alpha \rightarrow \text{coefficients of system}
\]

\[
b \rightarrow \text{bias of system}
\]

\[
\tau \rightarrow \text{threshold of system}
\]

\[
\eta \rightarrow \text{noise of system}
\]

\[
\phi \rightarrow \text{basis functions of system}
\]
where the inverse of correlation matrix is updated as

$$R_n^{-1} = R_{n-1}^{-1} - R_{n-1}^{-1}h_n(I + h_n^TR_{n-1}^{-1}h_n)^{-1}h_n^TR_{n-1}^{-1}. \quad (7)$$

3. Proposed method

3.1. An online network for approximated TER learning

Although existing recursive/online methods including the OSELM algorithm mentioned in the preliminary section can be used to solve the computational cost problem occurred from a large size of training data, there remained a mismatch between the residuals learning objective (for example, the least square residuals in OSELM) and the performance criterion (FPR and FNR) with respect to biometric verification. In this section, we propose an online version to learn the TER network instead of the residuals objective.

The TER includes a varying weight which is determined according to the number of samples of each class (class-specific). Since each incoming data can be either genuine-user or imposter, the entry of weighting matrix varies according to the class label of new data. This is different from the ordinary RLS and OSELM which are based on batch mode least squares fit and has no weighting for each incoming sequence. The recursive version of generalized (or weighted) least squares contains either a fixed or single exponential function weight. Hence, the task towards an online TER network learning is non-trivial.

3.1.1. Learning based on sequential data

In this part, we re-define related notations of the above batch mode TER network solution (4) for sequential data solution. Here, we use subscripts to express the sequential situation: when the training match score vectors arrive one by one, they are expressed as $x_1, x_2, \ldots, x_n$ from the $1^{st}$ to the $n^{th}$ sequence. Based on the randomly generated hidden node parameters $\{(\alpha_j, b_j)|j = 1, 2, \ldots, m\}$, the corresponding hidden layer output vectors are determined as $h_1, h_2, \ldots, h_n (h_i \in \mathbb{R}^m)$. Consider $H_n \in \mathbb{R}^{n \times m}$ as an accumulated hidden layer output matrix based on data from $1^{st}$ to $n^{th}$ sequence, $W_n \in \mathbb{R}^{n \times n}$ is an accumulated weight matrix, and $y_n \in \mathbb{R}^n$ is an accumulated label vector. Based on these notations, the original batch mode TER network solution (4) can be
rewritten as:

\[ \hat{\theta}_n = (H_n^T W_n H_n)^{-1} H_n^T W_n y_n = R_n^{-1} Q_n, \]  

(8)

where \( R_n = H_n^T W_n H_n \), \( Q_n = H_n^T W_n y_n \) are weighted correlation matrices for current parameter estimation, and \( W_n \in \mathbb{R}^{n \times n} \) is a weight matrix which consists of weight elements \( w_n \). Two alternative weight elements can exist for \( w_n \) according the label \( y_n \) for each \( n \)th sequence, i.e.

\[
    w_n = \begin{cases} 
        w_n^- = \frac{1}{n} & \text{if } y_n = -1, \\
        w_n^+ = \frac{1}{n+1} & \text{if } y_n = +1. 
    \end{cases}
\]

Notice that \( w_n \) indicates the weight after \( n \) number of sequences has arrived. Superscripts \( - \) and \( + \) indicates the negative and positive classes respectively.

Now we shall separate the current data for \( n \)th and those previous sequences from \( 1 \)st to \((n - 1)\)th. The hidden output vector \( h_n \) of the newly arrived data \( x_n \) and the previously accumulated hidden output matrix \( H_{n-1} \) can be separated too. So does the label of the newly arrived data \( y_n \) and the previously accumulated label vector \( y_{n-1} \). Here, the packed data can be expressed as:

\[
    H_n = \begin{bmatrix} H_{n-1} \\ h_n^T \end{bmatrix}, \quad Y_n = \begin{bmatrix} y_{n-1} \\ y_n \end{bmatrix}. 
\]

(9)

The packed weight matrix can be expressed as:

\[
    W_n = \text{diag} \left( \left[ w_n \, w_n \, \ldots \, w_n \right] \right) = \begin{bmatrix} W_{n-1}^{\{w_n\}} & 0 \\ 0 & w_n \end{bmatrix},
\]

(10)

where \( W_{n-1}^{\{w_n\}} \in \mathbb{R}^{(n-1) \times (n-1)} \) contains \( w_n \) as elements instead of \( w_{n-1} \) as a partition of the \( W_n \) matrix. \( W_{n-1}^{\{w_n\}} \) is for weighting the previously accumulated data and \( w_n \) is for weighting the newly arrived data. The above weighted correlation matrix \( R_n \) and \( Q_n \) can be rewritten based on the elements of the above partitioned matrices:

\[
    R_n = H_{n-1}^T W_{n-1}^{\{w_n\}} H_{n-1} + h_n w_n h_n^T, 
\]

(11)

\[
    Q_n = H_{n-1}^T W_{n-1}^{\{w_n\}} y_{n-1} + h_n w_n y_n.
\]

(12)

Since all data should be multiplied with the new weight generated at the current sequence, the previously accumulated data matrix should be rescaled according to the new total number of
positive-class data and negative-class data. The weighted correlation matrices at previous iteration are defined as \( R_{n-1} = H_{n-1}^T W_{n-1} H_{n-1} \) and \( Q_{n-1} = H_{n-1}^T W_{n-1} y_{n-1} \) where \( W_{n-1} = \text{diag} \left( [w_{n-1} \cdots w_{n-1}] \right) \in \mathbb{R}^{(n-1) \times (n-1)} \) with \( w_{n-1} \in \{ w_{n-1}^-, w_{n-1}^+ \} \). Here we note that \( H_{n-1}^T W_{n-1}^{\{w_n\}} H_{n-1} \) and \( H_{n-1}^T W_{n-1}^{\{w_n\}} y_{n-1} \) are not equivalent to \( R_{n-1} \) and \( Q_{n-1} \), respectively since \( W_{n-1} \) and \( W_{n-1}^{\{w_n\}} \) are different. The weight matrix \( W_{n-1} \) consists of \( w_{n-1} \) as elements and the weight matrix \( W_{n-1}^{\{w_n\}} \) consists of \( w_n \) as elements.

Our next task is to express the weighted correlation matrices at the last sequence \( R_n \) in (11) and \( Q_n \) in (12) in terms of those at previous sequence \( R_{n-1} \) and \( Q_{n-1} \). By introducing a scaling factor of \( w_n^c/w_{n-1}^c \) for each class \((c \text{ denotes either } - \text{ or } + \text{ class})\), (11) and (12) can be expressed in terms of \( R_{n-1} \) and \( Q_{n-1} \) respectively:

\[
R_n = \frac{w_n^-}{w_{n-1}^-} R_{n-1}^- + \frac{w_n^+}{w_{n-1}^+} R_{n-1}^+ + h_n w_n h_n^T \tag{13}
\]

\[
Q_n = \frac{w_n^-}{w_{n-1}^-} Q_{n-1}^- + \frac{w_n^+}{w_{n-1}^+} Q_{n-1}^+ + h_n w_n y_n \tag{14}
\]

Different from the generalized (weighted) least squares where the weight elements are tied to each sample, the weights for TER solution are tied to each class. Since the weight of the generalized least squares can be pre-defined, an update of the weight matrix is not required. However, the weight of TER varies according to the number of received samples. For this reason, the above scaling \( \frac{w_n^c}{w_{n-1}^c} \) is employed for weight matrix updating.

