mCLESS: The Multi-Class Least-Error Square Sum for Interpretable Classification

1st Hwan Hee Park
Department of Mathematics and Statistics
Mississippi State University
Mississippi State, MS 39762 USA
hp352@msstate.edu

2nd Seung Heon Lee
Department of Biochemistry
Mississippi State University
Mississippi State, MS 39762 USA
sl1939@msstate.edu, *Contact Author*

3rd Hwamog Kim
Department of Sciences & Mathematics
Mississippi University for Women
Columbus, MS 39701, USA
hkim@muw.edu

4th Seongjai Kim
Department of Mathematics and Statistics
Mississippi State University
Mississippi State, MS 39762 USA
skim@math.msstate.edu

***Abstract*—Some machine learning algorithms are considered as black boxes, because the models are sufficiently complex and they are not straightforwardly interpretable to humans. Lack of interpretability in predictive models can undermine trust in those models in many application areas. The article introduces a new interpretable machine learning algorithm, called the *Multi-Class Least-Error Square Sum* (mCLESS). It is linear, simple to implement, and interpretable. Its nonlinear expansion is discussed. This simple algorithm turns out to be superior to many popular machine learning algorithms. Various experimental results involving synthetic datasets and UCI datasets are given to verify the claim.**

***Index Terms*—machine learning, interpretable model, explainable AI, feature expansion, UCI datasets**

I. Introduction

In the field of machine learning, classification is the most common task and applicable for various real-world problems such as text/speech recognition [1]–[3], face identification [4]–[6], and document classification [7]–[9]. It can be categorized as a supervised or unsupervised classification depending on the presence or absence of labels. Algorithms for the supervised classification differ from the ones for the unsupervised classification in the sense that they make use of the labeled data to construct proper classifiers; more classification tasks in practical applications are supervised.

The basic strategy behind supervised classification algorithms is to establish classifiers that map the given data into specific categories utilizing special forms of measure which indicate the similarity of data. There have been various measures that serve as mathematical/statistical foundations for successful classifiers. Linear classifiers are regarded as preferred classification methods due to their simplicity and scalability. Many common algorithms such as the logistic classification method [10] and the *support vector machine* (SVM) [11] are linear classifiers. As a non-parametric classifier, the *k-nearest neighbors* (kNN) [12] is formulated to classify given data based on a chosen distance measure. In addition, ensemble-type methods such as the deep neural network [13], [14], the random forest [15], [16], and kernel PCAs [17] make use of measures from existing methods to enhance the performance of classification.

In this article, we suggest a new interpretable machine learning algorithm, called the *Multi-Class Least-Error Square Sum* (mCLESS), which makes use of the least-squares formulation, most popularly used in regression analysis. The basic mCLESS is linear, simple to implement, interpretable, and applicable for datasets of arbitrary numbers of classes. In order to enhance the performance of the mCLESS for linearly inseparable datasets, a feature expansion scheme is discussed as its nonlinearization. This simple algorithm turns out to be superior to many popular machine learning algorithms.

The outline of this paper is as follows. In the following section, we briefly review binary classifiers. Section III describes the proposed classifier; its nonlinear expansion is discussed in Section IV. In Section V, we present numerical experiments performed with various datasets, with comparisons between the proposed classifier and popular classifiers built in SKlearn [18]. Section VI summarizes our experiments.

II. Preliminaries

This section presents a brief review on binary classifiers.

The *Perceptron* [19] (or Adaline) is the simplest artificial neuron that makes decisions for datasets of two classes by *weighting up evidence*.

- Inputs: feature values $\mathbf{x} = [x_1, x_2, \cdots, x_d]$
- Weight vector and bias: $\mathbf{w} = [w_1,\, w_2,\, \cdots,\, w_d]^T$, w_0
- Net input:
$$z = w_0 + w_1\, x_1 + w_2\, x_2 + \cdots + w_d\, x_d \qquad (1)$$
- *Activation and classification*: for threshold θ,
$$\phi(z) = \begin{cases} 1, & \text{if } z \geq \theta \\ 0, & \text{otherwise,} \end{cases} \qquad (2)$$
where ϕ is an activation function.

