Binary Spectrum Feature for Improved Classifier Performance

Nalika Ulapane[1], Karthick Thiyagarajan [2] and Sarath Kodagoda[2]

Abstract— **Classification has become a vital task in modern machine learning and Artificial Intelligence applications, including smart sensing. Numerous machine learning techniques are available to perform classification. Similarly, numerous practices, such as feature selection (i.e., selection of a subset of descriptor variables that optimally describe the output), are available to improve classifier performance. In this paper, we consider the case of a given supervised learning classification task that has to be performed making use of continuous-valued features. It is assumed that an optimal subset of features has already been selected. Therefore, no further feature reduction, or feature addition, is to be carried out. Then, we attempt to improve the classification performance by passing the given feature set through a transformation that produces a new feature set which we have named the "Binary Spectrum". Via a case study example done on some Pulsed Eddy Current sensor data captured from an infrastructure monitoring task, we demonstrate how the classification accuracy of a Support Vector Machine (SVM) classifier increases through the use of this Binary Spectrum feature, indicating the feature transformation's potential for broader usage.**

I. Introduction

Supervised learning in regards to classification builds models of the distribution of given class labels in terms of given predictor variables (or features). The learned models (known as Classifiers) can then serve to assign class labels to provided testing instances where the predictor variable values (or features) are known, but the class labels are unknown [1]. Performing classification in such manner has become a common and vital component in the modern-day use of Artificial Intelligence. Many techniques such as Decision Trees, Discriminant Analysis, Perceptron-based techniques (e.g., Neural Networks), Logistic Regression, Bayesian Networks, Instance based learning (e.g., Nearest Neighbour Classifiers), Ensemble Classifiers, and Support Vector Machines (SVM) have been developed to learn and perform classification [1], [2], [3], [4]. More recently, deep learning based classification techniques too have been developed [5].

Over-fitting, lack of accuracy, and computation cost are some of the commonly encountered challenges when developing classifiers. It can be understood that over-fitting and computation cost become issues especially when working with high dimensional data. Feature selection (or feature reduction) is a commonly followed practice to overcome the curse of dimensionality. That practice in return does on occasions help alleviate some of over-fitting and accuracy-related issues as well. Following literature, one can categorize the methods available for feature reduction to be three-fold: (1) Filter methods; (2) Wrapper methods; and (3) Embedded methods [6], [7]. In our paper, we consider the case where an optimal feature selection has already been carried out. That means, we focus on a supervised learning classification task that has to be performed with a given set of features, with no further feature reduction or feature addition being allowed. We assume the features to be real and continuous-valued for this paper. Now suppose there is some benchmark accuracy that can be achieved by using the feature set as it is. Then, we ask the question, whether that benchmark accuracy can be overtaken by performing some transformation to the existing feature set. We contribute in this paper by answering that question, via introducing a feature transformation we name the "Binary Spectrum feature transformation".

Derivation of the Binary Spectrum feature is presented in detail in this paper. Following derivation, we demonstrate the effectiveness of this Binary Spectrum transformation by benchmarking accuracy, and overtaking that benchmarked accuracy of an SVM-based classification task. Classification is performed on some Pulsed Eddy Current (PEC) sensor data. This dataset has been collected from an automated infrastructure monitoring exercise performed on a ferromagnetic critical water pipe [8], [9]. The class labels are reflective of the thickness of the pipe wall on points where sensing has been done. Descriptor variables happen to be some real continuous-valued features extracted from the corresponding PEC signals [8], [9], [10], [11].

The Binary Spectrum feature transformation happens to transform a given set of real continuous-valued features to a set containing both continuous-valued and discrete-valued categorical-like data (i.e., binary data). This categorical-like data subset is derived from the original continuous-valued dataset. The rationale behind the evident improvement in classification accuracy resulting from this transformation can be argued to be the effect of this combination of two data-types: (1) The natural continuous-valued features; (2) Categorical-like component derived from the natural features.

