On non-iterative training of a neural classifier Part-II:Clustering of points and their classification using an NN architecture

Conference Paper · September 2017

CITATIONS
0

READS
33

2 authors, including:

K. Eswaran
Sreenidhi Institute of Science & Technology
56 PUBLICATIONS **185** CITATIONS

SEE PROFILE

On non-iterative training of a neural classifier Part-II:Clustering of points and their classification using an NN architecture

K Eswaran and K Damodhar Rao
Department of Computer Science and Engineering
Sreenidhi Institute of Science and Technology
Yamnampet, Ghatkesar, Hyderabad-501301,India
Email:kumar.e@gmail.com and kanduladamodar@gmail.com

***Abstract*—As described in Part I of this two paper series, an algorithm was recently discovered, which separates points in n-dimension by planes in such a manner that no two points are left un-separated by at least one plane. By using this new algorithm it can be shown that there are two ways of classification by a neural network, for a large dimension feature space, both of which are non-iterative and deterministic and one could apply both these methods to a classical pattern recognition problem and present the results. It is expected these methods will now be widely used for the training of neural networks for Deep Learning not only because of their non-iterative and deterministic nature but also because of their efficiency and speed and that they may supersede other classification methods which are iterative in nature and rely on error minimization.**

***Keywords*—non-iterative training, neural networks, pattern recognition, algorithms**

I. Introduction

Researchers like Fisher [1] and Mahalanobis [2] and McCullough and Pitts[3], Kolmogorov [4], Werbos[5], and Rumelhart, Hinton and Williams [6] and others [7], also the Deep Learning people[8] had tried to separate the clusters by planes. Though statisticians try to approximate the shape of each cluster as an ellipsoid or even as a simple sphere, clusters in general would require more parameters to mathematically correctly define their shapes than the number of coefficients required to define the planes which are supposed to separate them. So all along it was, perhaps, very naive of all of us to have tried to separate clusters when such entities are not mathematically well defined. It is far better to separate the individual points and to use the enormous space and degrees of freedom that is available in n-dimension space to separate each point, rather than try to separate clusters which will never be well defined. The above arguments was the genesis of the idea that gave an impetus to do the kind of research work in [9-12] and that which is being presently reported in this paper. This direction has been greatly encouraged by the recent discovery of an algorithm which enables one to separate any number of given points in large dimension space in such a manner that every point is separated from every other by at least one plane. The best part of this algorithm is that it is non-iterative and finds the coefficients of the equations of each one of the planes that separate a given set of points; with a computational complexity of approx. $O(n.Nlog(N)) + O(n^3log(N))$, where N is the given number of points and n is the dimension of space. (The steps of the algorithm and its detailed proof are given in [9-12]).

The subject of this paper is pattern recognition and the purpose of this paper is to show that by means of the above newly discovered algorithm one can build a noniterative classifier which can assign sample points in data to an appropriate (correct) class. Actually, as stated in the abstract, there are two methods which can be used for the classification of a new test-sample (point): (i) The first directly uses the information obtained from the planes that separate the training-sample points each of which are assumed to have a class label and then tries to classify the new test-sample. (ii) The second method is based on the first, but uses another "Bubble Discovery" algorithm to find the bubbles which belong to each class in the training-sample feature space and then proceeds to perform a classification by trying to find out the bubble that the test-point would most probably belong to. The second method is more like the traditional neural method, but it is non-iterative and deterministic. The first method would seemingly be like a nearest neighbour algorithm, but this is not so as this method determines the exact quadrant to which a neighbouring point belongs to and not just the distance. As will be clear (later on) in $n-$dimension space there are 2^nquadrants surrounding any neighbouring point, (which is a large number), in the present method since each point was separated by planes in such a manner that the quadrant information is taken into account, where as a nearest neighbour method loses all quadrant and directional information, by reducing everything to a single number: the distance. It is this aspect that makes the present method superior as actual application reveals.

In paper-I, 'N' points were separated in G space by using the hyperplanes. In this paper points which belong to the same class are regrouped into clusters. To regroup the points, one could use the algorithm called "Bubble Discovery algorithm". Each cluster is in spherical form. Then clusters are separated by using the planes obtained in the paper-I. From this an NN architecture is obtained which is non-iterative.

