A New Number Recognition Approach Using Typical Testors, Genetic Algorthims and Neural Networks

Eddy Torres, Julio Ibarra

Colegio de Ciencias e Ingenierías "El Politécnico",
Universidad San Francisco de Quito USFQ, Campus Cumbayá, Casilla Postal 17-1200-841, Quito, Ecuador
etorresc@estud.usfq.edu.ec, jibarra@usfq.edu.ec

***Abstract*—Testors, and particularly typical testors, have been used in feature selection and supervised classification problems. An objective on choosing what features select is to preserve accuracy. This also allows a model to keep its capacity to discriminate between classes. Consequently, we propose the Discriminator Construction Algorithm (DCA) as an strategy to achieve a subset of features that preserves accuracy. Some algorithms present in the literature focus only on finding the whole set of typical testors or to build minimum length ones which do not consider accuracy on prediction over classifiers. In this paper, we propose an algorithm that uses typical testors concepts and accuracy on prediction over neural networks as fitness functions for evolutionary strategy to build a subset of features that preserves the accuracy; in the same range as if we use the whole set of features to train and test the model. The paper presents some experimental results obtained by using a discriminator constructed by DCA to discuss assessment metrics of the model with the reduced set of features. We also contrast with other reported classifiers error rate over the same data set with the error rate of the new discriminator.**

***Keywords*—Handwritten number classification, typical testors, multilayer neural network, genetic algorithms, accuracy, evolutionary strategy, fitness function.**

I. Introduction

The concept of testor has been adapted to different applications over the years, one of them has been using testors to determine how much relevant information a set of features stores to describe a group of objects. This information is helpful on the classification and recognition of such objects [1]. Being able to determine which features are relevant and which are redundant is an important task in the field of supervised classification, as well as in patern recognition problems [2]. This is known as feature selection and it is a process of selecting a subset of features to reduce future search, training over data for classification or prediction spaces that go according to certain evaluation criteria for each of their fields [1].

The objective of reducing the dimensionality of the feature space is to find a minimum set of them that preserves the essential information and allows to distinguish and identify the compared classes, facilitating the training of models for areas of pattern recognition or data mining [3], [4], [5], [6].

A testor has the same ability to differentiate between objects that belong to different classes as the entire set of features does. This is why it has been used in supervised pattern recognition as in [7], [8], [9], [10]. The subspace where to find the testors and the cardinality of the set of all testors is enormous since they are searched among all possible subsets of the columns of the matrix where every column is a feature [1], [11], [12].

Some state-of-the-art algorithms include: LEX [13], YYC [14], all-NRD [15], CUDA based hill-climbing [16], Fast-BR [17], FAST-CT-EXT [18] which focus on returning the whole set of typical testors; it could be possible to compare the accuracy on prediction over all testors on these algortihms if it guarantees to find the whole set of typical testors, nevertheless in most real cases the exponential complexity does not allow it. It is also reported an algorithm to find minimum length typical testors [19]. However, heuristic algorithms as UMDA [20], PHC [21], HC [22] have a better performance over large datasets returning a subset of the of typical testors. The output of all types of algorithms focus on trying to reach the higher cardinality of the returned subset of typical testor or minimun-length subset of them. Therefore, we propose focusing on how effective can we build a testor so that we can use its properties of classification properly, since there are problems where the entire set or either a big subset is required [22]. Like in [23] where testors are used to improve the diagnosis of breast cancer cells. Furthermore, there are several problems in which there is a need to find an optimal discriminator (or close to it), that allows maximizing the performance of the classification by reducing the number of features used, is desired; specially in the fields of diagnostic diseases as in previous example and in [24], feature selection for text classification [7] and categorization [8]. Unlike other techniques developed for feature selection, testors have focused on this purpose, especially a certain type of them known as irreducible or typical [25], [26], [27], [28], [29].

As the time complexity increases as the number of features grow. As while finding the testors previous algorithms do not give metrics over how useful will be one testor over another or even the whole set when using it for discrimination. Furthermore, considering how a testor can be used in practice, is why we propose to build one. To achieve it, we propose starting from evolutionary strategy to reach typicality as near as possible. The objective function is also defined in terms of accuracy on testing, over a trained only with selected features multi-layer feed-forward back-propagation neural network, so that each result can be compared in terms of its efficiency in a reasonable computer time under given conditions of structure of the testor like the number of features desired in it. In this sense prediction accuracy plays an important roll over the evolving decision selecting not only the typical features but also the ones that increase the prediction accuracy. In fact, [20] considers that each typical testor can be consider as a local optimum for discrimination.

