This full-text paper was peer-reviewed and accepted to be presented at the IEEE WiSPNET 2017 conference.

Constructing a Linear Discrete System in Kernel Space as a Supervised Classifier

C. Florintina[1] and E.S. Gopi[2]
Department of Electronics and Communication Engineering,
National Institute of Technology, Tiruchirappalli, India
Email: [1]florintinachaarlas@gmail.com [2]esgopi@nitt.edu

***Abstract*—The pattern recognition techniques involve feature extraction from the data, dimensionality reduction (like PCA, LDA, K-LDA, etc) and constructing a classifier (NN, NM, SVM, etc.) using the training set and validating the constructed classifier using the testing set. The usage of digital signal processing (DSP) techniques in pattern recognition is always limited to the feature extraction stage such as collecting the Fourier, wavelet co- efficients, HMM, GMM, etc. In this paper we explore the usage of classical DSP techniques like convolution, FIR filter to construct the classifier and compare it with the state of the art techniques. The proposed technique paves the alternative way to construct a classifier that is helpful for Big data analysis.**

***Index Terms*—Signal processing, Convolution, FIR filter, ANN, SVM, Big data analysis.**

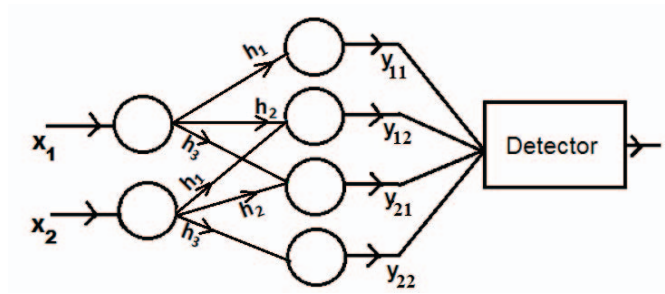

Fig. 1. Proposed Linear discrete model.

I. Introduction

With the ever growing technology and the on-going strive for automating almost everything, much attention has been given to Big Data Analysis. Web search engine companies like Google, social networking companies like Facebook and other such companies capture colossal amounts of data and analyzes them to discern who their clients are and to mould and improvise to present themselves as more proficient. Big Data Analysis is beginning to be of paramount importance in domains such agriculture, medicine, defense where analyzing huge amounts of data in the shortest period of time with the minimum storage space is vital.

[1] discusses in detail potential applications of pattern recognition in agriculture, especially pattern recognition from satellite imagery. The suitability of Back Propagation Neural Network (BPNN) for classification of remote sensing images is explored in [2]. [3] revolutionizes the technique of pattern recognition by Artificial Neural Network by concentrating on appropriately choosing their internal parameters, but these values vary for different scenarios and arriving at the appropriate parameters is time and effort consuming. [4] analyzes the performance of Multilayered Neural Networks (MLNs) and its training rule, Back propagation(BP) algorithm. The paper deliberates on some new methods to improve its training speed and introduce spacial information into pattern recognition of remote sensing images using a modified three-layered MLNs as classifier.

In the following section, we discuss how to construct a linear discrete system as a classifier. In Section III, we talk about how the input data is mapped into higher dimensional space and how the proposed LDS differs from convolutional neural network. The parameters that are to be chosen and how to deduce those parameters are presented in Section IV. In Section V, we provide the algorithm to be followed to employ the proposed LDS as a supervised classifier. The experiments done and the comparison with the state of the art techniques are presented in Section VI. Finally, we provide the perks and the limitations in using the proposed LDS as a classifier and the conclusion drawn in Section VII.

II. Linear Convolution Based Classifier

We construct the classifier as a Linear Discrete System (LDS) model. In this model the vector under test is given as input to the LDS model and the output sequence is given as input to the decision block for final classification. Let us consider that the input to the LDS is (number of attributes of the vector under test is 2) represented as $\mathbf{x} = [x_1\ x_2]^T$ and impulse response of the LDS is represented as $\mathbf{h} = [h_1\ h_2\ h_3]^T$ and the corresponding output is given as $\mathbf{y} = [y_{11}\ y_{12}\ y_{21}\ y_{22}\ ]^T$. They are related mathematically as $\mathbf{y} = \mathbf{x} * \mathbf{h}$ and can be illustrated as in Fig. 1. They are represented using the matrix as $\mathbf{y} = \mathbf{H} \times [x_1\ x_2\ 0\ 0]^T$, where $\mathbf{H}$ is the circulant matrix constructed using $\mathbf{h}$ and is represented as follows.

$$\mathbf{H} = \begin{bmatrix} h_1 & 0 & h_3 & h_2 \\ h_2 & h_1 & 0 & h_3 \\ h_3 & h_2 & h_1 & 0 \\ 0 & h_3 & h_2 & h_1 \end{bmatrix}.$$

We need to obtain the optimal value for $\mathbf{h}$ such that for the specific input vector $\mathbf{x_{ji}}$ (jth vector belonging to the ith class), we expect the corresponding output vector $\mathbf{y_{ji}}$ such that decision box is able to classify it as the ith class. The heuristic

method (based on Particle Swarm Optimization) to obtain the optimal solution for the vector $\mathbf{h}$ is as follows.