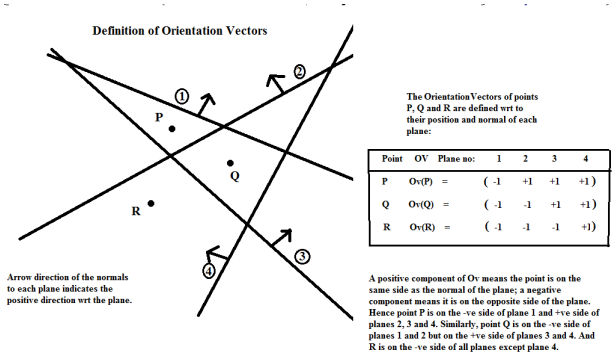
II. A neural classifier by using separation of individual points

In this method the process is as follows:

(i) All the data which consists of points in n-dimensional sample space is divided into two sets (a) a set of points,say N in number which will be considered as the "Traning Set" and (b) another set of points which will be considered as the "Test Set". It will be assumed that each point in the Training set belongs to some class which is known. The points in the Test Set are assumed to be belonging to some unknown class which has to be discovered by the classifier.

(ii) The new algorithm is then used to examine the co-ordinates of each point in the Training Set and then to find the coordinates of the planes that would separate each of the N points in this set so that each point is separated from the other by atleast one plane, let q be the number of planes that separate these points.[1]

(iii) Once all the q planes and their equations have been found out the Orientation Vector (OV) of each of the N points is found. The OV is a Hamming vector, of dimension q, which is associated with each point P belonging to the Training Set. If P is a point in the Training Set then its orientation vector $Ov(P)$ indicates the "orientation" of the point P w.r.t. each of the q planes See Figure below.



The Orientation Vector[2] has the property that two points (say) P and R will *not* have a plane between them *if and only if* $Ov(P) = Ov(R)$. That is $Ov(P) \neq Ov(R)$ automatically implies that P and R are separated by at least one of the q planes which separate each of the points in the Training Set.[3]

The algorithm that finds the planes which separate all the points is given in Part I, and [9 ,10] and will not be repeated here.

After the points are separated and the planes that have performed the separation are known then the method to process the classification of new points is easily devised. One can use the information obtained from the planes that separate the training-sample points each of which are assumed to have a class label and then try to classify the new test-sample.

A. An application to Data Classification and Retrieval

This sub section briefly describes how this algorithm can be applied to data retrieval. Suppose one has N points in a n dimension X-Space and the algorithm was used to separate all N points and it was found after running the algorithm that q planes were finally necessary. This information can be used as a data retrieval device. That is, all the data can now be stored in such a manner that retrieval can be done with great efficiency.

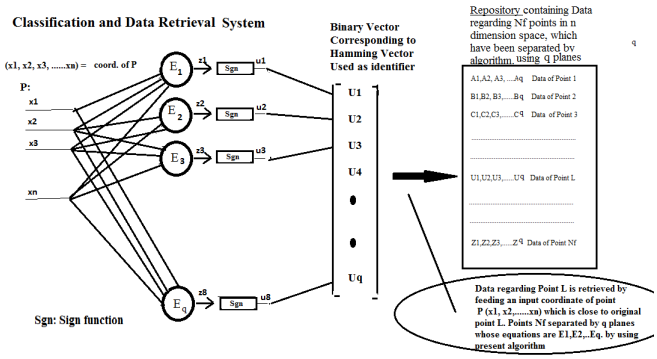
Now for purposes of this illustration one will assume that each point N represents one n dimensional sample in X-Space. This sample, data point, will have n numbers which may relate to a medical data of one person or alternatively it could be an image involving n pixels and represent a photograph of a person.Now one wishes to store many such samples, data points, in such a manner that classification and retrieval becomes easy. The idea is simple: after one runs the algorithm which separates each of the N data points (samples) by q number of planes, one would have the exact information of how each of the data points N reside in X-Space with respect to each other and the separation planes, because the Orientation Vector information for each point is available in S. One can use this information to (i) store and classify the data in such a manner that it is possible to (ii) retrieve it easily. What it means is that after the storage is done and if fresh (approximate) data of a person is given whose data is stored in the storage receptacle in the repository, it is possible to use this new (approximate) data to retrieve the stored data (image). In other words if a new sample point is given, one can retrieve another examplar which is closest to the present sample point. Example if the new sample point is P and it may represent the medical data of a person P, then one can retrieve another data point of a person L (stored in the data base), which is closest to the present sample point P.