The structure of the paper is as follows: Section II mathematically formulates the problem of improving the accuracy of a given classifier, or a classification task; Section III presents the derivation of the Binary Spectrum feature transformation and presents an algorithm to find best performing classifiers; Section IV presents the effectiveness of the proposed method via a demonstrative example performed on

[1]Nalika Ulapane is with the Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville VIC 3010, Australia. nalika.ulapane@unimelb.edu.au & nalika.ulapane@gmail.com

[2]Karthick Thiyagarajan and Sarath Kodagoda are with the UTS Robotics Institute, University of Technology Sydney, Ultimo NSW 2007, Australia. Karthick.Thiyagarajan@uts.edu.au, Sarath.Kodagoda@uts.edu.au

Corresponding Author: Karthick Thiyagarajan

PEC sensing data, and Section V concludes the paper by discussing the implications of the results, limitations of this study, and potential avenues for future work.

II. Problem Formulation

We consider a binary classification (i.e., two-class classification) supervised learning problem that has to be solved making use of continuous-valued features. As such, let there be a given binary classifier, trained by the training data $X_t \in \mathbb{R}^{a\times b}$ and $Y_t \in \mathbb{B}^a$ where $\mathbb{B}^a$ denotes an $a\times 1$ vector of binary digits. We make the following assumption about the training data.

Assumption 1: The training features (i.e., X_t) is an optimal subset of training features, i.e., no further feature reduction or feature addition is to be done.

Assumption 2: The two classes in the set of training labels (i.e., Y_t) are evenly (or equally) populated, i.e., there is approximately a 50:50 population ratio between the two classes.

The vector $Y_t \in \mathbb{B}^a$ containing training labels (or the training targets) is given by

$$Y_t = \begin{bmatrix} y_{t1} & y_{t1} & \dots & y_{t1} & y_{t2} & y_{t2} & \dots & y_{t2} \end{bmatrix}^T_{1\times a} \quad (1)$$

where $y_{t1} = 0$, $y_{t2} = 1$, and $[*]^T$ denotes matrix transpose. The corresponding training features contained in $X_t (\in \mathbb{R}^{a\times b})$ are given by

$$X_t = \begin{bmatrix} x_{t11} & x_{t12} & \dots & x_{t1j} & \dots & x_{t1b} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{ti1} & x_{ti2} & \dots & x_{tij} & \dots & x_{tib} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{ta1} & x_{ta2} & \dots & x_{taj} & \dots & x_{tab} \end{bmatrix}_{a\times b} \quad (2)$$

where $i, j \in \mathbb{Z}^+$ are generic subscript notations where $1 \le i \le a$ and $1 \le j \le b$.

Similarly, the testing dataset on which the classifier is to make predictions, is given by the corresponding matrices $X_{te} \in \mathbb{R}^{c\times b}$ and $Y_{te} \in \mathbb{B}^c$.

$$Y_{te} = \begin{bmatrix} y_{te1} & y_{te1} & \dots & y_{te1} & y_{te2} & y_{te2} & \dots & y_{te2} \end{bmatrix}^T_{1\times c} \quad (3)$$

Here, $y_{te1} = 0$, $y_{te2} = 1$.

$$X_{te} = \begin{bmatrix} x_{te11} & x_{te12} & \dots & x_{te1j} & \dots & x_{te1b} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{tei1} & x_{tei2} & \dots & x_{teij} & \dots & x_{teib} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{tec1} & x_{tec2} & \dots & x_{tecj} & \dots & x_{tecb} \end{bmatrix}_{c\times b} \quad (4)$$

$i, j \in \mathbb{Z}^+$ here are generic subscript notations where $1 \le i \le c$ and $1 \le j \le b$.

With this data, we define the following operations $o(X_t) = X_t u$ and $o(X_{te}) = X_{te} u$ with respect to $u \in \mathbb{R}^{b\times b}$. u here is an orthonormal basis of X_t.

Now suppose, a classifier trained with the above defined data (i.e., $o(X_t), Y_t$) is given, and it is denoted as C, $C: \mathbb{R}^{d\times b} \to \mathbb{B}^d$. This classifier would predict the classes (i.e., 0 or 1) for the testing data X_{te}. The prediction output will come in a vector $\hat{Y_{te}} \in \mathbb{B}^c$ given as follows.

$$\hat{Y_{te}} = C(o(X_{te})) \quad (5)$$

The vector err is then defined in order to compute the classification accuracy.

$$err = \hat{Y_{te}} - Y_{te} \quad (6)$$

Locations of err corresponding to instances of a correct prediction having been made would carry zeros. Therefore, we define the total number of zeros in err as sum_0. With that we define the classification accuracy acc in the following manner.

$$acc = \frac{sum_0}{c} \times 100\% \quad (7)$$

As such, acc can be represented as a function in the following manner.

$$acc = g(X_t, Y_t, X_{te}, Y_{te}) \quad (8)$$

The objective now will be to increase the classification accuracy (i.e., to increase acc) using the same set of features X_t and X_{te} without any reduction or addition of features respecting **Assumption 1**. Accomplishing this objective under **Assumption 1** is the reason for this paper introducing the Binary Spectrum feature transformation.