In this paper we propose by using previous evolutionary techniques and neural networks an approach to build an accuracy enough over prediction typical testor. As first we formally introduce the concept of an typical testor for a boolean matrix, we describe the multi-layer feed-forward neural networks, and also genetic algorithm strategy previously used to achieve an equal or closed enough typical testor with UMDA fitness function [20] as theoretical background; we detail our proposed method. Then we present results and analysis of the study. Finally, some conclusions and future work is presented.

II. Materials and Methods

A. Typical Testor

Let U be a collection of objects, these objects are described by a set of n features and are grouped into l classes. By comparing feature to feature each pair of objects belonging to different classes, we obtain a matrix $M = [m_{ij}]_{pxn}$ where $m_{ij} \in \{0,1\}$. $m_{ij} = 0$ and $m_{ij} = 1$ means that the objects of pair denoted by i are similar or different respectively, in the feature j. Let $I = \{i_1, ..., i_p\}$ be the set of the rows of M and Let $J = \{j_1, ..., j_n\}$ the set of labels of its columns.

Let a and b two rows of M.

Definition 1 [30]: We say that $a < b$ if $\forall\ ia_i \leqslant b_i$, and $\exists j$ such that $a_j \neq b_j$.

Definition 2 [30]: a is a basic row of M if there is no other row less than a in M.

Definition 3 [30]: The *basic matrix* of M is the matrix B only containing all different basic rows of M.

Let $T \subseteq J$, B^T be a subset of features obtained from B eliminating all columns not belonging to the set T.

Definition 4 [16]: Let p be a row of B^T; we say that p is a zero row if it contains only zeros.

Definition 5 [16]: A set $T = \{j_{k_s}, ..., j_{k_s}\} \subseteq J$ is a testor of M if no zero row in B^T exists.

Definition 6 [16]: The feature $x_i \in T$ is called a non-removable feature of T if there exists a row p in B^T such that if we eliminate from B^T the column corresponding to x_i, the remaining row p is a zero row of $B^{T-\{x_i\}}$. Otherwise, x_i is called a removable feature.

Definition 7 [16]: A set $T = \{j_{k_s}, ..., j_{k_s}\} \subseteq J$ is called typical with respect to the M matrix and of the collection U if it is a testor and each feature $x_i \in T$ is a non-removable feature of T.

Proposition 1 [11]: The set of all typical testors of M is equal to the set of all typical testors from the basic matrix B.

Let $\Psi^*(M)$ be the set of all typical testors of the matrix M. According to proposition 1, to search over the set $\Psi^*(M)$ it is very convenient to find the matrix B. Taking into account that B has equal or less number of rows than M, the efficiency of the algorithms should improve with B than with M [11].

B. Multi-layer feed-forward back-propagation Neural Networks (FFBPNN)

FFBPNN neural networks have been positioned as the most used type of neural networks [31]. They have been applied in several fields like prediction, image recognition, chemestry problems and more. A FFBPNN is built by neurons, which are ordered by layers. The first layer is the input layer and the last one is known as the ouput layer. All the layers in between are called hidden layers.

Let Γ be the mapping function that relates for each neuron i a subset $\Gamma(i) \subseteq V$ which consists of all ancestors of the given neuron.

The subset $\Gamma^{-1}(i) \subseteq V$ consists of all predecessors of the ith neuron.

Each neuron in a specific layer is connected with all the neurons of the next layer.

The connection between the ith and jth neuron is characterised by the weight coefficient w_{ij} and the ith neuron by the bias coefficient θ_i.

The weight coefficient measures the degree of significance of the given connection in the neural network.

The output value also called as activity of the ith neuron x_i holds that:

$$\begin{aligned} x_i &= h(\xi_i) \\ \xi_i &= \theta_i + \sum_{j \in \Gamma^{-1}(i)} w_{ij} x_j \end{aligned}$$

where ξ_i is the potential of the ith neuron and function $f(\xi_i)$ is the transfer function or activation function.

The supervised adaptation process varies the threshold coefficients θ_i and weight coefficients w_{ij} to minimise the objective function which relates computed and required output values.