3.1.2. An approximation to the weighted correlation matrix

In this part, we reformulate the above weighted correlation matrix in (13) which is expressed by summation of three components \( R_n = \frac{w_n^-}{w_{n-1}^-} R_{n-1}^- + \frac{w_n^+}{w_{n-1}^+} R_{n-1}^+ + h_n w_n h_n^T \) to summation of two components \((A + BCD)\). This is to utilize matrix inversion lemma [13], [16] so that the inverse calculation of the weighted correlation matrix can be removed from the batch mode solution. The
The matrix inversion lemma \cite{13}, \cite{16} is expressed as:

\[
(A + BCD)^{-1} = A^{-1} - A^{-1}B \left(C^{-1} + DA^{-1}B\right)^{-1} DA^{-1}.
\]  

(15)

Since (13) is in the form of \(A_1 + A_2 + BCD\), it cannot be applied to the matrix inversion lemma directly. Here, we attempt to reduce the three terms to two terms \((A, BCD)\) by utilizing a matrix approximation as follows: \(A_1 + A_2 \cong A\). We shall remove such summation expression in (13), (14) using partitioned matrices:

\[
\sum_{c \in \{-, +\}} \frac{w^n_c}{w^{n-1}_c} R_{n-1}^c = \begin{bmatrix} R_{n-1}^- & R_{n-1}^+ \end{bmatrix} \begin{bmatrix} w^{n}_{n-1} \\ w^{n}_{n-1}^{-1} \end{bmatrix},
\]

(16)

\[
\sum_{c \in \{-, +\}} \frac{w^n_c}{w^{n-1}_c} Q_{n-1}^c = \begin{bmatrix} Q_{n-1}^- & Q_{n-1}^+ \end{bmatrix} \begin{bmatrix} w^{n}_{n-1} \\ w^{n}_{n-1}^{-1} \end{bmatrix}.
\]

(17)

In order to express \(\begin{bmatrix} R_{n-1}^- & R_{n-1}^+ \end{bmatrix}\) as a single matrix, we utilize \(\begin{bmatrix} 1 \\ 1 \end{bmatrix}\) and its generalized inverse \(\begin{bmatrix} 1 \\ 1 \end{bmatrix}^\dagger\) for an approximated formulation:

\[
R_n = \begin{bmatrix} R_{n-1}^- & R_{n-1}^+ \end{bmatrix} \cdot I \cdot \begin{bmatrix} w^{n}_{n} \\ w^{n}_{n-1} \\ w^{n}_{n-1}^{-1} \\ w^{n+}_{n} \\ w^{n+}_{n-1} \\ w^{n+}_{n-1}^{-1} \end{bmatrix} + h_n w_n h_n^T
\]

\[
= u_n R_{n-1} + h_n w_n h_n^T
\]

(18)

Similarly,

\[
Q_n = \begin{bmatrix} Q_{n-1}^- & Q_{n-1}^+ \end{bmatrix} \cdot I \cdot \begin{bmatrix} w^{n}_{n} \\ w^{n}_{n-1} \\ w^{n}_{n-1}^{-1} \\ w^{n+}_{n} \\ w^{n+}_{n-1} \end{bmatrix} + h_n w_n y_n
\]

\[
= u_n Q_{n-1} + h_n w_n y_n
\]

(19)
Here we note that the inverse of \( \begin{bmatrix} 1 & 1 \end{bmatrix}^T \) does not exist due to its singularity. However, a pseudo inverse is possible. According to the Moore-Penrose generalized inverse theorem [17], a given matrix \( M \in \mathbb{R}^{m \times n} \) has a unique inverse \( M^+ \in \mathbb{R}^{n \times m} \) if the matrix satisfies the following four conditions (considering only the case where \( M \) consists of real numbers): (i) \( MM^+M = M \), (ii) \( M^+MM^+ = M^+ \), (iii) \( (MM^+)^T = MM^+ \), and (iv) \( (M^+M)^T = M^+M \). Vector \( \begin{bmatrix} 1 & 1 \end{bmatrix}^T \) has a unique inverse of \( \frac{1}{2} \begin{bmatrix} 1 & 1 \end{bmatrix} \) which satisfies all four conditions. The effect of using this generalized inverse is that the updating weight \( M \) becomes an approximated weight given by \( \frac{1}{2} \left( \frac{w_n}{w_{n-1}} + \frac{w_{n+1}}{w_{n-1}} \right) \) which is derived from \( \begin{bmatrix} 1 \\ 1 \end{bmatrix} \). The resulting net effect is that a single weight updating for the accumulated data (but, the incoming data is updated with a new weight according to its class label). We shall observe the impact of such approximation in the experiments.

3.1.3. An online learning network solution

Using the above approximations in (18) and (19), an online TER network learning solution can be written. The estimated parameter update rule can be written as:

\[
\hat{\theta}_n = R_n^{-1}Q_n = R_n^{-1}\left( u_nQ_{n-1} + h_nw_ny_n \right) \\
= R_n^{-1}\left( u_nR_{n-1}\hat{\theta}_{n-1} + h_nw_ny_n \right) \\
= R_n^{-1}\left( R_n - h_nw_nh_n^T \right) \hat{\theta}_{n-1} + h_nw_ny_n \\
= R_n^{-1}\left( R_n\hat{\theta}_{n-1} + h_nw_n\left( y_n - h_n^T\hat{\theta}_{n-1} \right) \right) \\
\hat{\theta}_n = \theta_{n-1} + w_nR_n^{-1}h_n\left( y_n - h_n^T\theta_{n-1} \right),
\]

where the weighted correlation matrix is updated as

\[
R_n^{-1} = (u_n)^{-1}R_{n-1}^{-1} - \frac{(u_n)^{-1}h_nh_n^T(u_n)^{-1}R_{n-1}^{-1}h_n}{(u_n)^{-1}h_n^T(u_n)^{-1}R_{n-1}^{-1}h_n}.
\]

Here, (21) which is inverse of \( u_nR_{n-1} + h_nw_nh_n^T \) was obtained based on the matrix inversion lemma [13], [16] in (15), by letting \( A = u_nR_{n-1} \), \( B = h_n \), \( C = w_n \), \( D = h_n^T \).

In summary, when a newly incoming sample arrives, its hidden node outputs are computed based on randomly pre-generated hidden node weights accordingly. Then the learning solution
is updated using these newly generated hidden node outputs and the previous updated solution. Since the new solution is based on the previous solution and new data, the online computational cost can be reduced as compared to that of the batch mode TER learning. The online TER network algorithm is summarized in Algorithm 1.

3.2. Analysis: comparison between the solutions of OTERN and OSELM

Due to the class-specific weighting, the solution of OTERN should be different from that of OSELM. We shall compare the difference between these two solutions in this section. Our goal here is to show that the solution of (approximate) OTERN converges to the solution of TERN which is based on classification error counting, and this is different from the solution of OSELM which is based on least squares error distance. To do this, we shall conduct the following experiments and evaluate it using a set of synthesized data.