A. Methodology to Obtain the Optimal Solution

In the case of two class problem, we could choose target vector $\mathbf{t_1}$ corresponding to class 1 as $[1\ 1\ 0\ 0]^T$ and target vector $\mathbf{t_2}$ corresponding to class 2 as $[0\ 0\ 1\ 1]^T$. Thus the decision box may be constructed to decide in favor of class 1 if $y_{11}^2+y_{12}^2 > y_{21}^2+y_{22}^2$ or else class 2. In general for a multi-class problem we choose the class k as $k = arg_{i=1}^{i=r} min \sum_{j=i}^{j=\frac{N}{r}} y_{ij}^2$, where r is the number of classes, N is the number of elements of the vector $\mathbf{y}$ (chosen as multiples of r), y_{ij} are the elements of the vector $\mathbf{y}$. The optimal solution for $\mathbf{h}$ is obtained by minimizing the cost function as given in (1).

$$J(\mathbf{h}) = \sum_{i=1}^{i=r} \sum_{j=1}^{j=n_i} (\mathbf{y_{ji}} - \mathbf{t_i})^T (\mathbf{y_{ji}} - \mathbf{t_i}) \tag{1}$$

where, r is the number of classes, n_i is the number of vectors in the ith class and $\mathbf{y_{ji}}$ is the jth vector of the ith class. Particle Swarm Optimization (PSO) is the technique inspired from the biological behavior of the birds to choose the shortest path to reach the target destination. The graphical representation of PSO is as shown in Fig. 2. The solution to the objective function under consideration is treated as the particle (bird) position and it's euclidean distance from the target is treated as the value of the objective function. The value of the objective function reaches zero if the particle position is at the target position. Thus all the particles should move towards the target to minimize the objective function. This is done by following the algorithm specified in [5].

1) Initialize the particle positions as $\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$, $\cdots \mathbf{p_r}$ randomly that are uniformly distributed.
2) Initialize the tentative decisions taken by the individual particles for the next set of positions as $\mathbf{q_1}$, $\mathbf{q_2}$, $\mathbf{q_3}$, $\cdots \mathbf{q_r}$ randomly that are uniformly distributed.
3) Compute the functional values for the individual particles $\mathbf{q_1}$, $\mathbf{q_2}$, $\mathbf{q_3}$, $\cdots \mathbf{q_r}$ using (1). Let them be $J_1, J_2, \cdots J_r$.
4) Identify g as $g = arg_{t=1}^{t=r} min J_t$. The particle $\mathbf{q_g}$ is declared as the global.
5) Identify the actually moved next set of positions as follows: $\mathbf{pnext_i} = \mathbf{p_i} + \alpha(\mathbf{q_i} - \mathbf{p_i}) + \beta(\mathbf{q_g} - \mathbf{p_i})$ for $i = 1\ 2\ \cdots\ r$. The values of α and β are decided based on the weightage given to individual decision and global decision respectively. This completes one iteration.
6) Now the current position of all the particles are described by $\mathbf{p_i} = \mathbf{pnext_i} \forall i$. To move further, we need the tentative decisions for $q_i \forall i$. This is obtained by choosing $\mathbf{q_i} = \mathbf{pnext_i}$ if $J(\mathbf{pnext_i}) < J(\mathbf{q_i})$, else by choosing $\mathbf{q_i} = \mathbf{q_i}$.
7) Repeat steps (3) to (6) for finite number of iterations and the best particle position is identified to obtain the optimal solution for $\mathbf{h}$.

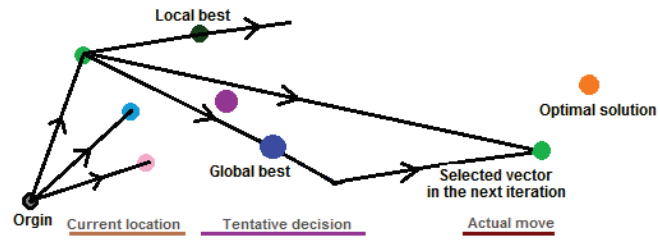

Fig. 2. Graphical representation of PSO algorithm.

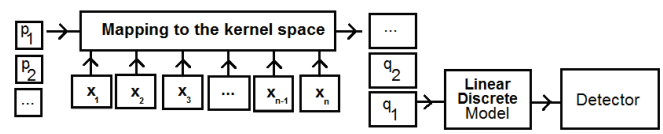

Fig. 3. Proposed classifier structure based on linear discrete model in the Kernel space.