The back-propagation algorithm, disperses the output error from the output layer through the hidden layers to the input layers so that the connection between the neurons can be recurrently calculated on training looking forward to minimize the loss function in each training iteration [32].

C. Genetic Algorithms (GA)

Genetic Algorithms are heuristic based search approaches, specially applicable to optimization problems. The main reason that make them useful in practice is their flexibility over a wide range of problems [33].

They begin with an initial population -set of initial points- and select a set of potential points to generate a new population. This operation is repeated until a stop criteria is reached. So it works by creating new populations of points (usually called chromosomes or individuals) by applying a set of genetic operators to the previous population of selected points [20].

Classical genetic operators are selection, crossover and mutation. Crossover recombines the genetic information contained in the parents of two individuals picked from the selection set, and mutation applies modifications to certain values (alleles) of variables (genes) in the points [34].

D. Database MNIST

The MNIST is a database of handwritten numbers. It is a widely used data set in machine learning. Handwriting recognition is a difficult problem and a good test for learning algorithms. The MNIST database has become a standard test. It collects 60,000 training images and 10,000 test images, taken from a previous database, simply called NIST1. These are black and white, normalized images centered at 28 pixels per side [35].

E. Proposed method

This section introduces an algorithm that searches for an optimal solution as a subset of features, in terms of efficiency over discrimination between classes of a collection U, based on a desired reduction of the total number of features.

Depending on the density of the elements of each class l of the collection U we choose up to n objects of each class, this is for practical calculation purposes, we call this sub-collection U'. The first step is to obtain the basic matrix B over the collection $U' \subseteq U$. If needed, previous image prepossessing is recommended to reduce noise.

The second step is to remove all the columns of B where all the rows are 0. We do this because this features do not help to discriminate. If a testor contains any of them. We can remove the feature or features since it will not generate a zero row by removing it. Furthermore, removing all this columns reduces the complexity on searching over the subspace of all possible combination of features.

Before introducing the Discriminator Construction Algorithm we present its components. For This purpose we will divide them in two.

Let P be a set of N randomly chosen subsets of features of B of size nxm such that $x_k = \{x_{k1}, ..., x_{km}\} \in P$ with $x_{ki} \in \{0,1\}$, $k = 1, ..., N$ and $i = 1, ..., m$. x_k is called a chromosome.

1) Genetic Components

As in [36] we use Univariate Marginal Distribution Algorthim (UMDA). The fitness function is defined as:

$$f(x_k) \quad = \quad \alpha \frac{t(x_k)}{n} + (1-\alpha)\frac{p(x_k)}{\sum_{v=1}^{m} x_{kv}} \qquad (1)$$

We define $T \subseteq x_k$ by eliminating all columns in x_k where $x_{ki} = 0$.

Where, $t(x_k)$ is the number of non-zero rows in B^T, $p(x_k)$ is the number of typical features of B^T and α is a weighting coefficient.

Recall that for any chormosome x_k, $0 \leqslant f(x_k) \leqslant 1$ where x_k is a typical testor if $f(x_k) = 1$

2) FFBPNN Components

The components for FFBPNN selected for modeling testor performance during evolution has one input layer, two hidden layers and one output layer. Let m' be the total number of test cases we explicit define accuracy of a chromosome as:

$$g(x_k) \quad = \quad \frac{\delta(x_k)}{m'} \qquad (2)$$

Where $\delta(x_k)$ is the number of correct predictions of the trained neural network over a common test set.

Recall that for any chromosome x_k, $0 \leqslant g(x_k) \leqslant 1$ where $f(x_k)$ approaching to 1 means a better performance on discriminating between the classes of U'.

Finally, we are able to describe DCA algorithm.

F. Experimental setup

For $U' \subseteq U$ we decided to choose 50 random objects of each class from training MNIST data set proceeding to binarize them to calculate B. With B calculated we removed all the zero columns in it, keeping record of the original indexes.

For the Genetic Components we settled α to 0.2 as we have higher probability of finding testors of large length [36]. We set the maximum number of iterations to 100 and the solution length to 1 as stop conditions. We searched only for one discriminator. Mutation was performed over 1% of the non-removed features in every generation.

For FFBPNN we set its topology as follows: the input layer and all the hidden layers has $relu(x) = max\{0, x\}$ as activation function. The first hidden layer has 52 neurons, and the second one has 26. The output layer has softmax as activation function which is defined as:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Where z is a vector of dimension k and $j = 1, ..., k$. In our case z has dimension 10 as we have that number of classes. Also we used Sparse Categorical Crossentropy loss function, set batch size equal to 1/5 of the training samples and used 10 epochs which definitions are detailed in [31], [37].