1) first we note that the solution of OSELM using sequential data converges to the solution of ELM using batch data.
2) then we show that the solution of the proposed approximate OTERN using sequential data converges asymptotically to the solution of TERN using batch data. The convergence is shown through the weight of OTERN solution which converges asymptotically to the weight of TERN solution. Since the solutions of ELM and TERN are different, we can then conclude that OTERN solves the classification problem similar to TERN instead of regression problem which is solved by ELM.

3.2.1. Convergence of OSELM solution based on sequential data

The online-based OSELM solution is known to converge to the batch-based ELM solution based on batch data [15]. We shall omit the proofs here as it is rather trivial.

3.2.2. Convergence of OTERN solution based on sequential data

Note again that in order to make the OTERN solution similar to the batch-based TERN solution after the \( n^{th} \) sequence has been learned, the past sequential data should be re-weighted with the weight at the \( n^{th} \) sequence. In online/recursive learning, the previous weight is continuously updated using updating terms instead of re-weighting using the new weight. The key observation
Algorithm 1 Online TER network (OTERN) learning algorithm (training)

Input: \(x_i \in \mathbb{R}^d, y_i \in \mathbb{R}^1, i = 1, 2, \ldots\)

Output: \(\hat{\theta}_i\).

Initialization:

0. Set number of hidden node \(m \geq d\).

1. Hidden node parameters: \(\alpha_j \leftarrow \text{random} \in \mathbb{R}^d, b_j \leftarrow \text{random} \in \mathbb{R}^1, j = 1, 2, \ldots, m\).

2. Initial solution (\(\hat{\theta}_0\)) and parameters: there are two ways to set,

   1) Random initial solution:
      i. Initial solution \(\hat{\theta}_0 \leftarrow \text{random} \in \mathbb{R}^m\).
      ii. Initial inverse of correlation matrix: \(R_0^{-1} \leftarrow \frac{1}{\lambda}I \in \mathbb{R}^{m \times m}, 0 < \lambda < 1\).
      iii. Initial weights \(w_0^- \leftarrow 1, w_0^+ \leftarrow 1, n_0^- \leftarrow 0, n_0^+ \leftarrow 0, n_0 \leftarrow 0\).

   2) Learned initial solution using initial chunk data:
      i. Initial chunk data: \(\{x_i\}_{i=1}^{n_0}\),
      ii. Initial hidden output matrix: \(H_0 \leftarrow \begin{bmatrix} \phi(\alpha_1 \cdot x_1 + b_1) & \cdots & \phi(\alpha_m \cdot x_1 + b_m) \\ \vdots & \ddots & \vdots \\ \phi(\alpha_1 \cdot x_{n_0} + b_1) & \cdots & \phi(\alpha_m \cdot x_{n_0} + b_m) \end{bmatrix} \in \mathbb{R}^{n_0 \times m}\).
      iii. Initial weights: \(w_0^- \leftarrow \frac{1}{n_0^-}, w_0^+ \leftarrow \frac{1}{n_0^+}\).
      iv. Initial inverse of correlation matrix: \(R_0^{-1} \leftarrow (H_0^T W_0 H_0 + \lambda I)^{-1} \) and \(Q_0 \leftarrow H_0^T W_0 y_0\) where \(W_0 \in \{w_0^-, w_0^+\}, 0 < \lambda < 1, I \in \mathbb{R}^{m \times m}\).
      v. Estimate initial solution \(\hat{\theta}_0 \leftarrow R_0^{-1} Q_0 \in \mathbb{R}^m\).

Sequential learning:

\[\text{for } i > n_0 \text{ do}\]

\[h(x_i) = \begin{bmatrix} \phi(\alpha_1 \cdot x_i + b_1) \\ \vdots \\ \phi(\alpha_m \cdot x_i + b_m) \end{bmatrix}^T\]

1. Update weights:

   if \(x_i \in \text{class } (-)\) then
   \[n_i^- \leftarrow (n_{i-1}^- + 1), n_i^+ \leftarrow (n_{i-1}^+).
   \]
   else \(\{x_i \in \text{class } (+)\}\)
   \[n_i^+ \leftarrow (n_{i-1}^+ + 1), n_i^- \leftarrow (n_{i-1}^-).
   \]
   end if

\[w_i^c \leftarrow \frac{1}{n_i^c}, c \in \{-, +\},\]
\[u_i \leftarrow \frac{1}{2} \left( \frac{w_i^-}{w_{i-1}^-} + \frac{w_i^+}{w_{i-1}^+} \right)\].

2. Update \(R_i^{-1}\) in (21) and solution \(\hat{\theta}_i\) in (20).

end for
for this is thus a convergence of a weight which is multiplied to each sequence when the number of sequences increases.

Fig. 1 illustrates that the weight for the previous sequence data is updated through learning iterations. The weight of OTERN at \(i^{th}\) sequence \((w_i)\) keeps being updated using a product of an updating weight \((u_k)\). At the current \(n^{th}\) sequence, the stacked weight becomes \(\tilde{w}_n\) which should converge to \(w_n\). Using the weight matrix consists of \(n\) number of weights for each sequence data,

\[
\theta_n^{OTERN} = \left( H_n^T W_n^{OTERN} H_n \right)^{-1} H_n W_n^{OTERN} y_n, \tag{22}
\]

where \(W_n^{OTERN} = \text{diag} \left( \tilde{w}_1, \tilde{w}_2, \tilde{w}_n \right) \in \mathbb{R}^{n \times n}\) is weight matrix at the \(n^{th}\) sequence.

Note again that \(\tilde{w}_n\) is the weight stacked after \(n\) number of sequences has been learned. The elements of the weight matrix \(W_n^{OTERN}\) is written in detail by

\[
\tilde{w}_n = \left( \prod_{k=i+1}^{n} u_k \right) \times w_i \quad \in \mathbb{R} : i = 1, \ldots, n-1, \text{ and } w_n, \tag{23}
\]
where a stacking weight update is

\[
 u_k = \begin{cases} 
 \frac{1}{2} \left( 1 + \frac{w_k}{w_{k-1}} \right) & \text{if } y_k = -1, \left( \because w_k = w_{k-1} \right), \\
 \frac{1}{2} \left( \frac{w_k}{w_{k-1}} + 1 \right) & \text{if } y_k = +1, \left( \because w_k = w_{k-1} \right), 
\end{cases}
\] (24)

When \( n \) is large, \( \tilde{w}_n \) converges to \( w_n \) since the stacking weight update approaches ‘1’ and \( w_{n-1} \) approaches \( w_n \) (A detailed proof is shown in Appendix A.2). As a result, we can conclude that:

\[
 \hat{\theta}^\text{OTERN}_n \approx \hat{\theta}^\text{TERN}_n. 
\] (25)

3.2.3. Comparison of converged solutions of OTERN, OSELM, and ELM

The reason why the OTERN solution differs from OSELM solution is mainly due to the OTERN’s weights. Here, we directly compare OTERN solution to OSELM solution to check whether these solutions are similar or not.

\[
 \hat{\theta}^\text{OSELM}_n - \hat{\theta}^\text{OTERN}_n \\
= (H_n^T H_n)^{-1} H_n y_n - (H_n^T W^\text{OTERN}_n H_n)^{-1} H_n W^\text{OTERN}_n y_n \\
= (H_n^T I_n H_n)^{-1} H_n I_n y_n - (H_n^T W^\text{OTERN}_n H_n)^{-1} H_n W^\text{OTERN}_n y_n \\
\neq 0, \left( \because I_n \neq W^\text{OTERN} \text{ where } W^\text{OTERN} \in \{ \tilde{w}_n \neq 1, w_n \neq 1 \} \right),
\] (26)

\[
 \therefore \hat{\theta}^\text{OSELM}_n \neq \hat{\theta}^\text{OTERN}_n.
\]

In a similar way, the OTERN solution is different from the ELM solution since \( \hat{\theta}^\text{ELM}_n - \hat{\theta}^\text{OTERN}_n \neq 0 \).