III. Kernel Based Linear Convolution Based Classifier

Let $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_T}$ be the training set used to construct the classifier in the feature dimensional space (actual set), where $T = \sum_{i=1}^{i=r} n_i$ is the total number of vectors used in the training set. The process of mapping the vector to the Higher Dimensional Space (HDS) plays an important role in getting well separated classes in the HDS. In the case of Kernel-LDA, Columns of the Gram- matrix ($\mathbf{G}$) are treated as the newly constructed training set and are computed as follows.

$$\mathbf{G}(m, n) = f_{kernel}(\mathbf{x_m}, \mathbf{x_n}) \tag{2}$$

where, f_{kernel} is the Kernel function. The first n_1 columns of the matrix $\mathbf{G}$ belong to class 1, the next n_2 columns of the matrix $\mathbf{G}$ belong to class 2 and so on. With this, we have formulated the linear discrete model based classifier in the Kernel space (refer Fig. 3). Here, the input vector, $\mathbf{p_i}$ is mapped to the HDS using the Kernel function and is then fed to the LDS as $\mathbf{q_i}$. It is noted that in this technique, number of elements in the Kernel mapped vector is equal to the number of training set data, i.e., n.

A. How the Proposed LDS is Different from Convolutional Neural Network (CNN or ConvNet)

ConvNet is a derivative of Multi-layer Perceptrons. It was inspired by the receptive field structures located in the cat's primary visual cortex, which was first identified by Hubel and Wiesel [6]. Unlike ConvNet, LDS has no hidden layer at all. Though, both in CNN and LDS, full connectivity is not used between the layers, in ConvNet, this partial connectivity is to exploit spatially local correlation and in LDS, it is nothing but the simple operation of convolution between the impulse response of the LDS and the input. Moreover, ConvNets are specifically designed for images, whereas, the proposed LDS can be used as a classifier without any such constraint.

IV. Parameters Selection

The two main parameters that have to be decided before beginning the experimentation with each dataset and those which play a vital role in obtaining the maximum possible percentage of success are

- Sigma used in Gaussian Kernel
- Number of LDS impulse response co-efficients

A. Sigma Used in Gaussian Kernel

Before the data is given as input to LDS, Gaussian Kernel is applied to it using

$$\mathbf{GaussianKernel}(\mathbf{X_i}, \mathbf{X_j}) = \mathbf{exp}(\frac{-||\mathbf{X_i} - \mathbf{X_j}||^2}{\mathbf{sigma}}). \quad (3)$$

The value of sigma is optimized to yield the maximum percentage of success [7], with the aid of Particle Swarm Optimization, by

- Minimizing the value of excess kurtosis
- Maximizing the homoscedasticity
- Maximizing the euclidean distance

1) Minimizing the Value of Excess Kurtosis: The excess-kurtosis is computed as

$$\mathbf{E}(\mathbf{Z^4}) - \mathbf{3E}(\mathbf{Z^2})^\mathbf{2}. \quad (4)$$

Minimizing the value of excess-kurtosis results in the individual classes becoming Gaussian distributed.

2) Maximizing the Homoscedasticity: Maximizing the homoscedasticity results in the individual classes having identical co-variance matrices in the HDS. This is done by maximizing

$$\mathbf{Q} = \frac{\sum_{\mathbf{i=1}}^{\mathbf{r-1}} \sum_{\mathbf{j=i+1}}^{\mathbf{r}} \mathbf{Q^{ij}}}{\mathbf{r(r-1)}} \quad (5)$$

where $\mathbf{Q^{ij}} = \frac{\mathbf{tr}(\sum_{\mathbf{i}\phi} \sum_{\mathbf{j}\phi})}{\mathbf{tr}(\sum_{\mathbf{i}\phi}^{\mathbf{2}}) + \mathbf{tr}(\sum_{\mathbf{j}\phi}^{\mathbf{2}})}$
r is the number of classes
$\sum_{i\phi}$ is the co-variance of the ith class

3) Maximizing the Euclidean Distance: Maximizing the euclidean distance is done by maximizing the distance between the centroids of all classes.

The purpose of this attempt at making the individual classes Gaussian distributed with identical co-variance matrices and centroids as farther apart as possible is to minimize the chances of the classes overlapping with each other and thus making the classes effortlessly separable.

B. Number of LDS Impulse Response Co-efficients

The number of impulse response co-efficients of the LDS is gradually increased and each time the LDS is trained and the accuracy with which the training data are classified is observed (refer Fig. 4). The minimum number of co-efficients which gives the highest percentage of success is chosen and sometimes a compromise is made between the percentage of success and the number of co-efficients.