III. Deriving the Binary Spectrum Feature

Binary Spectrum transformation is performed by applying a function $f, f: \mathbb{R}^{e\times b} \to \mathbb{B}^{e\times bn}$, $n \in \mathbb{Z}^+$ to features X_t and X_{te} in the following manner.

$$X_{tb} = \begin{bmatrix} X_t & f(X_t, n) \end{bmatrix} \quad (9)$$

$$f(X_t, n) = \begin{bmatrix} f(x_{t11}, n) & \dots & f(x_{t1j}, n) & \dots & f(x_{t1b}, n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ f(x_{ti1}, n) & \dots & f(x_{tij}, n) & \dots & f(x_{tib}, n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ f(x_{ta1}, n) & \dots & f(x_{taj}, n) & \dots & f(x_{tab}, n) \end{bmatrix} \quad (10)$$

$$X_{teb} = \begin{bmatrix} X_{te} & f(X_{te}, n) \end{bmatrix} \quad (11)$$

$$f(X_{te}, n) = \begin{bmatrix} f(x_{te11}, n) & \dots & f(x_{te1j}, n) & \dots & f(x_{te1b}, n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ f(x_{tei1}, n) & \dots & f(x_{teij}, n) & \dots & f(x_{teib}, n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ f(x_{tec1}, n) & \dots & f(x_{tecj}, n) & \dots & f(x_{tecb}, n) \end{bmatrix} \quad (12)$$

Here, the matrix sizes come as $a \times b(n+1)$ for X_{tb} and $c \times b(n+1)$ for X_{teb}. Also, $f(x_{tij}, n)$ and $f(x_{teij}, n)$ for any $x_{tij} \in \mathbb{R}$ or $x_{teij} \in \mathbb{R}$, is defined as

$$f(x_{i,j}, n) = \begin{bmatrix} . & . & . & . & . & . & . & . & . \end{bmatrix}_{1\times n} \quad (13)$$

where f does some scaling to $x_{i,j}$ and rounds it off to the nearest integer (discussed in the remainder of this section), and produces $\begin{bmatrix} . & . & . & . & . & . & . & . & . \end{bmatrix}$, which is the binary value of the scaled and rounded $x_{i,j}$ given to n bits.

To perform the transformation done by f for a prescribed number of bits (i.e., n), we first scale $x_{i,j}$ values to remain within the two bounds l_{low} and l_{up} defined as follows.

$$l_{low} = 0 \tag{14}$$

$$l_{up} = 2^n - 1 \tag{15}$$

Now consider the example where the training data point x_{tij} is to be scaled. To scale x_{tij}, governed by the column subscript of x_{tij}, we select the j^{th} column of the corresponding training feature matrix X_t. The j^{th} column symbolized as $X_{tj}| \in \mathbb{R}^a$ comes as follows.

$$X_{tj}| = \begin{bmatrix} x_{t1j} & \dots & x_{tij} & \dots & x_{taj} \end{bmatrix}^T_{1\times a} \tag{16}$$

The minimum and maximum values contained within $X_{tj}|$ are denoted as $\min(X_{tj}|)$ and $\max(X_{tj}|)$ respectively. With those, we define the scaling of any training feature value x_{tij} as follows:

$$x_{tijs} = l_{low} + \frac{x_{tij} - \min(X_{tj}|)}{\max(X_{tj}|) - \min(X_{tj}|)}(l_{up} - l_{low}) \tag{17}$$

where x_{tijs} is the scaled value of x_{tij}.

Now consider the case of scaling testing feature values in X_{te}. To scale any testing feature value x_{teij}, we consider the same $\min(X_{tj}|)$ and $\max(X_{tj}|)$ coming from $X_{tj}|$. This selection is governed by the column subscript of x_{teij}.