3.2.4. Simulation to compare between OTERN and OSELM solutions using artificial data

Here, we simulate the above derivations using a set of synthesized normally distributed two-dimensional data. \( \text{class}(-) \) has 2100 samples, and \( \text{class}(+) \) has 700 samples.

**Convergence trends of the weights.** Through observing the weight \( \tilde{w}_n, \prod_{k=i+1}^{n} u_k, u_k, \) and \( w_i \), we shall check whether the above results are reliable or not.
In Fig. 2 (a), the term $u_k$ shows convergence to ‘1’ as the number of sequences increases. Subsequently, the term $\prod_{k=i+1}^{n} u_k$ approaches ‘1’. These verify the first reason of $\tilde{w}_n$ convergence mentioned in the above.

In Fig. 2 (b), the value of $\tilde{w}_n$ is depicted using bar graph. Each bar indicates the weight which is multiplied to sequences from $1^{st}$ to $n^{th}$. There are two kinds of weight value corresponding to each class of sequence and they converge to the final weight of $w_n$ which consists of $w^-_n$ and $w^+_n$ values. Fig. 2 (c) shows zoomed-in view of Fig. 2 (b). Those gray-colored bars show that $\tilde{w}_n$ for negative-class converges $w^-_n$ and those black-colored bars show that $\tilde{w}_n$ for positives-class converges $w^+_n$.

Figure 2: (a) Trends of $w_i \in \{w^-_i, w^+_i\}$, $\prod_{k=i+1}^{n} u_k$ along sample index. (b) Lines: $w^-_i$ and $w^+_i$ (solid lines), $w^-_n$ and $w^+_n$ (dotted lines) along sample index, Bars: $\tilde{w}_n$. (c) Zoomed-in view of (b): Bar of $\tilde{w}_n$ where gray-colored bar is for class(−) and black-colored bar is for class(+).
**Sum of squared errors between the solutions of OTERN and OSELM.** Fig. 3 shows Sum of Squared Error (SSE) values between the solutions of OTERN and TERN, and between OTERN and OSELN. These solutions have been obtained based on random initialization. As shown in the figure, the SSE of OTERN and TERN solutions approaches very small value (or zero) but not for OTERN and OSELM.

![Figure 3: Sum of Squared Error (SSE) values between solution of OTERN and that TERN, and that OSELN.](image)

3.2.5. Conclusion of analysis

We conclude: (a) the solution of approximate OTERN converges to that of TERN when \( n \) tends to infinity, (b) the solution of approximate OTERN is different from that of OSELM.

4. Experiments

4.1. Data sets and computing environment

Several experiments have been performed on publicly available databases. The XM2VTS Score-level Fusion Benchmark Dataset [18] and the NIST Biometric Scores Set–Release 1 (BSSR1) [19] will be experimented under a scores fusion scenario.

4.1.1. XM2VTS score-level fusion benchmark data

The XM2VTS score-level fusion dataset consists of classifier’s output scores from face and speech biometrics. Total thirty two fusion cases were generated using face and speech modalities based on two fusion protocols (LP1, LP2), two different classifiers (MLP and GMM), and
several features (FH, DCTs, DCTb, LFCC, PAC, and SSC) [18]. The database contains both inter-modalities fusion (multimodal system) and intra-modalities fusion (multi-algorithms system). Table 1 shows a summary of the data sets used for our performance evaluation.

4.1.2. NIST–BSSR1 scores data

The NIST–BSSR1 database consists of raw outputs of similarity scores from comparing two templates. There are three biometrics score sets namely face–face, finger–finger, and finger–face in the database. (1) The face–face set was generated based on two different face recognition systems (C, G) which were applied to 3,000 users. The data set consists of 3,000 genuine-user scores and 8,997,000 imposter scores. This data set represents the case of a multi-algorithm system [4]. (2) The finger-finger set was generated based on one fingerprint system (V) which was applied to 6,000 users. The same matcher was used to generate scores from the right index finger and the left index finger of the same user. The data set consists of 6,000 genuine-user scores and 35,994,000 imposter scores. This data set represents the case of a multi-instance system [4]. (3) The finger-face set was generated based on the fingerprints and face templates of 517 users. The fingerprint scores were generated similar to that in (2) and face scores were generated similar to that in (1). The data set consists of 517 genuine-user scores and 266,772 imposter scores. This data set represents the case of a multimodal system [4]. Table 2 summarizes the data sets used in our biometric performance evaluation.
<table>
<thead>
<tr>
<th>(feature, classifier)</th>
<th>biometrics pair</th>
<th>protocol</th>
<th>#scores</th>
<th>(genuine + imposter)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>for training</td>
</tr>
<tr>
<td>(i)</td>
<td>{FH,MLP}, (LFCC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ii)</td>
<td>{FH,MLP}, (PAC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(iii)</td>
<td>{FH,MLP}, (SSC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(iv)</td>
<td>{DCTs,GMM}, (LFCC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(v)</td>
<td>{DCTs,GMM}, (PAC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(vi)</td>
<td>{DCTs,GMM}, (SSC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(vii)</td>
<td>{DCTb,GMM}, (LFCC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(viii)</td>
<td>{DCTb,GMM}, (PAC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ix)</td>
<td>{DCTb,GMM}, (SSC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x)</td>
<td>{DCTs,MLP}, (LFCC,GMM)</td>
<td></td>
<td></td>
<td>LP1</td>
</tr>
<tr>
<td>(xi)</td>
<td>{DCTs,MLP}, (PAC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xii)</td>
<td>{DCTs,MLP}, (SSC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xiii)</td>
<td>{DCTb,MLP}, (LFCC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xiv)</td>
<td>{DCTb,MLP}, (PAC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xv)</td>
<td>{DCTb,MLP}, (SSC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xvi)</td>
<td>{FH,MLP}, (DCTs,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xvii)</td>
<td>{FH,MLP}, (DCTb,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xviii)</td>
<td>{FH,MLP}, (DCTs,MLP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xix)</td>
<td>{FH,MLP}, (DCTb,MLP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xx)</td>
<td>{DCTs,GMM}, (DCTs,MLP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxi)</td>
<td>{DCTb,GMM}, (DCTb,MLP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxii)</td>
<td>{LFCC,GMM}, (SSC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxiii)</td>
<td>{PAC,GMM}, (SSC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxiv)</td>
<td>{FH,MLP}, (LFCC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxv)</td>
<td>{FH,MLP}, (PAC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxvi)</td>
<td>{FH,MLP}, (SSC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxvii)</td>
<td>{DCTb,GMM}, (LFCC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxviii)</td>
<td>{DCTb,GMM}, (PAC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxix)</td>
<td>{DCTb,GMM}, (SSC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxx)</td>
<td>{FH,MLP}, (DCTb,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxxi)</td>
<td>{LFCC,GMM}, (SSC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xxxi)</td>
<td>{PAC,GMM}, (SSC,GMM)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 2: Summary of NIST–BSSR1 data sets for scores fusion.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>biometric traits</th>
<th>#matchers (name): #users</th>
<th>#scores per matcher: (genuine+imposter)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Finger–Face</td>
<td>Fingerprint &amp; Face</td>
<td>4 (C &amp; G &amp; V ×2)</td>
<td>517 517+266,772</td>
</tr>
<tr>
<td>(b) Face–Face</td>
<td>Face (2 images/person)</td>
<td>2 (C &amp; G)</td>
<td>3,000 3,000+8,997,000</td>
</tr>
<tr>
<td>(c) Finger–Finger</td>
<td>Fingerprint (right&amp;left fingers)</td>
<td>2 (V×2)</td>
<td>6,000 6,000+35,994,000</td>
</tr>
</tbody>
</table>