979-8-3503-2759-5/23/$31.00 ©2023 IEEE
DOI 10.1109/CSCE60160.2023.00016

When the logistic sigmoid function is chosen for the activation function, i.e., $\phi(z) = 1/(1+e^{-z})$, with $\theta = 1/2$, the resulting classifier is called the *Logistic Regression*. In general, the activation function is incorporated in order (a) to keep the net input restricted to a certain limit as per our requirement and, more importantly, (b) to add nonlinearity to the network.

Note that the net input in (1) represents a hyperplane in $\mathbb{R}^d$. More complex neural networks can be built, stacking the simple artificial neurons as building blocks.

Machine learning is to train weights from datasets of an arbitrary number of classes. The weights must be trained in such a way that

> *data points in a class are heavily weighted by the corresponding part of weights.*

This observation will be utilized when we formulate our classification method in the following section.

III. The mCLESS Classifier

In this section we introduce a new classifier, which is based on a least-squares formulation and able to classify datasets having arbitrary numbers of classes.

A. The mCLESS as a simple neural network

In order to describe the proposed algorithm effectively, we exemplify a synthetic data of three classes, as shown in Fig. 1, in which each class has 100 points.

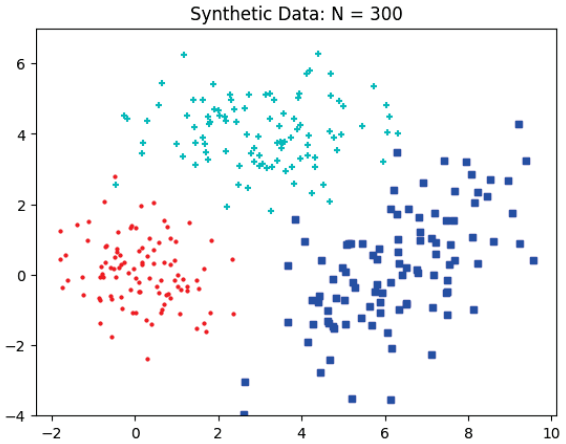

Fig. 1. A synthetic data of three classes.

A point in the c-th class is expressed as

$$\mathbf{x}^{(c)} = [x_1^{(c)}, x_2^{(c)}] = [x_1, x_2, c] \quad c \in \{0, 1, 2\},$$

where the number in () in the superscript denotes the class that the point belongs to. Let's consider an artificial neural network of the identity activation and no hidden layer, for simplicity.

Then, a set of weights can be trained in such a way that *points in a class are heavily weighted by the corresponding part of weights*, i.e.,

$$w_0^{(j)} + w_1^{(j)} x_1^{(i)} + w_2^{(j)} x_2^{(i)} = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad (3)$$

where δ_{ij} is called the Kronecker delta and $w_0^{(j)}$ is a bias for the class j.

Thus, for neural networks which classify a dataset of C classes with points in $\mathbb{R}^d$, the weights to be trained must have dimensions $(d+1) \times C$. The weights can be evaluated by the least-squares method. We will call the algorithm the *Multi-Class Least Error Square Sum* (mCLESS).

B. The Training Step in the mCLESS

This subsection formulates the mCLESS and its training step algebraically.

- **Dataset**: We express the dataset in Fig. 1 by

$$X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \in \mathbb{R}^{N \times 2}, \quad \mathbf{y} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \in \mathbb{R}^N, \quad (4)$$

  where $c_i \in \{0, 1, 2\}$, the class number.
- **The algebraic system**: It follows from (3), considering the bias $\{w_0^{(j)}\}$, an algebraic system can be formulated along with the information matrix and the source matrix.
  - *The information matrix*:

$$A = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ & \vdots & \\ 1 & x_{N1} & x_{N2} \end{bmatrix} \in \mathbb{R}^{N \times 3}. \quad (5)$$

  - *The source matrix*:

$$B = [\delta_{c_i, j}] \in \mathbb{R}^{N \times 3}. \quad (6)$$

    For example, if the i-th point is in Class 0, then the i-th row of B is $[1, 0, 0]$.
- **Least-squares formulation**: Then the *multi-column least-squares* (MC-LS) problem is formulated as

$$\widehat{W} = \arg\min_W ||AW - B||^2, \quad (7)$$

  which can be solved by the *method of normal equations*:

$$(A^T A)\, \widehat{W} = A^T B, \quad A^T A \in \mathbb{R}^{3 \times 3}. \quad (8)$$

- **The output of training**: The weight matrix

$$\widehat{W} = [\mathbf{w}^{(0)}, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}] = \begin{bmatrix} w_0^{(0)} & w_0^{(1)} & w_0^{(2)} \\ w_1^{(0)} & w_1^{(1)} & w_1^{(2)} \\ w_2^{(0)} & w_2^{(1)} & w_2^{(2)} \end{bmatrix}, \quad (9)$$

  where the j-th column weights heavily points in the j-th class.