With those, we define the scaling of any testing feature value x_{teij} as follows:

$$x_{teijs} = l_{low} + \frac{x_{teij} - \min(X_{tj}|)}{\max(X_{tj}|) - \min(X_{tj}|)}(l_{up} - l_{low}) \tag{18}$$

where x_{teijs} is the scaled value of x_{teij}.

When scaling testing feature values using (18), there is a chance of some scaled values lying outside the bounds specified by l_{low} and l_{up}. That is a limitation in this scaling method and to alleviate some of the adversity caused by outliers, for all i, j where $x_{teijs} < l_{low}$ we assign

$$x_{teijs} \leftarrow l_{low} \tag{19}$$

and for all i, j where $x_{teijs} > l_{up}$ we assign as follows.

$$x_{teijs} \leftarrow l_{up} \tag{20}$$

Following the assignments of (19) and (20), all training and testing feature values in X_t and X_{te} would have been scaled to map within the lower and upper bounds prescribed by l_{low} and l_{up} in (14) and (15).

Rounding the scaled training feature (x_{tijs}) and testing feature (x_{teijs}) values to their nearest integers, and converting the rounded numbers to binary, and representing the binary values in n bits is how the Binary Spectrum vectors symbolized by $\begin{bmatrix}\dots\dots\dots\end{bmatrix}$ in (13) are formed. Substituting the Binary Spectrum vectors constructed in that manner in (10), (12) and (9), (11) would yield the Binary Spectrum matrices X_{tb} and X_{teb}.

With this data, we define the following operations $o(X_{tb}) = X_{tb}v$ and $o(X_{teb}) = X_{teb}v$ with respect to $v \in \mathbb{R}^{b(n+1)\times b(n+1)}$. v here is an orthonormal basis of X_{tb}.

A new classifier C_n of the form $C_n, C_n : \mathbb{R}^{d\times b(n+1)} \rightarrow \mathbb{B}^d$ can now be trained with the training dataset $o(X_{tb})$ and Y_t. This classifier would predict the classes for the testing data X_{teb}. The prediction output will come in a vector $\hat{Y_{teb}} \in \mathbb{B}^c$ given as follows.

$$\hat{Y_{teb}} = C_n(o(X_{teb})) \tag{21}$$

The vector $errb$ can then be defined in order to compute the classification accuracy.

$$errb = \hat{Y_{teb}} - Y_{te} \tag{22}$$

Similar to the vector err in (6), locations of $errb$ corresponding to instances of a correct prediction having been made would carry zeros. Therefore, we define the total number of zeros in $errb$ as $sumb_0$. With that we define the classification accuracy $accb$ in the following manner.

$$accb = \frac{sumb_0}{c} \times 100\% \tag{23}$$

As such, $accb$ can be represented as a function in the following manner, similar to the function representation of acc shown in (8).

$$accb = g\left(\begin{bmatrix}X_t & f(X_t, n)\end{bmatrix}, Y_t, \begin{bmatrix}X_{te} & f(X_{te}, n)\end{bmatrix}, Y_{te}\right) \tag{24}$$

Now suppose a classifier C_n for some $n \in \mathbb{Z}^+$ can be found such that the condition $accb > acc$ is satisfied. Then, our objective of increasing the classifier accuracy will be accomplished, without removing any features from or adding new features to the feature sets X_t and X_{te}, i.e., by satisfying **Assumption 1**. As opposed to reducing or increasing the feature sets X_t and X_{te}, what enables superior classification accuracy in the proposed method will be the Binary Spectrum transformation of existing features.

Recall now the function representation of acc and $accb$ given in (8) and (24) respectively. With those, the ultimate solution one can seek following this method can be expressed in the following manner

$$[n^*, C_{n^*}] = \arg\max_n (accb - acc) \tag{25}$$

subject to the constraints $accb > acc$ and $n < n_{max}$, where $n_{max} \in \mathbb{Z}^+$ is some meaningful maximum number of bits to be allowed. The selection of an ideal value for n_{max} is an open question for the time being, and users of this method have freedom to experiment. An initial constraint some might hypothesize for n_{max} may be

$$n_{max} < \frac{a}{b} - 1 \tag{26}$$

with the objective of keeping the Binary Spectrum training feature matrix X_{tb} in (9) a tall and skinny matrix, assuming $a > b$ provided the training dataset in original form (i.e., X_t in (2)) has a instances and b features.