Remark: C, G, and V are the name of matchers.

In the following, we run all experiments using Matlab [20] on an Intel Core i7 3.4 GHz CPU PC with 8 GB RAM.

4.2. Evaluation setup

Our main goal for experimentation is to establish the fusion classification accuracy and the training CPU cost with respect to the proposed method and those state-of-the-art methods.

4.2.1. Fusion verification performance

In this part, our main goal is to benchmark the proposed online-based method with online-based state-of-the-art algorithms in terms of fusion verification accuracy performance.

Compared algorithms. The proposed online TER network algorithm (OTERN) is compared with: (i) batch-based TERN and (ii) online-based competing algorithms such as Online Sequential ELM (OSELM) [14], Perceptron [21], Online Passive Aggressive algorithm (PA) [22], and Online Confident Weighted algorithm (CW) [23].

Protocol. For the XM2VTS data set, the given partitioned sets are used. For the NIST–BSSR1 data set, the entire data is partitioned into two equal sets, each consisting of randomly permuted

---

1For an enhanced and stable performance, a regularization factor (0.0001) is adopted and initial solution is learned from batch based ELM solution using small initial examples.

2Matlab codes for PA and CW are acquired from http://webee.technion.ac.il/people/koby/code-index.html
samples. For statistical evidence, we perform ten runs of such holdout tests and the results will be reported in averaged form for NIST-BSSR1. The input data of all experiments were normalized to within the interval $[-1, 1]$ and the output label is set as $\{-1, 1\}$. We fixed $\tau = 0$ and $\eta = 1$ for TERN methods throughout all experiments (giving $(\tau - \eta = -1)$ and $(\tau + \eta = +1)$ as the learning targets).

**Parameter setting.**

- **XM2VTS data sets:** Both the batch-based TERN method and the state-of-the-art online-based methods are compared with the proposed algorithm. Our goal here is to show that our proposed online TER network has a comparable or better fusion verification performance with those state-of-the-arts.

  The parameters for algorithm tuning include: number of hidden nodes ($m$) in TERN, OTERN, and OSELM. Six different values of the classifier parameters are examined based on the holdout protocol: the number of hidden nodes $m$ is chosen within $\{d, 2 \cdot d, 5 \cdot d, 10 \cdot d, 20 \cdot d, 40 \cdot d\}$ where $d$ is the input feature dimension (number of individual biometric scores). The regularization term $\lambda$ is set at 0.0001 which was selected based on a two-fold cross validation using only training data. Other parameters (i.e. margin constraint $C$ in PA and CW methods) used the default values respectively.

- **NIST–BSSR1 data sets:** Since this data involves large scale sample size, the batch-based TERN method is excluded. Our goal here is to show that our proposed online TER network method has a comparable or better fusion verification performance with those online-based state-of-the-arts. For NIST–BSSR1 data sets, we apply a similar settings to XM2VTS data sets.

**Measurement.** For performance evaluation of all data sets, the half total error rate [18], [24] and the accuracy are adopted. Since biometric scores (‘genuine-user’ and ‘impostor’) distribution is quite imbalance, HTER measure is adopted apart from Accuracy measure. These performance measures are defined in Table 3. Both HTER and Accuracy require to set a threshold for decision. This threshold is set as ‘0’ for the signum function adopted in all methods.
Table 3: Performance measurements for evaluation.

<table>
<thead>
<tr>
<th>name</th>
<th>formulation</th>
<th>threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTER</td>
<td>( \frac{1}{2} \times \left( \frac{FP}{N} + \frac{FN}{P} \right) )</td>
<td>required (( \tau ))</td>
</tr>
<tr>
<td></td>
<td>( = \frac{1}{2} \times \left( \frac{1}{n^-} \sum_{j=1}^{n^-} 1_{f(x_j^-) \geq \tau} + \frac{1}{n^+} \sum_{i=1}^{n^+} 1_{f(x_i^+) &lt; \tau} \right) )</td>
<td></td>
</tr>
</tbody>
</table>

Accuracy \( \frac{TN + TP}{N + P} = \frac{1}{n^- + n^+} \left( \sum_{j=1}^{n^-} 1_{f(x_j^-) < \tau} + \sum_{i=1}^{n^+} 1_{f(x_i^+) \geq \tau} \right) \), required (\( \tau \))

when \( N=P \), Accuracy = 1 − HTER.

4.2.2. Computational efficiency

In this part, we shall compare the learning cost in terms of execution CPU time during training. The comparisons are performed as follows:

**CPU time using sequential data.** In the evaluation, the data set (i) from XM2VTS is adopted to show the CPU time during sequential training. We shall perform (a) Comparison of batch-based TERN and online-based OTERN, and (b) Comparison of OTERN and competing online-based state-of-the-arts.

**CPU time of online-based methods using batch data.** NIST-BSSR1 data set is used to show the CPU time during training. These CPU measurements are recorded from the experiment of fusion verification performance for NIST-BSSR1 data set.

4.3. Evaluation results

4.3.1. Scores fusion verification performance

**XM2VTS data sets.** Fig. 4 shows the HTER and Accuracy performances of the compared methods for all data sets. The HTER and Accuracy results of the compared methods are shown at which the parameter has shown the best performances in individual data sets. For OTERN and OSELM
methods, forty training samples (approximately 0.001% of total training samples) are used to estimate the initial solution (the second type of solution initialization, see Algorithm 1).

In Fig. 4, we see that OTERN shows a similar performance to that of the batch-based TERN. Both the batch and online based TERN methods (TERN and OTERN) show better HTER than those state-of-the-art methods (OSELM, Perceptron, PA, and CW) in Fig. 4 (a) and (b). In terms of Accuracy, OSELM shows the best performance in many individual data sets in Fig. 4 (c) and (d).

**NIST–BSSR1 data sets.** Table 4 shows the averaged HTER and Accuracy performances of individual data sets among the compared online based methods. These results are shown at which the parameter has shown the best averaged HTER and averaged Accuracy performances in individual data sets. For OTERN and OSELM method, one thousand training samples (less than 0.001% of
total training samples) are used to learn the initial solution.