The normal matrix A^TA is occasionally singular, particularly for small datasets. In the case, the MC-LS problem can be solved using the *singular value decomposition* (SVD).

Remark 3.1. Implementation of the Normal Equations. *Let $X^{(i)}$ be the collection of data points in the i-th class, $i \in \{0,1,2\}$, and the information matrix A in (5) denoted by*

$$A = [\mathbf{a}_0\, \mathbf{a}_1\, \mathbf{a}_2], \tag{10}$$

where $\mathbf{a}_j$ is the j-th column of A with $\mathbf{a}_0 = \mathbf{1}$. Then the matrices A^TA and A^TB in (8) can be expressed as

$$\begin{aligned} A^TA &= [\mathbf{a}_i \cdot \mathbf{a}_j], \\ A^TB &= \begin{bmatrix} |X^{(0)}| & |X^{(1)}| & |X^{(2)}| \\ \mathcal{C}(X^{(0)})^T & \mathcal{C}(X^{(1)})^T & \mathcal{C}(X^{(2)})^T \end{bmatrix}, \end{aligned} \tag{11}$$

where $\mathcal{C}$ is the column sum and $|\cdot|$ denotes the of rows (points). Note that the dot product $\mathbf{a}_i \cdot \mathbf{a}_j$ is not between point vectors but between feature columns.

C. The Prediction Step in the mCLESS

The prediction step in the mCLESS is quite simple:

(a) Let $[x_1, x_2]$ be a new point.
(b) Compute

$$[1, x_1, x_2]\,\widehat{W} = [p_0, p_1, p_2], \quad \widehat{W} \in \mathbb{R}^{3\times 3}. \tag{12}$$

Ideally, if $[x_1, x_2]$ is in class i, then p_i must be near 1, while other p_j's would be near 0. Thus p_i is the largest.
(c) Decide the class c:

$$c = \arg\max([p_0, p_1, p_2]). \tag{13}$$

D. Interpretation of the mCLESS

As a preprocessing, the dataset X in Fig. 1 is scaled column-wisely so that the maximum value in each column is 1 in modulus. The whole algorithm (training-prediction) is written in Python and run 100 times, with randomly splitting the dataset into 70:30 parts respectively for training and prediction; which results in 97.26% and 0.00171 sec for the average accuracy and the average elapsed-time, on a laptop of an Intel Core i7-10750H CPU at 2.60GHz.

- For the dataset X in Fig. 1, the output of the training, $\widehat{W}$, represents three sets of parallel lines.
- Let $[w_0^{(j)}, w_1^{(j)}, w_2^{(j)}]^T$ be the j-th column of $\widehat{W}$. Define $L_j(x_1, x_2)$ as

$$L_j(x_1, x_2) = w_0^{(j)} + w_1^{(j)}x_1 + w_2^{(j)}x_2, \;\; j = 0, 1, 2. \tag{14}$$

Fig. 2 depicts $L_j(x_1, x_2) = k$, $k = 0, 1$, for $j = 0, 1, 2$, superposed to the dataset. The mCLESS is easily interpretable. It follows from (13) that the mCLESS can be viewed as an one-versus-rest (OVR) classifier.

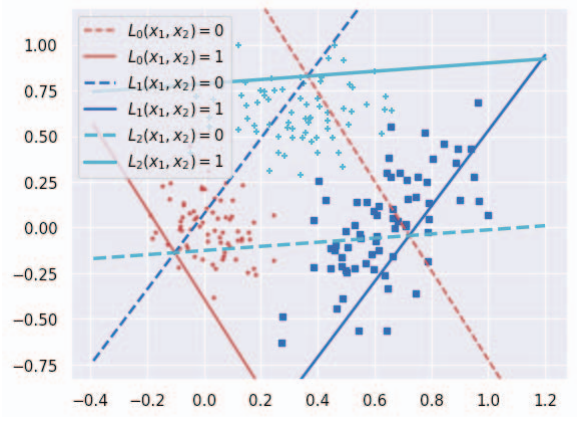

Fig. 2. Lines represented by the weight vectors.