If this new sample point P in X-space, is close to some other sample point L (say) stored in the receptacle then one can discover this fact by calculating $Ov(P)$, the Orientation Vector of P and taking a "dot product" with the Orientation Vector $Ov(Q)$, of all other points Q stored in the data base. Then by comparison one will find a point L such that the dot product $Ov(P).Ov(L)$ is a maximum. Oviously then the point L is the one closest to the sample point P and hence in all probability points P and L will belong to the same class. This is how one can associate the class of the new sample P with the class of point L, the latter being known. The above steps can be represented as a diagram which is nothing but a neural engine in the picture below:

[1]The number of planes,q, required to do this would be such that $q = O(log_2(N))$ this empirical estimate of q gets better when the number of dimensions, n, gets bigger. (For eg. for a particular set of random points, generated by the authors, where $N = 2,000$ and $n = 15$, it was only 22 planes (i.e. $q = 22$) where as for another case when $N = 50,000$ and $n = 25$ it was found that only 5 more planes ie 27 planes($q = 27$) separated all the points, this increase is as per the empirical formula since $log_2(50,000/2000) = log_2(25) = 4.65 \approx 5$) planes.)See Ref[12] for details.

[2]It is assumed that every plane will have a specific (defined) normal direction, a point which lies on the positive side of the normal is said to lie on the positive side of the plane, on the other hand if the point lies on the other side it is said to lie on the negative side.This direction is easily found if the equation of the plane is known for example if $1 + \alpha_1 x_1 + \alpha_2 x_2 + ..., +\alpha_n x_n = 0$ is the equation to a particular n- dimensional plane then a point P whose coordinates are $(p_1, p_2, ..., p_n)$ and is not on the plane, will be said to be on the positive side of this plane if $1 + \alpha_1 p_1 + \alpha_2 p_2 + ..., +\alpha_n p_n > 0$ and in the negative side if $1 + \alpha_1 p_1 + \alpha_2 p_2 + ..., +\alpha_n p_n < 0$.

[3]Another fact, that only adds to the prospect of success in this new direction is that: for large n dimension space where $2^n > N$ (N being the number of points), one will find that the number of planes q, needed for separating N points is far less than the number of points itself, in fact q = O(log2(N)).



In the figure E_1 represent the equation to the first plane stored in S; Assume that the equation to this plane is given by

$$1 + \alpha_1 x_1 + \alpha_2 x_2 + ..., + \alpha_n x_n = 0 \tag{1}$$

then one can define

$$z_1 = 1 + \alpha_1 x_1 + \alpha_2 x_2 + ..., + \alpha_n x_n \tag{2}$$

Sgn (Sign) function can be defined as: $Sgn(z)$ defined s.t. $Sgn(z) = 1$, if $z > 0$ and $Sgn(z) = -1$ if $z < 0$ and define $U_1 = Sgn(z_1)$, then U_1 is 1 if the point P is on the +ve side of plane 1 and is -1 if it is on the -ve side of plane 1. the same goes for the other planes $E_2, E_3, ..E_q$. So the output array $(U_1, U_2, U_3, ...U_q)$ is nothing but a Hamming Vector which is the orientation vector $Ov(P)$. Therefore the "Storage Plan" for each point L is: use the Hamming Vector which is the Orientation Vector $Ov(L)$ of a point L as its label (like a binary label); this label is like a pointer to the information about L stored in a receptacle in the space next to this label of L for easy retrieval.

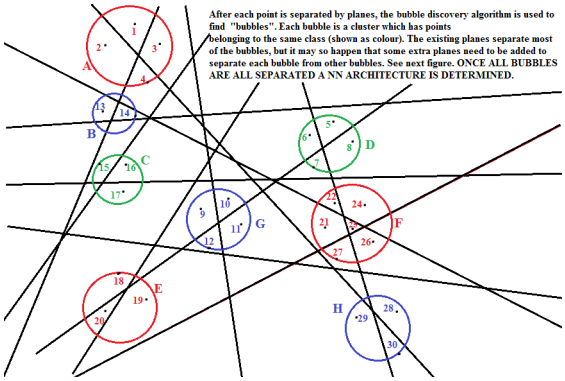
"Retrieval Plan": Present an approximate image of L say P to the network. It is then possible to immediately retrieve the information about L.

Since, this procedure was already explained in Sec. V (Iris problem) of paper I, it is not repeated here.

III. Classification by using a Bubble Discovery algorithm in addition to the point separation method

In this method one can use the information of the OV's by application of the previous algorithm, to form pure bubbles of the training data, in such a manner that each bubble has within itself only points belonging to the same class. This is done by another "bubble discovery" algorithm which is of complexity $O(N^2.log_2(N))$. After which it is easily possible to determine the architecture of a three layer neural network of processing elements and write down the weights of each processinng elements without any more ado. It may be recalled that the weights of the processing elements are nothing but the coefficients of the planes that separate the bubbles, since these are known the weights are known. Thus one has obtained a non-iterative and deterministic method of building a neural classifier.