In Table 4, we see that the proposed OTERN shows either comparable or better HTER and Accuracy as compared with those online-based state-of-the-art methods (OSELM, Perceptron, PA, and CW).

Table 4: Comparison of average verification performances (HTER and Accuracy), standard deviation (±), and execution CPU time (seconds) for NIST–BSSR1 data sets.

<table>
<thead>
<tr>
<th>HTER</th>
<th>OTERN</th>
<th>OSELM</th>
<th>Perceptron</th>
<th>PA</th>
<th>CW</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIST finger-face</td>
<td>0.0030 (±0.0014)</td>
<td>0.0604 (±0.0102)</td>
<td>0.1098 (±0.0302)</td>
<td>0.5000 (±0.0000)</td>
<td>0.1099 (±0.0132)</td>
</tr>
<tr>
<td>NIST face-face</td>
<td>0.0379 (±0.0015)</td>
<td>0.3464 (±0.0049)</td>
<td>0.5000 (±0.0000)</td>
<td>0.5000 (±0.0000)</td>
<td>0.5000 (±0.0000)</td>
</tr>
<tr>
<td>NIST finger-finger</td>
<td>0.0346 (±0.0023)</td>
<td>0.1925 (±0.0085)</td>
<td>0.4996 (±0.0012)</td>
<td>0.5000 (±0.0000)</td>
<td>0.5000 (±0.0000)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>OTERN</th>
<th>OSELM</th>
<th>Perceptron</th>
<th>PA</th>
<th>CW</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIST finger-face</td>
<td>0.9999 (±0.0001)</td>
<td>0.9998 (±0.0000)</td>
<td>0.9993 (±0.0002)</td>
<td>0.9981 (±0.0000)</td>
<td>0.9994 (±0.0001)</td>
</tr>
<tr>
<td>NIST face-face</td>
<td>0.9732 (±0.0016)</td>
<td><strong>0.9998 (±0.0000)</strong></td>
<td>0.9997 (±0.0000)</td>
<td>0.9997 (±0.0000)</td>
<td>0.9997 (±0.0000)</td>
</tr>
<tr>
<td>NIST finger-finger</td>
<td><strong>1.0000 (±0.0000)</strong></td>
<td>0.9999 (±0.0000)</td>
<td>0.9994 (±0.0010)</td>
<td>0.9998 (±0.0000)</td>
<td>0.9998 (±0.0000)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU Time</th>
<th>OTERN</th>
<th>OSELM</th>
<th>Perceptron</th>
<th>PA</th>
<th>CW</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIST finger-face</td>
<td>24.50 (±0.20)</td>
<td>53.35 (±2.84)</td>
<td><strong>0.94 (±0.01)</strong></td>
<td>1.05 (±0.01)</td>
<td>7.42 (±0.09)</td>
</tr>
<tr>
<td>NIST face-face</td>
<td>372.25 (±3.66)</td>
<td>579.95 (±24.66)</td>
<td><strong>32.31 (±0.61)</strong></td>
<td>34.45 (±0.49)</td>
<td>245.84 (±1.04)</td>
</tr>
<tr>
<td>NIST finger-finger</td>
<td>1408.90 (±11.33)</td>
<td>942.83 (±9.06)</td>
<td><strong>124.47 (±1.24)</strong></td>
<td>163.63 (±4.25)</td>
<td>969.02 (±12.35)</td>
</tr>
</tbody>
</table>

4.3.2. **Computation efficiency**

**CPU time using sequential data.**

- **Comparison of batch-based TERN and online-based OTERN:** In Fig. 5, the training execution (CPU) time of batch-based and online-based OTERN methods are shown along the arrival of each training data samples. The experiments is performed for six different number of hidden neurons chosen within \{d, 2 \cdot d, 5 \cdot d, 10 \cdot d, 20 \cdot d, 40 \cdot d\} where \(d\) is the input feature dimension (number of individual biometric score). The batch-based TERN shows a much higher training CPU time than that of online-based OTERN in general. This is due to the re-computation of the entire data in the batch mode. As data is accumulated, the batch-based TERN shows a dramatically increase of training time. Moreover, the training time
increases when the number of hidden nodes increases. The proposed online-based OTERN shows a nearly constant training time as compared to the batch-based TERN method.

![Graph showing training CPU time comparison between online and batch methods.](image)

Figure 5: Training CPU time of batch-based and online based TERN (with different number of hidden nodes) versus number of data samples.

- **Comparison of OTERN and competing online-based state-of-the-arts**: Fig. 6 shows training CPU time along with the arrival of each training data sample. The proposed online OTERN method shows a comparable training execution time with the online-based state-of-the-art methods. The proposed online method spends more execution time approximately 70 µ (micro, 10^{-6}) seconds per sample than Perceptron method which shows minimum training CPU time among the compared online-based methods (OTERN, OSELM, Perceptron, PA, and CW).

**CPU time online-based methods using batch data.** In Table 4, we see that the proposed OTERN shows a comparable training time with those online-based state-of-the-art methods (OTERN, OSELM, Perceptron, PA, and CW).
4.3.3. Statistical significance test among the classifiers

In order to examine whether the classifiers are statistically different, a statistical significance test and post-hoc analysis can be applied. Firstly, for statistical significance, a Friedman test [25] is adopted. This approach utilizes a comparison of the ranks among the algorithms. The algorithms are ranked for each data set separately where the best performing algorithm is assigned to a rank 1, the second best rank 2, and so on. In case of ties, an average rank is assigned for each algorithm. Hypothesis is tested for $p$-value (here, we used Matlab function ‘friedman’) to check whether it is lower than a specific confidence level $\alpha$. If $p < \alpha$, the null-hypothesis that all algorithms are same is rejected. If the null-hypothesis is rejected, a post-hoc Nemenyi test can next be performed to observe the difference among the algorithms. If the rank difference between a pair of algorithms is larger than the critical difference (CD) at a certain confidence level, the pair of classifiers is considered significantly different.

Based on the above fusion verification performance results on both XM2VTS data sets and NIST–BSSR1 data sets, we examine the statistically difference test for five online learning algorithms (OTERN, OSELM, Perceptron, PA, and CW). In other words, total thirty five test results are
accumulated from XM2VTS data sets and NIST–BSSR1 data sets. Since the three NIST–BSSR1 data sets is too a small number for the significance test separately, the results from these three data sets are combined with the results from the thirty-two XM2VTS data sets. Under the Friedman test, the null-hypothesis is rejected at confidence level $\alpha = 0.01$ since the $p$-values for all HTER and Accuracy are respectively $3.8446 \times 10^{-20}$ and $4.2417 \times 10^{-17}$. Next, a post-hoc analysis is conducted where the critical difference (CD) is calculated as $1.23$ at $p = 0.01$. If the difference between a pair of the algorithms is larger than $1.23$, it is determined to be statistically different by Nemenyi test.

Fig. 7 (a) and (b) visualize the Nemenyi post hoc test at $p = 0.01$ following Demsár’s analysis [25]. The mean ranks over the data sets for each method are shown in descending rank order for the horizontal-axis. Statistically similar methods are grouped with horizontal bars in the figures. In HTER measure, the proposed OTERN method is clearly separated as shown in Fig. 7 (a). In Accuracy measure, OSELM method is seen to be clearly separated from all competing online algorithms as shown in Fig. 7 (b).