IV. Nonlinearization: Feature Expansion

A. Feature Expansion

The mCLESS so far is a linear classifier. As for other classifiers, its nonlinear expansion begins with a data transformation, more precisely, *feature expansion*. For example, the *Support Vector Machine* (SVM) replaces the dot product of point vectors with the result of a kernel function applied to the point vectors:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) \approx \sigma(\mathbf{x}_i) \cdot \sigma(\mathbf{x}_j),$$

where σ is a function for feature expansion. Thus, without an explicit expansion of point vectors, the SVM can incorporate the effect of data transformation effectively. Such a technique is called the *kernel trick*.

However, the mCLESS does not incorporate dot products between point vectors; see Remark 3.1. Thus we must explicitly perform feature expansion without using a kernel trick, which results in an augmented normal matrix. Let's see some details.

A feature expansion is expressed as

$$\begin{cases} \mathbf{x} = [x_1, x_2, \cdots, x_d] \\ \mathbf{w} = [w_0, w_1, \cdots, w_d]^T \end{cases} \Rightarrow \begin{cases} \widetilde{\mathbf{x}} = [x_1, x_2, \cdots, x_d, \sigma(\mathbf{x})] \\ \widetilde{\mathbf{w}} = [w_0, w_1, \cdots, w_d, w_{d+1}]^T \end{cases} \tag{15}$$

where $\sigma(\mathbf{x})$ is a nonlinear feature function of $\mathbf{x}$, which will be specified later. Then, the expanded weights must be trained to satisfy

$$\begin{aligned} [1, \widetilde{\mathbf{x}}^{(i)}] \cdot \widetilde{\mathbf{w}}^{(j)} &= w_0^{(j)} + w_1^{(j)}x_1^{(i)} + \cdots \\ &= +w_d^{(j)}x_d^{(i)} + w_{d+1}^{(j)}\sigma(\mathbf{x}^{(i)}) = \delta_{ij}, \end{aligned} \tag{16}$$

for all points in the dataset. Equation (16) is an expansion of (3).

The corresponding expanded information and weight matrices read

$$\widetilde{A} = \left[\begin{array}{ccccc:c} 1 & x_{11} & x_{12} & \cdots & x_{1d} & \sigma(\mathbf{x}_1) \\ 1 & x_{21} & x_{22} & \cdots & x_{2d} & \sigma(\mathbf{x}_2) \\ & \vdots & & \ddots & & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Nd} & \sigma(\mathbf{x}_N) \end{array}\right],$$

$$\widetilde{W} = \left[\begin{array}{cccc} w_0^{(0)} & w_0^{(1)} & \cdots & w_0^{(C-1)} \\ w_1^{(0)} & w_1^{(1)} & \cdots & w_1^{(C-1)} \\ \vdots & & \ddots & \vdots \\ w_d^{(0)} & w_d^{(1)} & \cdots & w_d^{(C-1)} \\ \hdashline w_{d+1}^{(0)} & w_{d+1}^{(1)} & \cdots & w_{d+1}^{(C-1)} \end{array}\right], \quad (17)$$

where $\widetilde{A} \in \mathbb{R}^{N\times(d+2)}$, $\widetilde{W} \in \mathbb{R}^{(d+2)\times C}$, and C is the number of classes.

Feature expansion can be performed multiple times. When α features are added, the optimal weight matrix $\widetilde{W} \in \mathbb{R}^{(d+1+\alpha)\times C}$ is the least-squares solution of

$$\widetilde{W} = \arg\min_{W} ||\widetilde{A}W - B||^2, \quad (18)$$

which again can be solved by the method of normal equations:

$$(\widetilde{A}^T\widetilde{A})\,\widetilde{W} = \widetilde{A}^T B, \quad (19)$$

where B is the same as in (6) and

$$\widetilde{A}^T\widetilde{A} \in \mathbb{R}^{(d+1+\alpha)\times(d+1+\alpha)}.$$

Various feature functions $\sigma(\cdot)$ can be considered. Here, as a simple example of feature expansion, we will focus on the feature function of the form

$$\sigma(\mathbf{x}) = \|\mathbf{x} - \mathbf{p}\|, \quad (20)$$

the Euclidean distance between data points $\mathbf{x}$ and a prescribed point $\mathbf{p}$. Now, the question is: "*How can we determine the point* $\mathbf{p}$*?*"

B. Data Analysis for Determining $\mathbf{p}$

The goal is to find $\mathbf{p}$ such that the points transformed into the one-higher dimensions are more distinguishable class-by-class. For a simple analysis, we exemplify the `iris.data` from the UCI Machine Learning Repository. The dataset includes total 150 points in four dimensions, 50 in each of three classes. As a preprocessing, the dataset is scaled by column maximums.