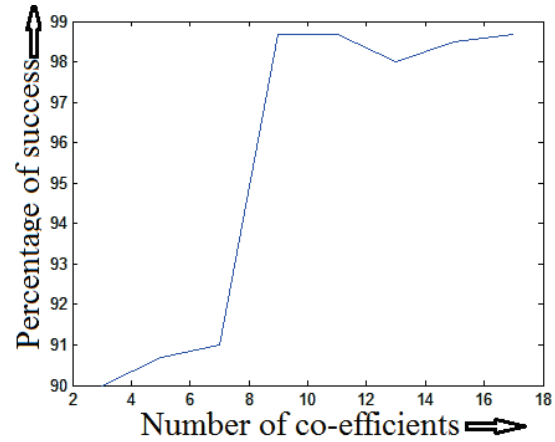

Fig. 4. Percentage of success for varying number of LDS impulse co-efficients

V. Algorithm

1) The data to be worked on is collected in the form of vectors. In the case of images, the image is taken block by block and is re-shaped into a vector.
2) The data is separated into two groups, for training and testing. Care must be taken that a considerable number of instances belonging to every class are available in the training set to ensure the proper training of the classifier.
3) The optimum value of sigma is obtained by using PSO such that the individual classes are Gaussian distributed with identical co-variance matrices and their centroids as far as possible to avoid overlapping of classes.
4) Once the value of sigma is obtained, the training data is mapped to the HDS by applying Gaussian Kernel.
5) The output that is expected of the LDS for each and every class is decided in advance. Care must be taken that the targets are so chosen such that they could be distinguished into their respective classes with ease by the decision maker. The targets must also be chosen such that it is within the bounds of being achievable.
6) Once the targets are decided, the Kernel mapped data is fed in the form of vectors, one by one to the LDS and the output is obtained by the convolution of the input with the impulse response co-efficients of the LDS. How different the obtained output is from the pre-determined target for that particular class is calculated in terms of means square error and this error is used to adjust the co-efficients of the impulse response of the LDS by employing PSO.
7) Step 6 is repeated for several iterations till the mean square error reduces and becomes stable as in Fig. 5.
8) Steps 6 and 7 are repeated for various number of impulse response co-efficients of the LDS and the minimum value which gives the maximum percentage of success for the training is chosen as the number of impulse response co-efficients of the LDS and the LDS is trained.
9) The testing data is then mapped into the HDS and is fed into the trained LDS, one by one and the output of the

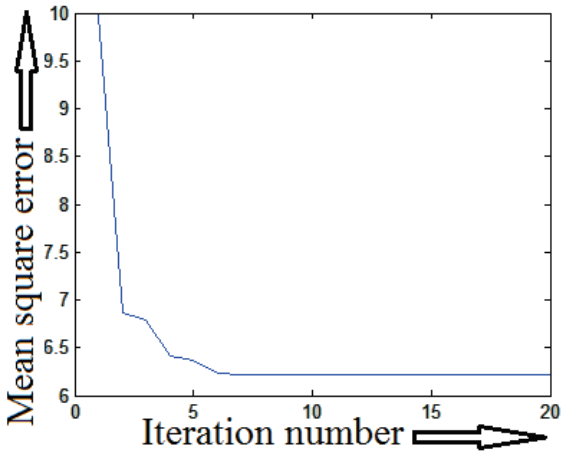

Fig. 5. Mean square error Vs iteration number

LDS is collected every time. The decision maker decides in favor of the class whose pre-determined target closely resembles the output obtained.

10) The percentage of success for the testing data is calculated to check the efficient functioning of the LDS as a supervised classifier in Kernel space.

VI. Experimental Results (Comparison with the State of the Art Techniques)

A. Experiments Done with Synthetic Datasets

The proposed LDS was initially put to test on synthetic dataset including 2D representation of 9 different clusters with different number of classes and with varying levels of complexity. Initially, when the raw data was directly used on the proposed LDS, it was observed that the percentage of success was very less and in most cases, the trained LDS did not even function as a classifier. When the data was mapped into HDS using Gaussian Kernel, though the process became more time consuming, the percentage of success showed a tremendous increase. Finally, when the LDS was designed as a linear phase FIR filter, the storage space required reduced to about half the original space required and the percentage of success was found to increase as the number of co-efficients to be deduced by PSO had also decreased.

B. Experiment Done with 2D NIR Images

Once it became evident that the proposed LDS is capable of functioning as an efficient supervised classifier, it was put to test with the aid of 2D NIR images obtained from [8]. The data was collected from corn cultivated agricultural lands of USA. Unmanned Aerial Vehicles (UAVs) were programmed to fly over hectares of land and capture NIR images in certain intervals and the images were stored along with co-ordinate details of each image. When manually monitoring vast agricultural fields becomes tedious, automating it eases man's efforts. In case of natural disasters such as floods and storms, assessing the total damage and taking actions to minimize the loss could be done swiftly and with more

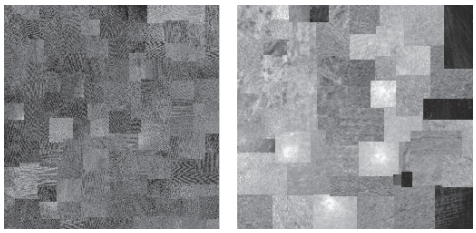

Fig. 6. Training data for corn cultivated region (left) and training data for barren or damaged land (right)