4.3.4. Summary of experimental results and discussions

Scores fusion verification performance. Our proposed online learning method is shown to have a comparable or better fusion verification accuracy performance with competing online-based methods. The performance results are shown in terms of HTER and Accuracy: (1) on thirty-two XM2VTS data sets in Fig. 4, (2) on three large scale NIST-BSSR1 data sets in Table 4.

The online-based OTERN shows a similar performance to that of the batch-based TERN in Fig. 4. This is because our proposed online learning solution converges to the solution of TERN when the number of training samples is large.

The statistical significant test shows that OTERN method is significantly better than those competing online learning methods in terms of verification HTER performance.

Computation efficiency. The proposed online-based TERN shows a nearly constant training time while the batch-based TERN method which shows a much higher training CPU time when the number of arriving data grows higher. The proposed method shows a comparable training execution time with competing online-based methods in Fig. 6.
Figure 7: Visualization of Nemenyi post-hoc test: (a) HTER and (b) Accuracy. This analysis is to check statistically significant differences between tested algorithms. The mean rank of the entire dataset for each method is plotted in the x-axis. Connected methods are not significantly different at $p = 0.01$.

5. Conclusion

In this paper, an online classification error rate minimization network was proposed for multibiometric fusion. This is different from those existing learning algorithms such as online sequential ELM [15] which aimed to optimize the residuals (distance errors) between the actual output and the predicted output. The main issue in formulating an online classification based learning is to update a class-specific weight which varies according to the new total number of training genuine-user (positive-class) and imposter (negative-class) data. Through a generalized inverse of re-scaling weight vector, an approximate online learning for classification
has been formulated. The convergence of the proposed online learning solution with respect to the original batch-based error rate optimization was analyzed. Extensive empirical evaluations were performed on publicly available score fusion data sets. The proposed online TER network algorithm was shown to operate on large scale multibiometric scores fusion scenarios. The proposed online method showed promising performances in terms of good fusion verification accuracy and low learning cost comparing with competing state-of-the-art methods.

As current investigation is limited to SLFNs with a fixed network architecture, we can extend our future research towards adding hidden nodes during classifier updates. This is inspired by an incremental ELM method reported recently [26].

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No.2011–0010938).

References

  URL http://www.itl.nist.gov/iad/894.03/biometricscores
  URL http://www.mathworks.com/products/matlab/

30
Appendix A. Convergence of OTERN

Appendix A.1. Convergence of OTERN solution based on sequential data

\[
\hat{\theta}_n^{\text{OTERN}} = \hat{\theta}_{n-1}^{\text{OTERN}} + M_n^{\text{OTERN}} h_n w_n \left( y_n - h_n^T \hat{\theta}_{n-1}^{\text{OTERN}} \right); \quad \text{(from the solution (20))}
\]

\[
R_n^{\text{OTERN}} \hat{\theta}_n^{\text{OTERN}} = R_{n-1}^{\text{OTERN}} \hat{\theta}_{n-1}^{\text{OTERN}} + R_n^{\text{OTERN}} M_n^{\text{OTERN}} h_n w_n \left( y_n - h_n^T \hat{\theta}_{n-1}^{\text{OTERN}} \right)
\]

\[
= (R_n^{\text{OTERN}} - h_n w_n h_n^T) \hat{\theta}_{n-1}^{\text{OTERN}} + h_n w_n y_n
\]

\[
= u_n R_{n-1}^{\text{OTERN}} \hat{\theta}_{n-1}^{\text{OTERN}} + h_n w_n y_n
\]

\[
= u_n R_{n-1}^{\text{OTERN}} \left( \hat{\theta}_{n-2}^{\text{OTERN}} + M_{n-1}^{\text{OTERN}} h_{n-1} w_{n-1} \left( y_{n-1} - h_{n-1}^T \hat{\theta}_{n-2}^{\text{OTERN}} \right) \right) + h_n w_n y_n
\]

\[
= u_n u_{n-1} R_{n-2}^{\text{OTERN}} \hat{\theta}_{n-2}^{\text{OTERN}} + u_n u_{n-1} w_{n-1} h_{n-1} y_{n-1} + w_n h_n y_n
\]

\[\vdots\]

\[
= u_n u_{n-1} \cdots u_2 w_1 h_1 y_1 + u_n u_{n-1} \cdots u_3 w_2 y_2 + \cdots + u_n w_{n-1} h_{n-1} y_{n-1} + w_n h_n y_n
\]

where

\[
R_n^{\text{OTERN}} = u_n u_{n-1} \cdots u_2 w_1 h_1^T + u_n u_{n-1} \cdots u_3 w_2 h_2^T + \cdots + u_n w_{n-1} h_{n-1}^T + w_n h_n h_n^T,
\]

\[
= \hat{\theta}_n^{\text{OTERN}}
\]

\[
= \left( \frac{u_n u_{n-1} \cdots u_2 w_1 h_1^T}{= w_n} + \frac{u_n u_{n-1} \cdots u_3 w_2 h_2^T}{= w_n} + \cdots + \frac{u_n w_{n-1} h_{n-1}^T}{= w_n} + w_n h_n h_n^T}{= w_n} \right)^{-1}
\]

\[
\times \left( \frac{u_n u_{n-1} \cdots u_2 w_1 h_1 y_1}{= w_n} + \frac{u_n u_{n-1} \cdots u_3 w_2 h_2 y_2}{= w_n} + \cdots + \frac{u_n w_{n-1} h_{n-1} y_{n-1}}{= w_n} + w_n h_n y_n}{= w_n} \right)
\]

\[
= (H_n W_n^{\text{OTERN}} H_n)^{-1} H_n W_n^{\text{OTERN}} Y_n,
\]
where

$$W_n^{\text{OTERN}} \in \left\{ \tilde{w}_n = \left( \prod_{k=i+1}^{n} u_k \right) w_i \in \mathbb{R} : i = 1, \ldots, n - 1, \text{ and } w_n : i = n \right\},$$

(A.4)

where \( w^*_i = \frac{1}{n_i^*} (n_i^*: \text{number of } \bullet \text{ class' samples until } i^\text{th} \text{ sequence})\),

$$u_k \in \left\{ \begin{array}{ll}
\frac{1}{2} \left( 1 + \frac{n^-}{w^-_{k-1}} \right) & \text{if } y_k = -1, (\therefore w^+_k = w^-_{k-1}), \\
\frac{1}{2} \left( \frac{w^+_k}{w^-_{k-1}} + 1 \right) & \text{if } y_k = +1, (\therefore w^-_{k-1} = w^+_{k-1})
\end{array} \right\},$$

(A.5)

and \( n = n^+ + n^- \). Here, \( \tilde{w}_n \) is an approximated updated weight.