Since DCA requires a threshold to change fitness function from $f(x_k)$ to $g(x_k)$ we decide to set t as the maximum accuracy that the model can reach under a batch size of 1/5 of the training samples and the double epochs than used in $g(x_k)$ minus 0.04. Once a solution was found it was translated its original indexes.

For assessment metrics let us call *discriminator* to one solution found by DCA. Following the same topology of $g(x_k)$ and as the length of the selected features of *discriminator* turns

```
Data: Basic Matrix $B$
Result: Collection of subsets $T$ of the columns of $B$
1  $P \longleftarrow$ generate initial population;
2  $HP \longleftarrow$ empty list with the same size of $P$;
3  $S \longleftarrow$ solution list;
4  $t \longleftarrow$ set a threshold for typicality in chromosomes to
     start training neural networks;
5  $O(x_k) \longleftarrow$ set objective function to $f(x_k)$;
6  while not reached max number of iterations do
7      while not reached number of solutions do
8          for chromosome in $P$ do
9              $fv \longleftarrow$ Calculate $O(x_k)$ ;
10             if $fv > \alpha$ and $fv >$ all $fv$ in $HP$ then
11                 Append chromosome into $HP$ with it
                     fitness value;
12                 if $HP.size > P.size$ then
13                     Remove the element of $HP$ with
                         lowest $fv$;
14                 end
15             end
16             if $fv > minFitnessValue$ then
17                 Append chromosome into $S$;
18             end
19         end
20         $\bar{fv} \longleftarrow$ fitness value average;
21         if $\bar{fv} > t$ then
22             $O(x_k) \longleftarrow$ set objective function to $g(x_k)$;
23         end
24         $BP \longleftarrow$ Join top scored elements of P with all
             elements in HP;
25         $MF \longleftarrow$ calculate marginal frequency over $BP$;
26         $P \longleftarrow$ Generate a new population with $MF$
             distribution;
27         $NMF \longleftarrow$ calculate marginal frequency over
             new $P$;
28         if $NMF = MF$ then
29             Mutate $P$;
30         end
31     end
32 end
33 return $S$
```

Algorithm 1: Discriminator Construction Algorithm

the design of the neural network to be a unique model. This model was trained with Stratified K-Folds cross-validation for 5 folds over the train set each fold with 20 epochs whit the same batch size. For one-vs-all multi-class classification we also used Stratified K-Folds cross-validation for model training over the same topology but with only two folds over the train set [38]. We measured model's classification performance by accuracy, loss, multi-class precision. For calculate model precision we used three approaches: micro, macro and weighted. This types of precision contribute to present how samples and classes contribute to detailed view of model precision. We also measured multi-class log loss [39], one-vs-all ROC curves and AUC scores [40], and one vs all precision vs recall metrics [41]. Finally, we compared the *discriminator* error rate to some test error reported on the MNIST documentation.

Since the selection criteria was developed directly in evolution we choose the first solution that DCA report. All source code was implemented in *Python* language version 3.7.10 [42] with the scikit-learn (SKlearn) library [43] and Keras [44].

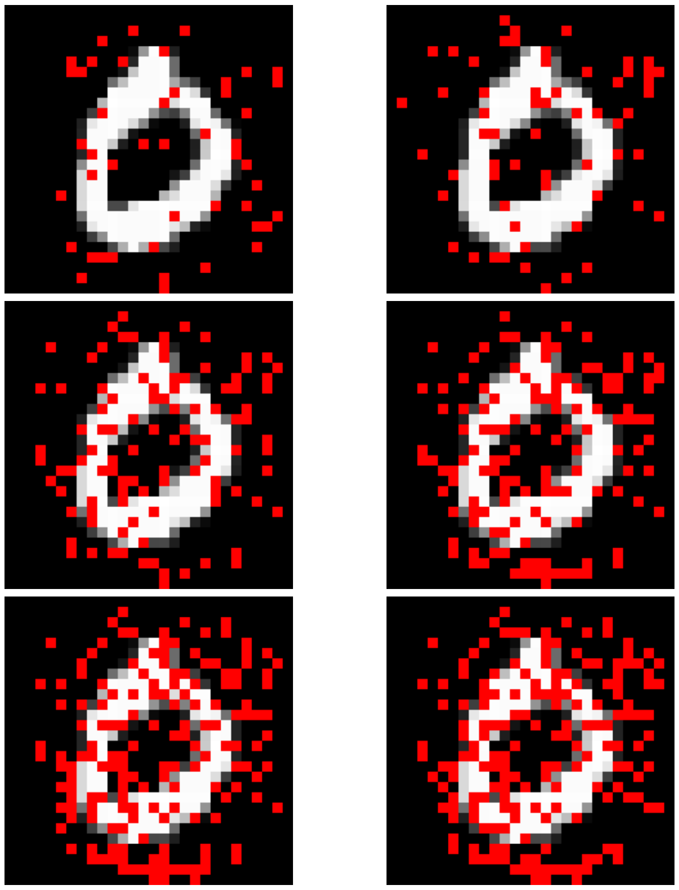
Fig. 1: Generations 0, 10, 20, 30, 40, 50 (With left right up down direction respectively)