The Bubble Discovery Algorithm: This is a simple algorithm for obtaining "pure bubbles" from data. The meaning of a pure bubble is a spherical region containing only points belonging to one class. if the collection of input points are given then the algorithm finds the "bubbles" which separate points belonging to one class from points belonging to a different class.



A set of data points are given in G belonging to various classes.

Step 1: Find the distance from each point 'p' from the set G to all other points in the set G. If set G has 'n' points, then distance-matrix of size n x n is obtained.

Step 2: Sort the distances of each row. Let us consider row_i = $[d_{i1}, d_{i2}, ..., d_{in}]$. Here d_{ij} is the distance from point 'i' to point 'j'. After sorting, row_i = $[d_{i1}, d_{i4}, d_{i7}, d_{i9}, d_{i2}]$. It means that point '1' is nearest to the point 'i', point '4' is next nearest to the point 'i', and so on.

Step 3: In each row_i, count the first 'm' number of consecutive points, whose class is same as the class of point 'i'. Consider the row_i = $[d_{i1}, d_{i4}, d_{i7}, d_{i9}, d_{i2}]$. Points i, 1, 4, and 7 belongs to class '1', point '9' belongs to class '3' and point '2' belongs to class '2'. So make count of row_i as four.

Step 4: (i). Pick the row_i, whose count is highest.

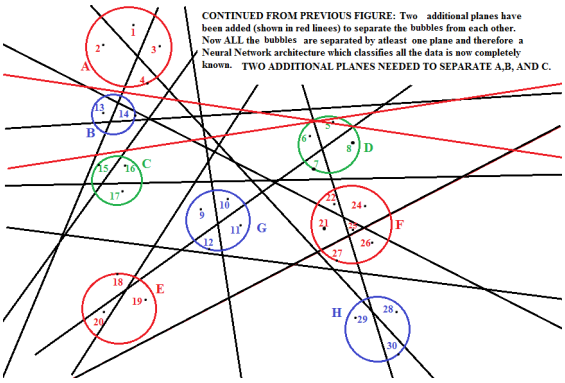
(ii). Consider row_i = $[d_{i1}, d_{i4}, d_{i7}]$. Create a new bubble by taking $point_i$ as bubble center and add $point_j$(j=1, 4, 7) to the bubble, when $point_j$ does not belong to any other bubble, otherwise stop adding point to the bubble. Record the radius of the bubble by considering the distance between the bubble center to the last added point. If all points are covered by the bubbles, then stop the algorithm, otherwise goto step4(iii).

(iii). Skip the rows whose points are already covered by the bubbles such as row numbers i, 1, 4 and 7 and pick the row, whose count is next highest, and goto step4(ii).

This algorithm finds the bubbles (spherical clusters) of various sizes each bubble having points belonging to the same class and a known radius.

Once all the points are separated by planes. This algorithm will discover the small pure bubbles, now most of the bubbles will be separated from one another by the existing planes. However it will some times be required to add additional planes so that two bubbles are separated by at least one plane. These additional planes are shown in red in the figure below. A simple algorithm can be used to separate two bubbles (say bubble A from B) by drawing a plane perpendicular to line joining the centers of A and B, the position of this plane can be suitably chosen to be outside the two bubbles. The OVs of

each point can be used to discover which particular bubbles need to be separated by additional planes.(See Figure below:)



NOTE: In order to check if the bubbles are separable by planes one needs to calculate the bubble OV's, to do this one can adopt the same procedure as that of points, that is if a bubble is to the +ve side of plane no. 5 (say) its 5th component will be +1 and if it is on the negative side of plane 5 it will be -1. However if a plane passes through the bubble then its component is made 0, example if plane 7 passes through the bubble the 7th component is made 0, when this happens we say that the plane 7 is **disabled** for that bubble. The bubble OV is the common OV of the points in the bubble after disabling the planes that pass through the bubble. The procedure is adopted to quickly find if every two bubbles say A and B are separated from one and another a requirement which will be met if atleast one plane (which is not disabled wrt to A and B) separates the two bubbles. (Of course while separating points by planes no planes are disabled because it is ensured that no plane passes through a point, but this is not guaranteed for extended objects like bubbles).