Appendix A.2. OTERN’s weight convergence

As shown in (A.5), updating weight \( u_k \) has two alternative updatings which are dependent on class label of the \( k^\text{th} \) sequence. (A.4) can be re-written as:

$$\tilde{w}_n = \left( \prod_{k=i+1}^{n} u_k \right) w_i = \left( \prod_{y_k \in \{\text{class(-)}\}} u_k \right) \left( \prod_{y_k \in \{\text{class(+)}\}} u_k \right) w_i$$

(A.6)

where

$$\prod_{y_k \in \{\text{class(-)}\}} u_k = \frac{1}{2} \left( 1 + \frac{n^- - 1}{n^-} \right) \cdot \frac{1}{2} \left( 1 + \frac{n^- - 2}{n^- - 1} \right) \cdot \ldots,$$

(A.7)

$$\prod_{y_k \in \{\text{class(+)}\}} u_k = \frac{1}{2} \left( 1 + \frac{n^+ - 1}{n^+} \right) \cdot \frac{1}{2} \left( 1 + \frac{n^+ - 2}{n^+ - 1} \right) \cdot \ldots,$$

(A.8)

with \( \frac{w^-_{n-1}}{w^-_{n}} = \frac{n^- - 1}{n^-}, \frac{w^+_n}{w^+_n} = \frac{n^+ - 1}{n^+}, n^- + n^+ = n \), and \( p_1 + p_2 = i \).

Through observing convergence of \( \prod_{k=i+1}^{n} u_k \) and \( w_i \) when \( n \) goes to large value (or infinite), we shall observe the convergence of \( \tilde{w}_n \).

1) Convergence of \( \prod_{k=i+1}^{n} u_k \):

(a) If \( n \to \infty \), then \( \frac{n-1}{n} \to 1 \).
(b) If \( k \to \infty \) then \( \prod_{k=i+1}^{n} u_k \to 1. \)

(c) If \( k \to 1 \), then \( \prod_{k=i+1}^{n} u_k \to \epsilon_1 \) \( (0 < \epsilon_1 < 1) \).

2) Convergence of \( w_i \):

   (a) \( \|w_{n-1} - w_n\| < \epsilon_2 \) for \( \epsilon_2 > 0 \).

   (b) If \( n \to \infty \), \( \epsilon_2 \to 0 \) (\( \because w_n = \frac{1}{n} \)), so does \( w_{n-1} \to w_n \).

3) Trend of \( \tilde{w}_n \): set a point \( \nu \) \( (1 < \nu < \infty) \) which is an arbitrary boundary point during convergence of \( \prod_{k=i+1}^{n} u_k \),

   a. If \( 1 \leq i < \nu \), then \( \tilde{w}_n = \epsilon_1 w_i \) \( (0 < \epsilon_1 < 1) \),

   b. If \( \nu \leq i \leq n \) \( (n \to \infty) \), then \( \tilde{w}_n \approx w_i \approx w_n \) (\( \because 1)-(a) \) and 2)-(b)).

   Hence,

\[
\tilde{w}_n^{(y_i)} \approx w_n \in \{ w_n^-, w_n^+ \}. \quad \text{(A.9)}
\]

where \( 0 < \tilde{w}_n < 1 \). (See Fig. 2 (b) and Fig. 2 (c).)

As a result, we can conclude that \( \theta_n^{\text{OTERN}} \approx \theta_n^{\text{TERN}} \). \( \square \)
Brief Biography

Youngsung Kim received the B.S. and M.S. degrees in electrical and electronic engineering from Yonsei University, Seoul, Korea in 2006 and 2008 respectively. He is currently a Ph.D. candidate in the School of Electrical and Electronic Engineering at Yonsei University. His research interests include pattern recognition, machine learning, and biometrics.

Kar-Ann Toh is a full professor in the School of Electrical and Electronic Engineering at Yonsei University, South Korea. He received the PhD degree from Nanyang Technological University (NTU), Singapore. He worked for two years in the aerospace industry prior to his post-doctoral appointments at research centres in NTU from 1998 to 2002. He was affiliated with Institute for Infocomm Research in Singapore from 2002 to 2005 prior to his current appointment in Korea. His research interests include biometrics, pattern classification, optimization and neural networks. He is a co-inventor of a US patent and has made several PCT filings related to biometric applications. Besides being an active member in publications, Dr. Toh has served as a member of technical program committee in international conferences related to biometrics and artificial intelligence. He is currently an associate editor of Pattern Recognition Letters and a senior member of the IEEE.

Andrew Beng Jin Teoh obtained his BEng (Electronic) in 1999 and Ph.D degree in 2003 from National University of Malaysia. He is currently an assistant professor in EE Department, College Engineering of Yonsei University, South Korea. His research interest is in biometrics security and pattern recognition. He had published around 180 refereed international journal and conference papers in his area.

How-Lung Eng received the B.Eng. and Ph.D. degrees from Nanyang Technological University, Singapore, in 1998 and 2002, respectively, both in electrical and electronic engineering. Currently, he is with the Institute for Infocomm Research, Singapore, as a Senior Research Fellow. His research interest includes real-time vision, pattern classification, and machine learning for abnormal event detection. He has made several PCT filings related to video surveillance applications and has actively published his works in the above areas of interest. Dr. Eng was a recipient of the 2000 Tan Kah Kee Young Inventors Award (Silver, Open Section) for his Ph.D. work, and a recipient of the 2002 TEC Innovator Award, and the 2006 and 2008 IES Prestigious Engineering Awards for his works in the areas of visual surveillance.
Dr. Wei-Yun Yau received his BEng (Electrical) from the National University of Singapore (1992), MEng degree (1995) and PhD degree (1999) from the Nanyang Technological University. From 1997 to 2002, he was a Research Engineer and then Program Manager at the Centre for Signal Processing, Singapore leading the research and development effort in biometrics signal processing. His team won the top 3 positions in both speed and accuracy at the international Fingerprint Verification Competition 2000 (FVC2000). Currently, he is a Programme Manager with the Institute for Infocomm Research, leading the research in Interactive Social Tele-Experience programme. Wei-Yun also serves as a member of the IAPR’s Technical Committee on Biometrics (TC4) and an Exco member of the Asian Bio-metric Consortium. He is also currently the Chair of the IPTV Working Group, Singapore and the Chair of Biometrics Technical Committee, Singapore. He was also the project editor of ISO/IEC JTC1 SC37 29794-4 on fingerprint quality score normalization and co-editor for ISO/IEC JTC1 SC37 29794-1. Wei-Yun is the recipient of the TEC Innovator Award 2002, the Tan Kah Kee Young Inventors Award 2003 (Merit), Standards Council Merit Award 2005, IES Prestigious Engineering Achievement Awards 2006 and the Standards Council Distinguished Award 2007. His research interest includes biometrics, active vision system, personalized media and interactive visual interface and has published widely, with 8 patents granted and 100 publications in these areas. A paper he co-published in 2008 received the Pattern Recognition Journal Honorable Mention 2010.
Authors’ photos

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Youngsung Kim" /></td>
<td><img src="image2.png" alt="Kar-Ann Toh" /></td>
</tr>
<tr>
<td>Youngsung Kim</td>
<td>Kar-Ann Toh</td>
</tr>
<tr>
<td><img src="image3.png" alt="Andrew Beng Jin Teoh" /></td>
<td><img src="image4.png" alt="How-Lung Eng" /></td>
</tr>
<tr>
<td>Andrew Beng Jin Teoh</td>
<td>How-Lung Eng</td>
</tr>
<tr>
<td><img src="image5.png" alt="Wei-Yun Yau" /></td>
<td></td>
</tr>
<tr>
<td>Wei-Yun Yau</td>
<td></td>
</tr>
</tbody>
</table>