III. Results and Discussion

A total of 50 generations were needed to build a discriminator with 21.81% of the total features. We present some evolution steps with the corresponding feature selection for each generation.

A. Performance evaluation

The *discriminator* reported from DCA has a length of 171 features which represents a 21.81% of the total amount of features. By reducing data sets matrices to those only containing the selected features and training a FFBPNN as described before we end up with the an accuracy on training of 99.65% and a validation accuracy of 97.83% on testing. In terms of loss the model reported a loss of 0.0191 on training and a loss of 0.081 on testing. Detailed view of this metrics versus epochs for each fold is presented below.

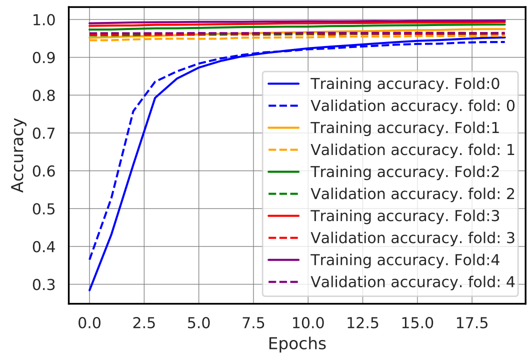

Fig. 2: Accuracy vs Epochs

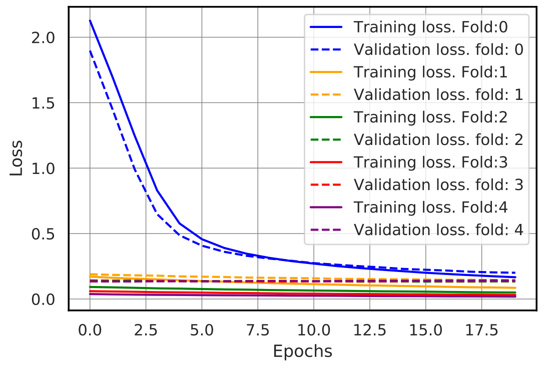

Fig. 3: Accuracy vs Epochs

The confusion matrix shows how the model performs on classifying between multiple classes. As we can see below almost the whole diagonal is almost highlighted. This describes a high rate between predicted labels and true labels. Results are presented in percentages.

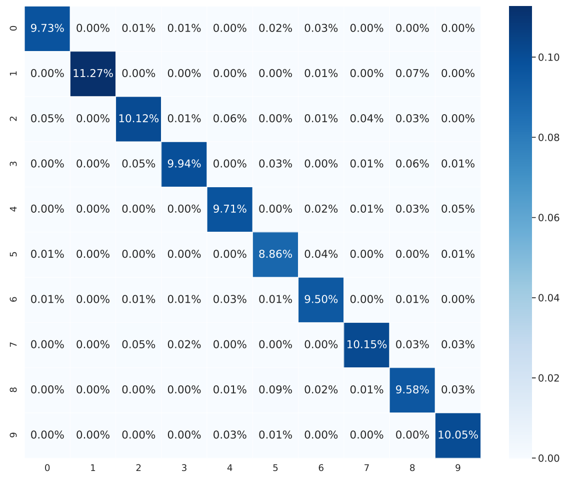

Fig. 4: Confusion Matrix for *discriminator* in percentage.