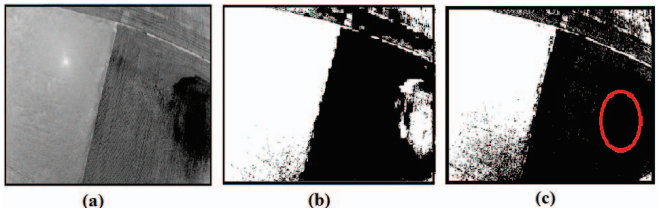

Fig. 7. (a) is the input image. (b) is the appropriately demarcated output with corn cultivated region depicted in black and the rest of the region in white, obtained with the proposed LDS as the classifier. (c) is the output obtained with ANN as the classifier. The red circle signifies the misclassified area.

ease with the aid of machine intelligence. This can be done by initially converting the images into gray images and by choosing randomly about 10 of the UAV captured images and arbitrarily collecting blocks of the desired region (corn cultivated region) as a single image, to be used as training data for the desired class and the process is repeated to obtain the undesired region (barren or damaged land) training data as shown in Fig. 6. The training data of the two classes thus acquired are to be mapped into HDS with the aid of Gaussian Kernel and given as input to the LDS (with 7 impulse response co-efficients) in the form of vectors to train it. Once the training is complete, the NIR images excluding those used for training are now tested. Each image is given block by block as input to the trained LDS after being mapped to HDS and the decision maker assigns zeros for the block classified as corn and ones for the block classified as undesired class. The output is used to demarcate the corn cultivated field from the barren or damaged land. When a randomly chosen image from the database was fed to the trained classifier, the output obtained is as shown in Fig. 7.

To compare the proposed LDS with the state of the art techniques, Artificial Neural Network with one hidden layer containing 7 nodes was employed as the classifier. Since the dimension of the input was high and too taxing on the ANN, Principal Component Analysis (PCA) was employed to reduce the dimensionality of the input. The output obtained (as in Fig. 7) was found to have misclassified some damaged land as corn cultivated region. It is interesting to notice that the proposed classifier which required the storage of only 4 co-efficients by exploiting the linear phase property performed better than the ANN which required 98 times more storage.

To emphasize the need for a classifier to do the task of identifying the corn cultivated region, the NIR image was

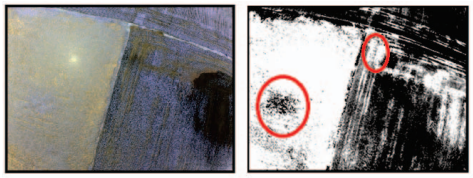

Fig. 8. Input image (left) and output with corn cultivated region depicted in black and the rest of the region in white, obtained by classification based on hue content. The red circles signify the misclassified regions

directly analyzed and the pixels which had their hue content within predetermined threshold values were classified as corn cultivated region as in Fig. 8. Even after exploiting the hue content of the image which the proposed LDS did not do, some of the barren land was misclassified as corn cultivated region since the texture details were not made use of.

C. Experiments Done with Real Datasets

The efficiency of the proposed LDS as a classifier was further verified with the aid of real datasets mentioned in Table I.

In the case of each dataset, about half of the instances of each class were grouped together into training data, were mapped to HDS and were used to train the LDS. The rest of the dataset was used to test the trained LDS after being mapped into the HDS. The results observed in the case of iris and image segmentation dataset are as shown in Fig. 9 and Fig. 10 respectively.

The efficiency of the proposed LDS is verified with the aid of recent work on the same datasets as in Table II. [11] makes an attempt to compare the performance of different machine learning algorithms by classifying 11 complex UCI datasets with the aid of Weka and finds VFI (Voting Feature Intervals) to have the highest classification accuracy in the case of iris dataset. [12] explicitly works on Iris plant dataset and analyzes in a meticulous way how efficiently the Multilayer Feed Forward Neural Network classifies the Iris plant for various number of training epochs. Most of the research on stacking selects by hand the right combination of classifiers and their parameters. Instead of starting from these initial strong assumptions, [13] uses genetic algorithms to search for good stacking configurations. To avoid over-fitting, the overall efficiency of the approach is evaluated empirically. But the stacking of classifiers obviously leads to a bigger requirement on storage. [14] investigates the impact of discretization on classification and establishes that lazy k-star on raw data, without discretization gives the maximum percentage of success. [15] proposes a novel approach for EEG eye state identification using Incremental Attribute Learning (IAL) based on neural networks. [16] compares the performance of k-Nearest Neighbors algorithm and Multilayer Perceptron Neural Networks models with the aid of EEG Eye dataset. The highest success was achieved in the classification made with kNN algorithm as 84.06%. [17] compares the performance of several constructive morphological models and of conventional Multi-layer Perceptrons.