In order to nonlinearize the mCLESS, we need to understand the algorithm and analyze the data as well.

Performance of the mCLESS: The mCLESS is run 100 times, with randomly splitting the `iris.data` into 70:30 parts respectively for training and prediction. Thus, in each run, each class sub-dataset is splitted to 35 and 15 points respectively for the training and the test. Fig. 3 depicts the cumulative class-wise prediction tables, one in counts and the other in accuracies. The average accuracy becomes 82.78%.

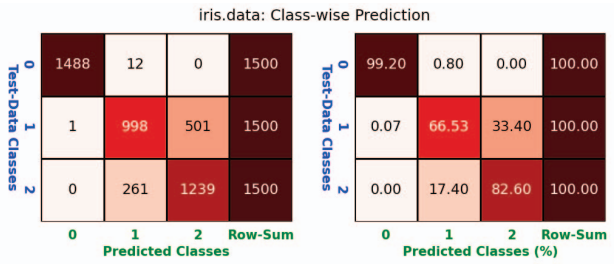

Fig. 3. mCLESS: The class-wise prediction for the `iris.data`.

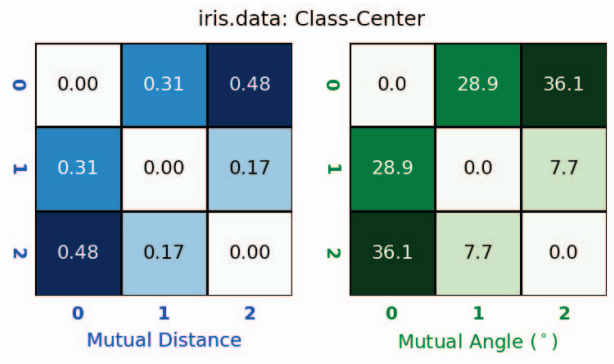

Fig. 4. Iris.data: Mutual distances and mutual angles between the class centers are presented.

As one can see from the figure, the major error occurs from the classification between `Class 1` and `Class 2`. This observation will be utilized when we try to determine the point $\mathbf{p}$.

Data analysis: As the first step for data analysis, we configure the centers of classes. The distances from the origin to the centers of classes show

$$[1.02933416\ \ 1.27413273\ \ 1.56731375]. \quad (21)$$

Fig. 4 shows mutual distances and mutual angles between the class centers of the dataset. From the centers and Fig. 4, we can see the following:

- The centers of three classes are not collinear but somewhat bent.
- Class 2 is the farthest one from the origin.
- Class 1 is nearer to Class 2 than to Class 0.

Determining $\mathbf{p}$: From the data analysis and observation, we select two candidates for $\mathbf{p}$:

$$\begin{aligned} \mathbf{p}_0 &= \texttt{CC}[1,:] - \eta_0(\texttt{CC}[2,:] - \texttt{CC}[0,:]), \\ \mathbf{p}_1 &= \texttt{CC}[1,:] - \eta_1(\texttt{CC}[2,:] - \texttt{CC}[1,:]), \end{aligned} \quad (22)$$

where `CC[k,:]` denotes the center of the training dataset in Class k, $k = 0, 1, 2$, and the scaling factors (η_0 and η_1) are to be set experimentally. The above points $\mathbf{p}_i$, $i = 0, 1$, are selected intuitively but in such a way that the feature column to be added will contain distinguishable values between Class

1 and Class 2. On the other hand, we must choose η_i appropriately not to be harmful for the highly-accurate classification of Class 0. By trial and error, we have found the following.

- The candidate $\mathbf{p}_0$ performs its best when $\eta_0 = 0.25$.
- The candidate $\mathbf{p}_1$ performs its best and better than $\mathbf{p}_0$ when $\eta_1 = 1.0$.