Finding an optimal solution n^* by solving (25) will result in an optimal Binary Spectrum transformation $f(X_t, n^*)$ and a classifier C_{n^*} trained from $o(X_{tb})$ and Y_t, that performs superior to the classifier C trained from sets $o(X_t)$ and Y_t. Thus, the objective of increasing classification accuracy will

be accomplished via the Binary Spectrum transformation. As a preliminary effort, we propose **Algorithm** 1 to find n^* and corresponding C_{n^*} iteratively.

Algorithm 1: Find n^* and C_{n^*} iteratively.

Result: n^*, C_{n^*}
$n^* \leftarrow 0$;
$C_{n^*} \leftarrow C$, C is trained with $o(X_t)$, Y_t;
$\hat{Y_{te}} \leftarrow C(o(X_{te}))$;
$acc \leftarrow acc$, calculate acc of C from (7);
$n \leftarrow 1$;
$n_{max} \leftarrow n_{max}(\in \mathbb{Z}^+)$, $n_{max} > 1$;
while $n \leq n_{max}$ **do**
  $X_{tb} \leftarrow \begin{bmatrix} X_t & f(X_t, n) \end{bmatrix}$;
  train C_n with $o(X_{tb})$, Y_t;
  $X_{teb} \leftarrow \begin{bmatrix} X_{te} & f(X_{te}, n) \end{bmatrix}$;
  $\hat{Y_{teb}} \leftarrow C_n(o(X_{teb}))$;
  calculate $accb$ of C_n from (23);
  if $accb > acc$ **then**
    $acc \leftarrow accb$;
    $n^* \leftarrow n$;
    $C_{n^*} \leftarrow C_n$;
  end
  $n \leftarrow n + 1$;
end

IV. Demonstrative Example, Experiments & Results

In this section we demonstrate how the proposed Binary Spectrum feature (or transformation) improves classification performance. We consider a Support Vector Machines (SVM) binary (or two-class) classifier working with two features (or descriptor variables), i.e., $X \in \mathbb{R}^{h\times 2}$, $Y \in \mathbb{B}^h$.

A. The Dataset

The dataset used for this work consists 8,400 Pulsed Eddy Current (PEC) signal measurements captured on different wall thickness values of grey cast iron. The dataset has been collected through works [8], [9], [10], [12], [13], [14]. The class labels (in Y) are decided based on the wall thickness (measured in mm). To respect **Assumption 2** (i.e., to have approximately 50:50 population split for the two classes in the training dataset), the cut off thickness value was chosen to be 23.3 mm after examining the data. Thickness values less than or equal to 23.3 mm are considered as Class 1 having class label '0'. Class 1 has 4,169 instances accounting to 49.63% of the total population. Thickness values greater than 23.3 mm are considered as Class 2 having class label '1'. Class 2 has 4,231 instances accounting to 50.37% of the total population. The thickness histogram (in percentage frequency) of this total dataset is shown in Fig. 1.

The input set (i.e., X) has two real-valued feature vectors. That means the domain of X becomes $X \in \mathbb{R}^{h\times 2}$ to be corresponding to the vector of labels $Y \in \mathbb{B}^h$. These features have been extracted from time domain PEC signals [10], [8], [9]. Shown in Fig. 2 is a scatter of the two features corresponding to all 8,400 measurements (or instances) in the total dataset. The two classes (i.e., Class 1 and Class 2) are colour coded in Fig. 2.

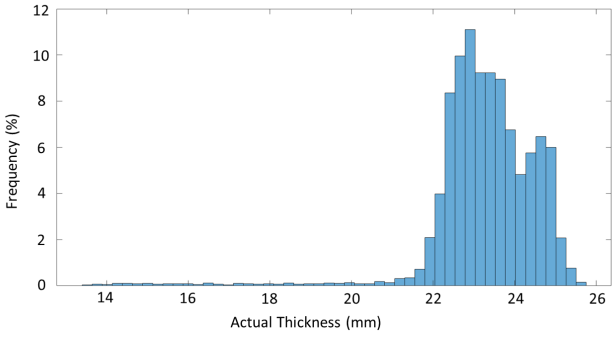

Fig. 1. Thickness histogram of the total data population, in percentage frequency.