VII. Conclusion

The very commonly used supervised classifiers are Support Vector Machine and Artificial Neural Network. In an ANN with two hidden layers having 'n' and 'p' number of nodes and if the dimension of the input is 'm', then 'm×n×p' weights and 'n+p' bias values have to be stored. If SVM is employed to classify data amidst 'n' classes and the dimension of data is 'm', then it becomes essential to store 'n-1' weight vectors, each of size 'm×1'. Thus a total of 'm×(n-1)' values have to be stored. With Big data analysis, the dimension of input data, 'm' increases tremendously and hence using ANN or SVM directly becomes very much time and storage consuming. To avoid that, dimensionality reduction techniques have to be used. Thus more computation is needed and this decreases the speed of operation.

After a series of experimentation on the proposed LDS with the aid of varied datasets, it could be concluded that it mostly performed as well as the other state of the art classifiers and sometimes even surpassed them. The added boon is that when the other supervised classifiers almost definitely require various pre-processing techniques, the proposed LDS has been observed to perform well in Kernel space without the aid of such pre-processing techniques. This makes the LDS less complex and more time efficient. The fact that the storage space required has never been for more than 7 co-efficients for the various experiments conducted, makes the LDS very storage efficient. All of this paves the way for the proposed classifier to be adept for big data analysis, especially when the data to be worked with is continuously streaming like EEG, since, the speed of processing is of utmost importance to avoid any lag in actions being taken and a storage requirement as minimum as possible becomes vital.

For the proposed LDS to function efficiently as a supervised classifier in Kernel space, choosing the apt value for sigma used in Gaussian Kernel, the number of impulse response co-efficients for the LDS and the appropriate target vector for each class is of utmost importance. Arriving at the most suitable combination of the above mentioned parameters is quite taxing, but the other promising features of the proposed LDS makes it worth the efforts.

References

[1] M.P. Raj, P.R. Swaminarayan, J.R. Saini and D.K. Parmar, "Applications of Pattern Recognition Algorithms in Agriculture: A Review," International Journal of Advanced Networking and Applications, vol. 6, issue. 5, pp. 2495-2502, March 2015.

[2] Jiefeng Jiang, Jing Zhang, Gege Yang, Dapeng Zhang and Lianjun Zhang, "Application of Back Propagation Neural Network in the Classification of High Resolution Remote Sensing Image," 18th International Conference on Geoinformatics, pp. 1-6, 2010.

[3] L. Zhou and X. Yang, "Use of Neural Networks for Land Cover Classification from Remotely Sensed Imagery," International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 37, part B7, Beijing 2008.

TABLE I
DETAILS OF THE DATASET USED.

Name of the dataset	Iris	Wine	EEG Eye	RCV1	Image segmentation
Source	UCI Machine Learning Repository [9]	UCI Machine Learning Repository [9]	UCI Machine Learning Repository [9]	Reuters, Ltd. [10]	UCI Machine Learning Repository [9]
Number of classes	3	3	2	4	7
Dimension of the input/ Number of attributes	4	13	14	29,992	19
Number of instances	150	178	14,980	9,625	2,310
Number of instances used for training	75	88	2,000	4,812	210
Number of instances used for testing	75	90	3,745	4,813	2,100