Thus we decide to add the following feature to all data points (training and test).

$$\sigma(\mathbf{x}) = ||\mathbf{x} - \mathbf{p}_1||, \tag{23}$$

where

$$\mathbf{p}_1 = \texttt{CC}[1,:] - (\texttt{CC}[2,:] - \texttt{CC}[1,:]) = 2 * \texttt{CC}[1,:] - \texttt{CC}[2,:].$$

Note that the class centers `CC` are obtained using the randomly-selected training data-subset only, in each run.

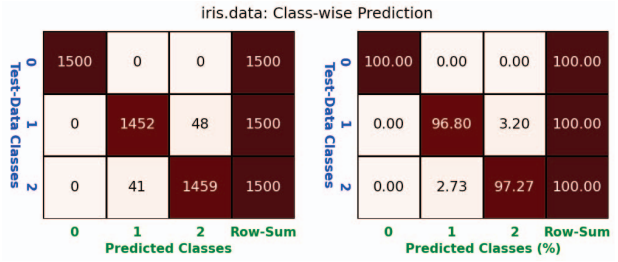

Fig. 5. mCLESS: The class-wise prediction for the `iris.data`, when a feature is added using (23).

Fig. 5 includes the cumulative class-wise prediction tables, when the mCLESS admits an extra feature column, as in (23), and runs 100 times. Compared with the results in Fig. 3, one can see a significant error reduction; the average accuracy becomes 98.02%. (For the basic mCLESS, the average accuracy was 82.78%.) Such a dramatic improvement in accuracy is due to the fact that the mCLESS is simple and interpretable and therefore conveniently reformed.

Here we summarize how to determine the point $\mathbf{p}$.

Summary 4.1. *The point* $\mathbf{p}$ *must be determined such that the extra column in the dataset has distinguishable values at least in two different classes. In order to find such an effective* $\mathbf{p}$*, one may perform the following. Note that in Python,* `CC[c] = CC[c,:]`*, the c-th row of* `CC`*. Define the center-to-center directional vectors.*

$$\texttt{DC}[\texttt{c}] = \texttt{CC}[\texttt{c}] - \texttt{CC}[(\texttt{c}+1)\%\texttt{nclass}]. \tag{24}$$

Then search the point $\mathbf{p}$ *of the form*

$$\mathbf{p} = \texttt{CC}[\texttt{i}] + \eta\,\texttt{DC}[\texttt{j}], \tag{25}$$

where η *is a parameter to determine. One can guess an appropriate pair* `(i, j)` *reasonably well, using the class-wise prediction scores* (Fig. 3) *and the mutual distance and mutual angle tables* (Fig. 4)*. After a few times of trials on* η*, one can finalize* $\mathbf{p}$*.* □

V. Numerical Experiments

The mCLESS classifier is compared with 10 well-known classifiers, built in scikit-learn or SKlearn [18], for the `synthetic.data` in Fig. 1 and various real datasets, available from the UCI Machine Learning Repository [20]; as shown in Tables I and II.

TABLE I
Classifiers, used for numerical experiments.

Classifier	Description
mCLESS	the linear mCLESS
mCLESS-E	the nonlinear expansion of the mCLESS with (20)
QDA	`QuadraticDiscriminantAnalysis()`
RF	`RandomForestClassifier()`
KNN	`KNeighborsClassifier()`
L-SVM	`SVC(kernel="linear")`
R-SVM	`SVC(gamma=2)`, the RBF SVM
DNN	`MLPClassifier()`
GNB	`GaussianNB()`, the Naïve Bayes
LR	`LogisticRegression()`
GP	`GaussianProcessClassifier()`
AB	`AdaBoostClassifier()`, Adaline Boost

TABLE II
Datasets, used for numerical experiments.

Dataset	# points	# classes	Source
synthetic.data	300	3	Synthetic
wine.data	178	3	UCI
iris.data	150	3	UCI
seeds_dataset.txt	210	3	UCI
australian.data	690	2	UCI
cmc.data	1473	3	UCI
digits.data	1797	10	UCI
banknote.data	1372	2	UCI

Each classifier is run 100 times for each dataset, randomly splitting the dataset into 70:30 parts respectively for training and prediction.

Table III shows average accuracy comparisons between the mCLESS and the best classifiers from SKlearn for datasets in Table II. As one can see from the table, the suggested classifier, mCLESS-E, has won the highest accuracy for five datasets (out of eight). It should be noticed that for `seeds_dataset.txt`, the simple linear classifier, mCLESS, performs better than any classifiers built in SKlearn.