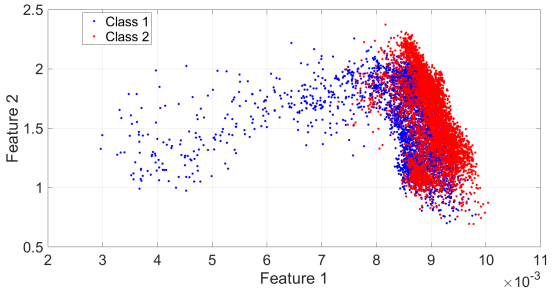

Fig. 2. Scatter between the two features in X corresponding to the full dataset (i.e., 8,400 instances), with Class 1 and Class 2 colour coded.

B. Splitting Training and Testing Sets

Authors assumed that only 30% of the total dataset will be available for training. To start with, authors performed random nonstratified partitioning of the 8,400 measurements, to a 30:70 split, 100 times. This means, the authors would have, 100 subsets of 30% of the total dataset, and 100 corresponding subsets of 70% of the total dataset. This 100-fold splitting, would provide authors with 100 trials to assess classifier performance. The authors' intention was to test the 100 trials separately. That is, to learn 100 classifiers, and perform 100 corresponding validations. If the 100 classifiers and the accuracy of the 100 corresponding validations statistically exhibit some convergence, that would imply the success (or the unsuccessfulness) of the work of this paper.

As an example, Fig. 3 shows the thickness histogram (with percentage frequency) of the training set (i.e. 30% subset) of the 100$^{\text{th}}$ trial (or partitioning). This dataset has 2,520 instances, and the histogram is comparable to the thickness histogram (with percentage frequency) of the total population (i.e., 100% of the data) shown in Fig. 1. Having comparable distributions as such is expected for model training / testing exercises, and it appeared that with the availability of a total of 8,400 measurements (or instances), random nonstratified

partitions of 30% would usually have distributions comparable with the total population. This observation was common among all the obtained partitions.

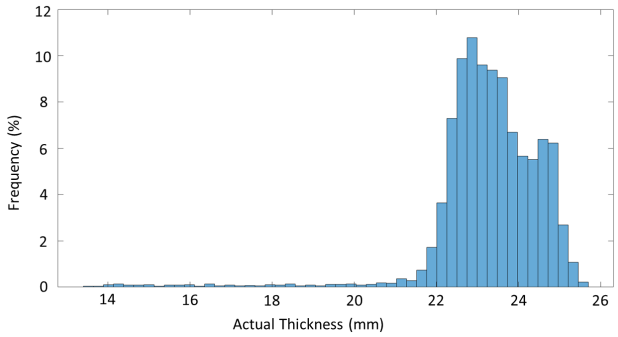

Fig. 3. Training data (i.e., 2,520 instances), thickness histogram of the 100th trial, in percentage frequency.

Shown in Fig. 4 is a scatter of the two features corresponding to the 2,520 measurements (or instances) in the training dataset that came from the 100th trial (or partitioning). The two classes (i.e., Class 1 and Class 2) are colour coded in Fig. 4. The distribution of instances in Fig. 4 is comparable to that of Fig. 2, indicating that the training dataset has a distribution that is comparable with the total population. This observation was common among all the obtained partitions. Further, this training dataset in Fig. 4 had 1,273 instances (i.e., 50.52%) in Class 1, and 1,247 instances (i.e., 49.48%) in Class 2, indicating that the training sample is in accordance with **Assumption 2** (i.e., the training dataset having approximately a 50:50 split among the two classes). This compliance with **Assumption 2** as well, was common among all the obtained partitions.

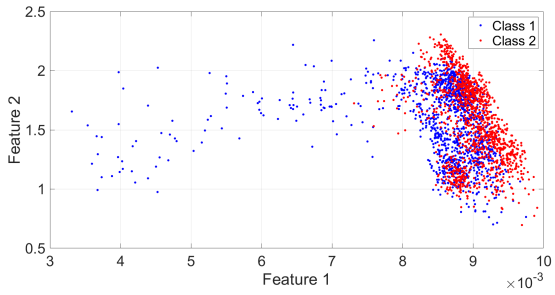

Fig. 4. Training data (i.e., 2,520 instances) of the 100th trial, scatter between the two training features, with Class 1 and Class 2 colour coded.