For micro-averaged precision we obtained a precision value of 97.15%. For macro-averaged a precision value of 97.14%. For weighed-average a precision value of 97.15%. Therefore the model is consistent and predicts accurate for distinct classes.

To show in more detail this fact we plotted the ROC curves for all one-vs-all with the corresponding AUC.

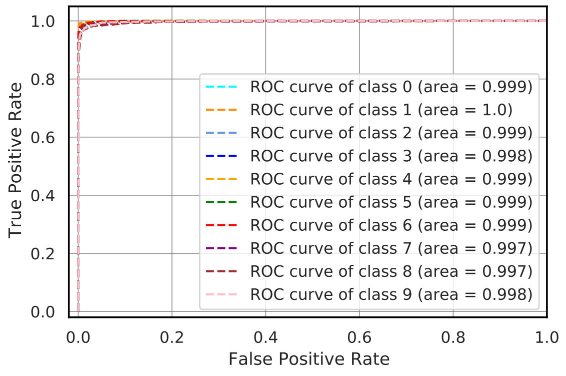

Fig. 5: ROC Curve one-vs-all

Furthermore, precision versus recall plot for one-vs-all shows another perspective of the reported precision values.

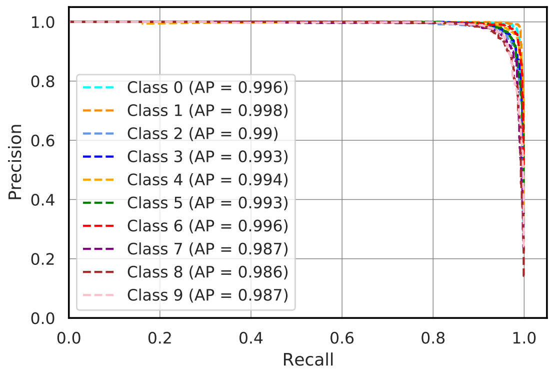

Fig. 6: Precision versus Recall one-vs-all

As we see the model can distinguish between all classes with precision.

Finally, we calculate multi-class log loss with a value of 0.2240 which means the model assigns a high probability for each class to predict correctly. For this value in particular the fact of having 10 classes makes the reported value a positive metric.

B. State of art based comparison

Despite the fact that there is no other discriminator in the literature, we can evaluate the performance of the model against the error rate reported in the MNIST documentation.

From table 1, it is possible to notice that the *discriminator* is the only model that used less features of the data set. Since for the state of the art-based comparisons we are going to focus on test error, it should be noted that the error rate is clearly in the range of most of them. Therefore, the proposed model based on DCA could be considered in less complex and faster in computational time, since it selects some features of the entire set.

Method	Features(%)	Error(%)
K-nearest-neighbors, Euclidean (L2)	100	5.00
Boosted stumps	100	7.70
40 PCA + quadratic classifier	100	3.30
SVM, Gaussian Kernel	100	1.40
3-layer NN, 300+100 hidden units	100	3.05
Convolutional net, cross-entropy	100	0.60
DCA	**21.81**	**2.18**

TABLE I: MNIST documentation comparison to DCA

IV. Conclusions and Future Work

This work proposed an algorithm (DCA) to build a discriminator for a group of classes. With the knowledge of testors we were able to establish a starting point for an intelligent search. With evolution strategy we searched for some features with properties of testors and typicality. Once this features was found, we changed the fitness value to reach the accuracy goal for this reduced featured set. In this point FFBPNN play with their accuracy on predicting played the role of fitness function. With this strategy we built a discriminator with 21.81% of the total amount of features. Taking in consideration all calculated assessment metrics and the interpretation of them, with the discriminator we can build a model that predicts with a precision over 97% and it can distinguish between all the classes. Therefore we conclude that DCA algorithm is able to build a reduced in features discriminator for training and testing over MNIST dataset.

As future work, we plan (1) to use typical testors properties to look for a minimun-length optimal classifier, (2) to explore other feature selection techniques to relate them to generate prediction models, as well as (3) to reduce complexity over calculations to search over bigger spaces.

Acknowledgment

Authors thank ....

References

[1] E. Alba-Cabrera, S. Godoy-Claderon, and J. Ibarra-Fiallo, “Generating synthetic test matrices as a benchmark for the computational behavior of typical testor-finding algorithms,” *Pattern Recognition Letters*, 2016.

[2] H. Liu and H. Motoda, *Computational methods of feature selection*. CRC Press, 2007.