TABLE III
Average accuracy comparisons: mCLESS and mCLESS-E vs. SKlearn Best Classifiers.

Dataset	mCLESS	mCLESS-E	SKlearn Best-Accuracy
synthetic.data	97.26	**97.67**	KNN – 97.28
wine.data	98.61	**99.24**	QDA – 98.78
iris.data	82.78	**98.02**	QDA – 97.82
seeds_dataset.txt	96.37	**96.43**	QDA – 93.95
australian.data	86.24	**86.46**	LR – 86.45
cmc.data	50.92	53.43	**AB – 54.22**
digits.data	93.36	94.17	**KNN – 97.96**
banknote.data	97.52	98.86	**KNN – 99.80**

Discussion: We close the section with discussion.

- *Nonlinearization*. As an effective nonlinearization of the (linear) mCLESS, one may consider various measures rather than the Euclidean distance from a specific point $\mathbf{p}$. Also, one may try to expand more than one features, using combinations of selected measures. A set of data analysis must precede the introduction of measures.
- *Outliers/Noise*. The least-squares formulation is sensitive to (and may be biased by) outliers or noise. It would better incorporate an effective noise removal algorithm.

We will address these issues in a forthcoming article.

VI. Conclusions

We have introduced a machine learning classifier called the *Multi-Class Least-Error Square Sum* (mCLESS) which is simple to understand, easy to implement, and therefore interpretable. The mCLESS is a linear classifier based on least-squares formulation. Its nonlinearization is suggested through a feature expansion, where an extra feature is chosen as the Euclidean distance measured from a specific point $\mathbf{p}$ to each data point. The point $\mathbf{p}$ is data-dependent; an effective strategy is discussed for the selection of the point. Compared with many classifiers built in SKlearn, the nonlinearization of the mCLESS, mCLESS-E, has won the highest accuracy for many datasets available from the UCI Machine Learning Repository.

References

[1] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012, pp. 3304–3308.

[2] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.

[3] M. El Ayadi, M. S. Kamel, and F. Karray, "Survey on speech emotion recognition: Features, classification schemes, and databases," *Pattern Recognition*, vol. 44, no. 3, pp. 572–587, 2011.

[4] M. Guillaumin, J. Verbeek, and C. Schmid, "Is that you? metric learning approaches for face identification," in *2009 IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 498–505.

[5] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, and D. P. Hughes, "Deep learning for image-based cassava disease detection," *Frontiers in Plant Science*, vol. 8, 2017. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpls.2017.01852

[6] K. Teoh, R. Ismail, S. Naziri, R. Hussin, M. Isa, and M. Basir, "Face recognition and identification using deep learning approach," *Journal of Physics: Conference Series*, vol. 1755, no. 1, p. 012006, feb 2021. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1755/1/012006

[7] J. M. Nicholson, M. Mordaunt, P. Lopez, A. Uppala, D. Rosati, N. P. Rodrigues, P. Grabitz, and S. C. Rife, "scite: A smart citation index that displays the context of citations and classifies their intent using deep learning," *Quantitative Science Studies*, vol. 2, no. 3, pp. 882–898, 11 2021. [Online]. Available: https://doi.org/10.1162/qss_a_00146

[8] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.

[9] L. M. Manevitz and M. Yousef, "One-class SVMs for document classification," *Journal of machine Learning research*, vol. 2, no. Dec, pp. 139–154, 2001.

[10] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[11] B. Schölkopf, A. J. Smola, F. Bach *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[12] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," *International Statistical Review/Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989.

[13] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Oct. 2012, pp. 3642–3649, generated from Scopus record by KAUST IRTS on 2022-09-14.

[14] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[15] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[16] N. Sari, Rokhmatuloh, and M. Manessa, "Monitoring dynamics of vegetation cover with the integration of obia and random forest classifier using sentinel-2 multitemporal satellite imagery," *Geoplanning*, vol. 8, no. 2, pp. 75–84, 2021.

[17] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[18] SKlearn. scikit-learn: Machine Learning in Python. [Online]. Available: https://scikit-learn.org/(Dateaccessed:21.03.2023)

[19] F. Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automaton Project Para*, ser. Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957. [Online]. Available: https://books.google.com/books?id=P_XGPgAACAAJ

[20] D. Dua and C. Graff. (2017) UCI machine learning repository. [Online]. Available: http://archive.ics.uci.edu/ml(Dateaccessed:16.03.2023)