In all the 100 trials, the 100% of the data (i.e., the total population) was used as the testing dataset. Assessing classifier performance in that manner on a single large dataset makes the performance of each classifier statistically comparable.

C. Benchmarking Classification Accuracy with Features' Original Form

SVM is used for classification. The objective is to first benchmark the performance of an SVM classifier over the 100 trials (described in subsection IV-B) with the features in their original decimal form (i.e., prior to performing Binary Spectrum transformation). The subsequent intention is to assess the performance of an SVM classifier trained by Binary Spectrum features over the same 100 trials. In this subsection, we present the former analysis, i.e., benchmarking performance of an SVM classifier trained with the original form of the features over the 100 trials.

Given the non-linear nature of the data, the Gaussian kernel was chosen for SVM classification. The commonly known hyper-parameters named the Box Constraint and the Kernel Scale were set to be optimized at training (done with the 30% splits explained in subsection IV-B). The $o(X)$ features (or the predictor variables) were made to standardize before being fed to the classifier. On every trial, the initial value given to both those parameters (i.e., Box Constraint and Kernel Scale) was 1. The optimized classifier resulting from every trial was then evaluated with the testing data (i.e., the 100%, or the total dataset as mentioned in subsection IV-B). The *acc* value (recall from (7)) for each of the 100 trials was recorder to serve as the metric for performance evaluation. Depicted by the broken black line in Fig. 5 are the *acc* values resulting from the 100 trials of training a classifier with the raw values of features.

D. Evaluating Classification Accuracy with Binary Spectrum Features

The *accb* value (recall from (23)) of the best performing classifier (i.e., C_{n^*} identified from **Algorithm** 1) for each of the 100 trials (the same ones benchmarked in subsection IV-C) was recorder. These *accb* values would serve as the metric for performance evaluation of the Binary Spectrum feature. Depicted by the solid black line in Fig. 5 are the *accb* values recorded likewise from the corresponding 100 trials. As for the preliminary work reported in this paper, n_{max} (recall from **Algorithm** 1) was set to 10. Greater n_{max} values as well can be evaluated. Selection of SVM kernel, hyper-parameter initialization, and the training procedure (now considering the Binary Spectrum feature), were identical to those described in subsection IV-C).

As evident from Fig. 5, it was possible to achieve superiorly performing classifiers on every single trial by using the Binary Spectrum feature. On average, an improvement of 1.46% in classification accuracy (calculated as the average of $(accb - acc)/acc \times 100\%$) was observed across the 100 trials. The maximum improvement of classification accuracy was 3.49% in the 59th trial. Minimum observed improvement of classification accuracy was 0.1% in the 9th trial.

Depicted in Fig. 6 is the variation of the *accb* of all 100 trails across the 10 bits (i.e. $n_{max} = 10$ for the work reported in this paper). There are 100 line graphs in Fig. 6 spanning across 1 through 10. Each line graph corresponds to a trial and illustrates the variation of the trial's *accb*. It can be observed that there is a clear trend of reduction of accuracy past around the $5 \sim 7$ bit mark. Whether this downward trend persists for greater number of bits was not investigated at this stage. What we intend to report as a finding, is the fact

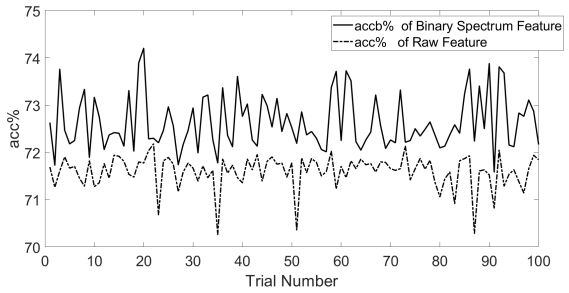

Fig. 5. Variation of classification accuracy over the 100 trials.

that it is possible to increase the classification accuracy of a given supervised learned classifier with a given fixed set of continuous-valued features, by using the proposed Binary Spectrum feature (or transformation).