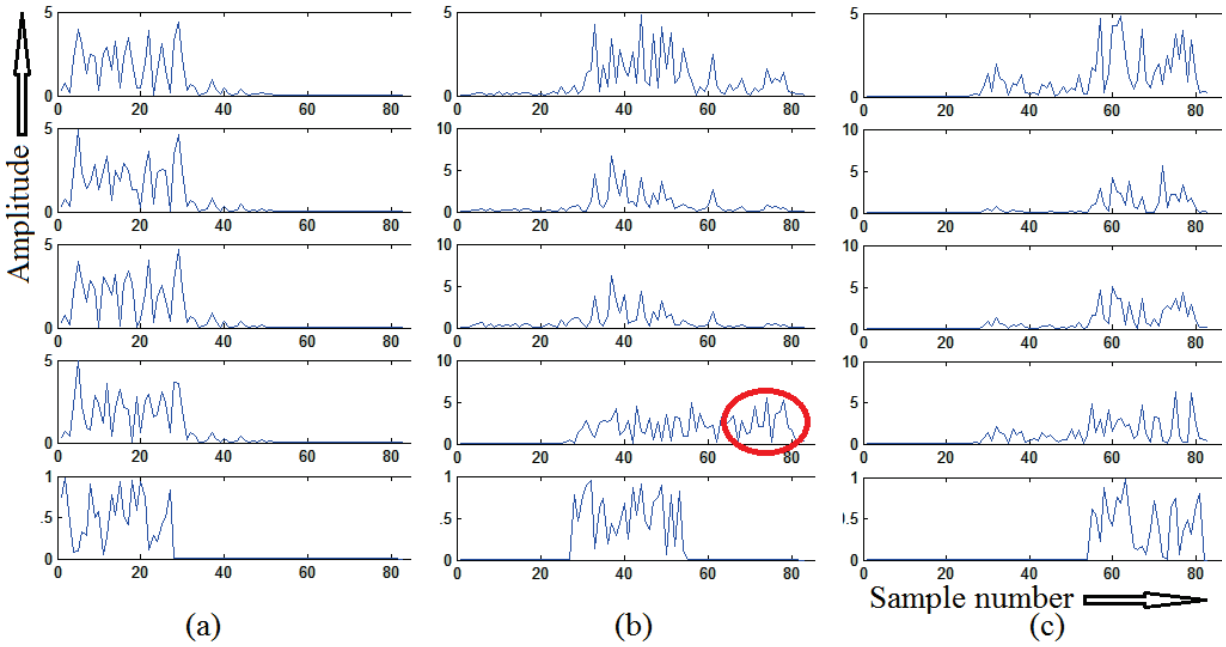

Fig. 9. Iris plant dataset: The first four rows of (a), (b) and (c) are output of the LDS for class 1, 2 and 3 data respectively. The fifth row of (a), (b) and (c) are the intended ones or targets predetermined for class 1, 2 and 3 respectively. The red circle signifies the misclassified data and the mismatched region compared with the target.

TABLE II
VALIDATING THE PERFORMANCE OF THE PROPOSED LDS WITH THE AID OF RECENT WORK.

Name of the dataset	Proposed LDS			Previous work/ Existing classifers
	Number of coefficients for LDS	Number of coefficients to be stored	Percentage of success achieved	Maximum percentage of success achieved
Iris	9	5	**98.67%**	96.67% in [11] (2015) 96.66% in [12] (2012) 95.29 in [13] (2004)
Wine	11	6	89.77%	**96.85%** in [13] (2004)
EEG Eye	9	5	**95.67%**	95.03% in [14] (2016) 72.60% in [15] (2014) 84.06% in [16] (2015)
RCV1	7	4	**94.10%**	91.20% (using PCA followed by SVM)
Image segmentation	13	7	85.62%	**90%** in [17] (2009)