[3] M. M. Mafarja and S. Mirjalili, “Hybrid binary ant lion optimizer with rough set and approximate entropy reducts for feature selection,” *Soft Computing*, vol. 23, no. 15, pp. 6249–6265, 2019.

[4] A. Saxena, K. Saxena, and J. Goyal, “Hybrid technique based on dbscan for selection of improved features for intrusion detection system,” in *Emerging Trends in Expert Applications and Security*. Springer, 2019, pp. 365–377.

[5] M. Wang, C. Wu, L. Wang, D. Xiang, and X. Huang, “A feature selection approach for hyperspectral image based on modified ant lion optimizer,” *Knowledge-Based Systems*, vol. 168, pp. 39–48, 2019.

[6] H. Zhou, Y. Zhang, Y. Zhang, and H. Liu, “Feature selection based on conditional mutual information: minimum conditional relevance and minimum conditional redundancy,” *Applied Intelligence*, vol. 49, no. 3, pp. 883–896, 2019.

[7] J. A. Carrasco-Ochoa and J. F. Martínez-Trinidad, “Feature selection for natural disaster texts classification using testors,” in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2004, pp. 424–429.

[8] A. Pons-Porrata, R. Gil-García, and R. Berlanga-Llavori, “Using typical testors for feature selection in text categorization,” pp. 643–652, 2007.

[9] S. Lopez-Perez, M. Lazo-Cortes, and H. Estrada-Garcia, “Medical electro-diagnostic using pattern recognition tools,” in *Proceedings of the iberoamerican workshop on pattern recognition (TIARP 97)*, 1997, pp. 237–244.

[10] V. Valev and J. I. Zhuravlev, “Integer-valued problems of transforming the training tables in k-valued code in pattern recognition problems,” *Pattern Recognition*, vol. 24, no. 4, pp. 283–288, 1991.

[11] E. Alba-Cabrera, M. Lazo-Coréz, and J. Ruiz-Shulcloper, “An overview of the concept of testor,” *Pattern Recognition Journal*, 2000.

[12] V. Valev and A. Asaithambi, “On computational complexity of non-reducible descriptors,” pp. 208–211, 2003.

[13] Y. S. Alganza and A. P. Porrata, “Lex: A new algorithm for calculating typical testors,” *Revista Ciencias Matematicas, Cuba*, vol. 21, no. 1, pp. 85–95, 2003.

[14] E. Alba-Cabrera, J. Ibarra-Fiallo, S. Godoy-Calderon, and F. Cervantes-Alonso, “Yyc: A fast performance incremental algorithm for finding typical testors,” in *Iberoamerican Congress on Pattern Recognition*. Springer, 2014, pp. 416–423.

[15] A. Asaithambi and V. Valev, “Construction of all non-reducible descriptors,” *Pattern Recognition*, vol. 37, no. 9, pp. 1817–1823, 2004.

[16] I. Piza-Davila, G. Sanchez-Diaz, M. S. Lazo-Cortes, and L. Rizo-Dominguez, “A cuda-based hill-climbing algorithm to find irreducible testors from a training matrix,” *Pattern Recognition Letters*, vol. 95, pp. 22–28, 2017.

[17] A. Lias-Rodriguez and G. Sanchez-Diaz, “An algorithm for computing typical testors based on elimination of gaps and reduction of columns,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 27, no. 08, p. 1350022, 2013.

[18] G. Sanchez-Diaz, I. Piza-Davila, M. Lazo-Cortes, M. Mora-Gonzalez, and J. Salinas-Luna, “A fast implementation of the ct_ext algorithm for the testor property identification,” pp. 92–103, 2010.

[19] I. Piza-Dávila, G. Sánchez-Díaz, M. S. Lazo-Cortés, and I. Villalón-Turrubiates, “An algorithm for computing minimum-length irreducible testors,” *IEEE Access*, vol. 8, pp. 56 312–56 320, 2020.

[20] G. Diaz-Sanchez, I. Piza-Davila, G. Sanchez-Diaz, M. Mora-Gonzalez, O. Reyes-Cardenas, A. Cardenas-Tristan, and C. Aguirre-Salado, “Typical testors generation based on an evolutionary algorithm,” pp. 58–65, 2011.

[21] I. Piza-Davila, G. Sanchez-Diaz, C. A. Aguirre-Salado, and M. S. Lazo-Cortes, “A parallel hill-climbing algorithm to generate a subset of irreducible testors,” *Applied Intelligence*, vol. 42, no. 4, pp. 622–641, 2015.