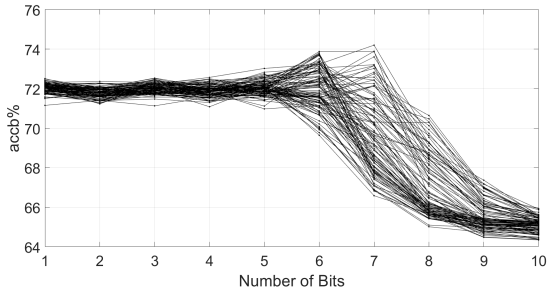

Fig. 6. Variation of *accb%* classification accuracy of all trials across 10 bits (i.e., n_{max} set to 10).

V. Conclusions

The case of improving classification accuracy for a given supervised learning classification task that has to be performed with a given reduced set of continuous-valued features was considered. The case imposes that further reduction of features or addition of new features is not possible. All the provided features have to be used. It was shown via a demonstrative binary classification (i.e., two-class classification) example, that it is possible to increase classification accuracy within the considered premise, via a feature transformation that produces a novel feature set the authors have named the "Binary Spectrum feature". An increase of classification accuracy by about 1.46%($\approx$ 1.5%) was observed for the considered example following Binary Spectrum transformation. The derivation of the Binary Spectrum feature was presented in detail along with a preliminary algorithm to identify best performing classifiers. The findings indicate potential for broader usage of the Binary Spectrum feature and may provoke interest for further investigation.

Limitations of this study include the following: (1) Only a binary classification task was examined (i.e., multi-class classification was not examined); (2) The descriptor variables were imposed to be continuous-valued (i.e., the more general case of both continuous-valued and categorical descriptor variables being present was not considered); (3) Class populations were imposed to be even (i.e., the case of uneven class populations was not examined); (4) The feature set of the case study included only two feature vectors (i.e., a higher dimensional example was not examined). As such, future work can investigate on relaxing some of the assumptions imposed on this work. Performance evaluation of the Binary Spectrum transformation on more sophisticated classification tasks that involve multiple classes and higher dimensional data also remains unchecked.

References

[1] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.

[2] S. Pang, S. Ozawa, and N. Kasabov, "Incremental linear discriminant analysis for classification of data streams," *IEEE transactions on Systems, Man, and Cybernetics, part B (Cybernetics)*, vol. 35, no. 5, pp. 905–914, 2005.

[3] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in neural information processing systems*, 2002, pp. 841–848.

[4] J. Kodovsky, J. Fridrich, and V. Holub, "Ensemble classifiers for steganalysis of digital media," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 432–444, 2011.

[5] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected topics in applied earth observations and remote sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

[6] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for svms," in *Advances in neural information processing systems*, 2001, pp. 668–674.

[7] S. Maldonado and J. López, "Dealing with high-dimensional class-imbalanced datasets: Embedded feature selection for svm classification," *Applied Soft Computing*, vol. 67, pp. 94–105, 2018.

[8] N. Ulapane, A. Alempijevic, T. Vidal Calleja, and J. Valls Miro, "Pulsed eddy current sensing for critical pipe condition assessment," *Sensors*, vol. 17, no. 10, p. 2208, 2017.

[9] N. Ulapane, A. Alempijevic, J. V. Miro, and T. Vidal-Calleja, "Non-destructive evaluation of ferromagnetic material thickness using pulsed eddy current sensor detector coil voltage decay rate," *NDT & E International*, vol. 100, pp. 108–114, 2018.

[10] A. M. N. N. B. Ulapane, "Nondestructive evaluation of ferromagnetic critical water pipes using pulsed eddy current testing," Ph.D. dissertation, University of Technology Sydney, 2016.

[11] N. Ulapane and L. Nguyen, "Review of pulsed-eddy-current signal feature-extraction methods for conductive ferromagnetic material-thickness quantification," *Electronics*, vol. 8, no. 5, p. 470, 2019.

[12] J. V. Miro, D. Hunt, N. Ulapane, and M. Behrens, "Towards automatic robotic ndt dense mapping for pipeline integrity inspection," in *Field and Service Robotics*. Springer, 2018, pp. 319–333.

[13] J. Valls Miro, N. Ulapane, L. Shi, D. Hunt, and M. Behrens, "Robotic pipeline wall thickness evaluation for dense nondestructive testing inspection," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1293–1310, 2018.

[14] N. Ulapane, K. Thiyagarajan, D. Hunt, and J. Valls Miro, "Quantifying the relative thickness of conductive ferromagnetic materials using detector coil-based pulsed eddy current sensors," *J. Vis. Exp. (155), e59618*, 2020.