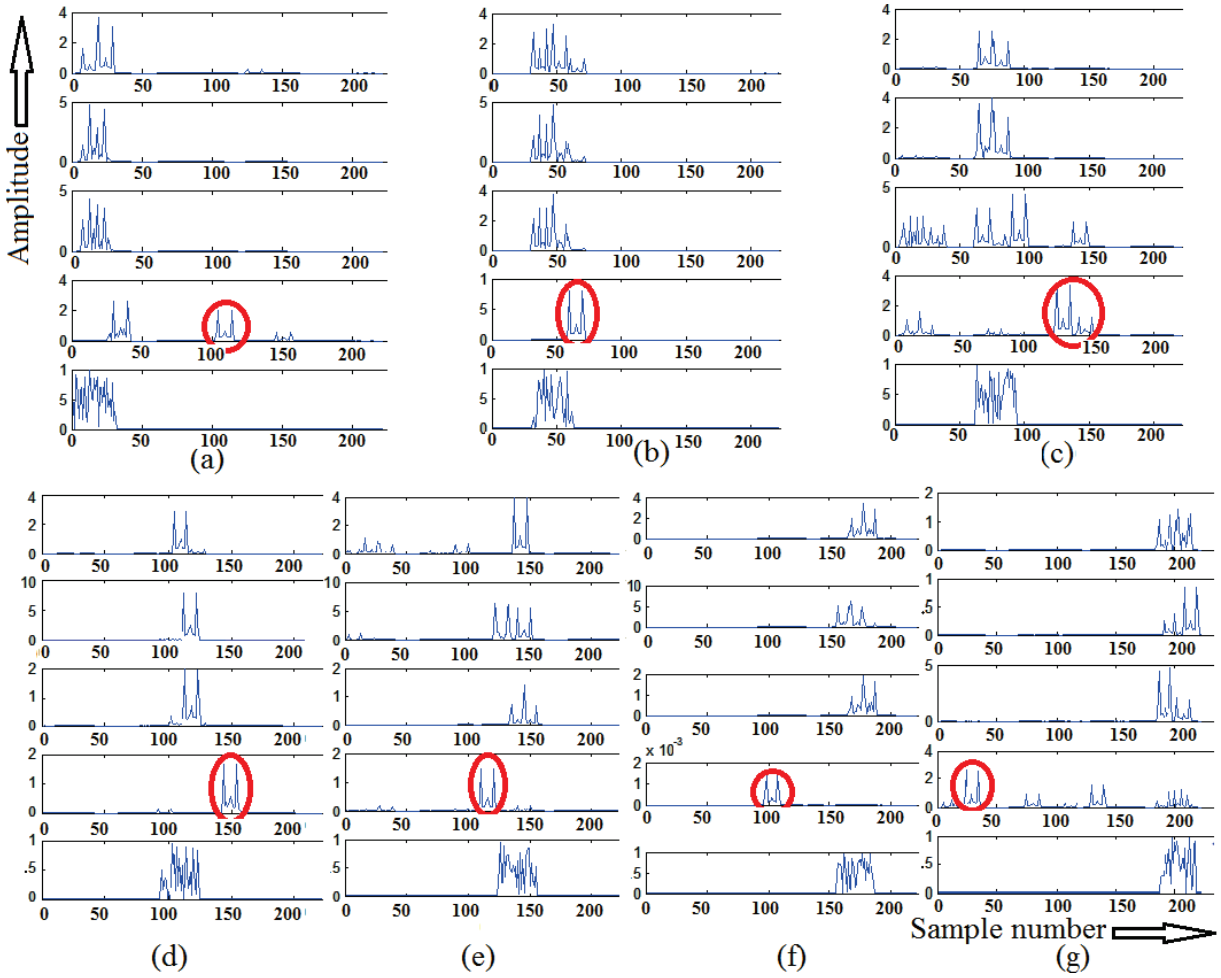

Fig. 10. Image segmentation dataset: The first four rows of (a), (b), (c), (d), (e), (f) and (g) are output of the LDS for class 1, 2, 3, 4, 5, 6 and 7 data respectively. The fifth row of (a), (b), (c), (d), (e), (f) and (g) are the intended ones or targets predetermined for class 1, 2, 3, 4, 5, 6 and 7 respectively. The red circle signifies the misclassified data and the mismatched region compared with the target.

[4] Houqiang Li, Zhengkai Liu and Weidong Sun, "A New Approach to Pattern Recognition of Remote Sensing Image using Artificial Neural Network," Geoscience and Remote Sensing Symposium, IGARSS 93, vol. 2, pp. 713-715, 1993.

[5] E.S. Gopi, "Artificial Intelligence," in *Algorithm Collections for Digital Signal Processing Applications using Matlab*, Dordrecht, Netherlands, Springer, ch. 1, sec. 1, pp. 1-8, 2007.

[6] D.H. Hubel and T.N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," The Journal of physiology, vol. 160, pp. 106-154, 1962.

[7] E.S. Gopi and P. Palanisamy, "Maximizing Gaussianity using kurtosis measurement in the kernel space for kernel linear discriminant analysis," Neurocomputing, vol. 144, pp. 329-337, http://dx.doi.org/10.1016/j.neucom.2014.05.020, 2014.

[8] PIX4D support site, Example Datasets Available for Download: Agriculture—https://support.pix4d.com/hc/en-us/articles/202561429-Example-Datasets-Available-for-Download-Agriculture#label1>, 2016.

[9] M. Lichman, "UCI Machine Learning Repository," Irvine, CA: University of California, School of Information and Computer Science, http://archive.ics.uci.edu/ml, 2013.

[10] David D. Lewis, Yiming Yang, Tony G. Rose and Fan Li, "Rcv1: A New Benchmark Collection for Text Categorization Research," Journal of Machine Learning Research, vol. 5, pp. 361-397, 2004.

[11] Anuj Gupta, "Classification Of Complex UCI Datasets Using Machine Learning And Evolutionary Algorithms," International Journal of Scientific and Technology Research, vol. 4, issue. 5, May 2015.

[12] Madhusmita Swain, Sanjit Kumar Dash, Sweta Dash and Ayeskanta Mohapatra, "An Approach for Iris Plant Classification using Neural Network," International Journal on Soft Computing (IJSC), vol. 3, February 2012.

[13] Agapito Ledezma, Ricardo Aler, Araceli Sanchis and Daniel Borrajo, "Empirical Evaluation of Optimized Stacking Configurations," 16th IEEE International Conference on Tools with Artificial Intelligence, pp. 49-55, 2004.

[14] Mridu Sahu, Shreya Sharma, Vyom Raj, N.K. Nagwani and Shrish Verma, "Impact of Discretization on Classification of Data using Divide and Conquer Paradigm," International Conference on Electrical, Electronics and Optimization Techniques (ICEEOT), 2016.

[15] Ting Wang, Sheng-Uei Guan, Ka Lok Man and T.O. Ting, "EEG Eye State Identification Using Incremental Attribute Learning with Time-Series Classification," International Symposium on Computer, Consumer and Control, pp. 158-161, 2014.

[16] Kadir Sabanc and Murat Koklu, "The Classification of Eye State by using kNN and MLP classification models according to the EEG Signals," International Journal of Intelligent Systems and Applications in Engineering, pp. 127-130, 2015.

[17] P. Sussner and E.L. Esmi, "Constructive Morphological Neural Networks: Some Theoretical Aspects and Experimental Results in Classification", in *Constructive Neural Networks*, Studies in Computational Intelligence, Berlin/Heidelberg, Germany, Springer Verlag, vol. 258, pp. 123-144, 2009.