[22] G. Sanchez-Diaz, G. Diaz-Sanchez, M. Mora-Gonzalez, I. Piza-Davila, C. A. Aguirre-Salado, G. Huerta-Cuellar, O. Reyes-Cardenas, and A. Cardenas-Tristan, “An evolutionary algorithm with acceleration operator to generate a subset of typical testors,” *Pattern Recognition Letters*, vol. 41, pp. 34–42, 2014.

[23] A. Gallegos, D. Torres, F. Álvarez, and A. T. Soto, “Feature subset selection and typical testors applied to breast cancer cells.” *Res. Comput. Sci.*, vol. 121, pp. 151–163, 2016.

[24] M. Ortíz-Posadas, J. Martínez-Trinidad, and J. Ruiz-Shulcloper, “A new approach to differential diagnosis of diseases,” *International journal of bio-medical computing*, vol. 40, no. 3, pp. 179–185, 1996.

[25] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell, “Dimensionality reduction for visualizing single-cell data using umap,” *Nature biotechnology*, vol. 37, no. 1, pp. 38–44, 2019.

[26] D. Fernandez, C. Gonzalez, D. Mozos, and S. Lopez, “Fpga implementation of the principal component analysis algorithm for dimensionality reduction of hyperspectral images,” *Journal of Real-Time Image Processing*, vol. 16, no. 5, pp. 1395–1406, 2019.

[27] M. S. Lazo-Cortés, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and G. Sanchez-Diaz, “On the relation between rough set reducts and typical testors,” *Information Sciences*, vol. 294, pp. 152–163, 2015.

[28] G. I. Sayed, A. E. Hassanien, and A. T. Azar, “Feature selection via a novel chaotic crow search algorithm,” *Neural computing and applications*, vol. 31, no. 1, pp. 171–188, 2019.

[29] D. Singh and B. Singh, “Hybridization of feature selection and feature weighting for high dimensional data,” *Applied Intelligence*, vol. 49, no. 4, pp. 1580–1596, 2019.

[30] J. Ruiz-Shulcloper, A. Soto, and A. Fuentes, “A characterization of the typical testor concept in terms of a notable set of columns,” *Rev Cien Mat (in Spanish)*, vol. 1, no. 2, pp. 123–134, 1980.

[31] S. Haykin, *Neural Networks - A Comprehensive Foundation*, Macmillan, Ed., 2008.

[32] D. Svozil, V. Kvasnicka, and J. Pospichal, “Introduction to multi-layer feed-forward neural networks,” *Chemometrics and Intelligent Laboratory Systems*, vol. 39, no. 1, pp. 43–62, 1997.

[33] O. Kramer, *Genetic algorithms*. Springer, 2017.

[34] H. Muhlenbein, T. Manhnig, and A. Ochoa-Rodriguez, “Schemata, distributions and graphical models in evolutionary optimization,” *Journal of Heuristic*, vol. 5, no. 2, 1999.

[35] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[36] E. Alba-Cabrera, R. Santana, A. Ochoa-Rodriguez, and M. Lazo-Cortes, “Finding typical testors by using an evolutionary strategy,” in *Proceedings of the 5th Ibero American Symposium on Pattern Recognition*, 2000, p. 267.

[37] W. Liu, Y. Wen, Z. Yu, and M. Yang, “Large-margin softmax loss for convolutional neural networks.” vol. 2, no. 3, p. 7, 2016.

[38] C. A Ramezan, T. A Warner, and A. E Maxwell, “Evaluation of sampling and cross-validation tuning strategies for regional-scale machine learning classification,” *Remote Sensing*, vol. 11, no. 2, p. 185, 2019.

[39] E. Hazan and S. Kale, “Newtron: an efficient bandit algorithm for online multiclass prediction.” in *NIPS*, vol. 11. Citeseer, 2011, pp. 891–899.

[40] M. S. Wandishin and S. J. Mullen, “Multiclass roc analysis,” *Weather and Forecasting*, vol. 24, no. 2, pp. 530–547, 2009.

[41] D. M. Powers, “Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation,” *arXiv preprint arXiv:2010.16061*, 2020.

[42] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[44] F. Chollet *et al.*, “Keras,” https://github.com/fchollet/keras, 2015.