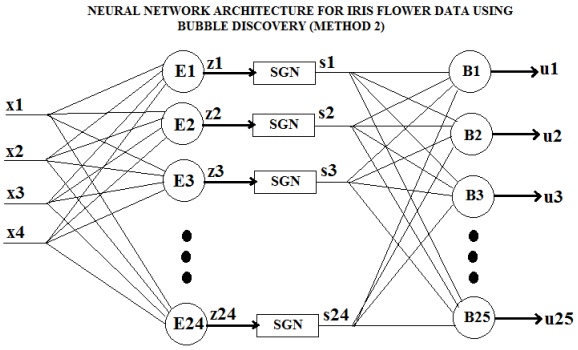
By using the above method planes are obtained which separate bubbles after they have been discovered by the "Bubble Discovery" algorithm. The methods described in Ref [16] one determines the neural architecture which can classify the data points. Very importantly in this case the coefficients of all the planes are already known, hence the weights of the processing elements are already determined and it is easy to merely write down the neural architecture. This method is non iterative.

A. Results on the IRIS FLOWER Classification Problem

The above method was applied for the same IRIS Flower data set [13-15] (see Appendix for data). In this case it was not found necessary to introduce more planes to separate the bubbles, 24 planes were sufficient. And the final number of bubbles was 25.

In the following figure E1 represents the equation to the first plane stored in S; Assumed that the equation to this plane is given by $1 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 = 0$

then one can define $z_1 = 1 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4$ Using the sign function Sgn(z) defined as: $Sgn(z) = 1$, if $z > 0$ and $Sgn(z) = -1$ if $z < 0$ and define $s_1 = Sgn(z_1)$, then s_1 is 1 if the point P is on the +ve side of plane 1 and is -1 if it is on the -ve side of plane 1. The same goes for the other planes$E_1, E_2, E_3..., E_{24}$ One can examine all the out puts $u_1, u_2, u_3..., u_{25}$ obtained from the orientation vector Ov(P) (since by definition $u_j = Ov(P).Ov(B_j)$) and one can choose that u_j which has the maximum positive value (dot product). Supose u_j is maximum then bubble B_j is considered as the "attractor" and the input point P, whose coordinates are (x_1, x_2, x_3, x_4) is considered to belong to that class which B_j belongs. If u_j value is same for two or more bubbles, then find the distance between the point P and the bubble center B_j (j=1,2,..., whose u_j is same). Now classify the point P by assigning class of the bubble, which is nearest to the point P. In this manner all the 29 test points were classified.



Procedure and Results: A total of 25 bubbles were discovered. In this particular IRIS case it was not needed to draw any more planes to separate the bubbles. All bubble OVs were unique.

The crucial and essential point to be remembered is that the procedures described in this paper is non-iterative and leads to a NN architecture which classifies the data.

The given train data was tested on the neural network shown in the figure, by first applying it on 120 training points. All 120 points were correctly classified. Then the test data was passed through the neural network i.e, each of the 29 test points was given as input to the NN. Out of 29 points, two points (point nos. 85 and 120) were wrongly classified. All the other 27 points were correctly classified.

B. Comparison with other methods (KNN and Classical NN)

The K-nearest neighbors (KNN) takes much more time than the Method 1 because distances of the test data point to every train data point needs to be calculated, whereas Method 1 needs to compute only Hamming vectors.

Method 1 takes $Nlog_2(N)$ calculations to find the planes which separate N points. Method 2 would take an additional calculations of $N^2log_2(N)$ to create bubbles. Both these methods are non-iterative and they gives the neural architecture two. Since the coefficients of each plane are nothing but the

weights of the associated processing elements. Therefore it is much more advantageous than the back propagation technique which is used after 'guessing' an architecture and then using iteration for training. Execution time for a few typical cases are given in conclusion section for comparison.

Since every cluster is being approximated as a collection of pure bubbles one can use this method for even linearly non separable problems. Each bubble is separated from other bubbles by planes. Therefore this is tantamount to the separation of bubbles by numerous (piecewise) planes.

IV. Latest work and Conclusions

Subsequent to sending this paper to the conference, this algorithm was recently applied on the digit recognition data set MNIST[17]. This dataset has 60,000 train points and 10,000 test points. In this data set, the original dimensions of each sample was 28 X 28. It was reduced to 6 X 6 dimensions. Results of the Method 1 can be found in the Paper-I of this two part series. In this Method 2 it was found that the total number of bubbles were 51030. It took 65 planes (same planes which separated the 60,000 training points) to separate all the bubbles. (The code is being fine-tuned to reduce the execution time). All train points were correctly classified. The classification accuracy of the test points was 93.03%. The time taken to classify the test points was 0.6 minutes. The details of this result can be found in[18]. This Method 2 was also applied on the Face recognition Extended Yale Database[19] and approximately 10,000 pure bubbles were identified in the train data and an accuracy of 80% on the test data was obtained. The results are being sent for publication [20].

In this paper the new algorithms were used, which separate the clusters by planes to develop non-iterative classification techniques for solving pattern recognition problems using neural networks. The bubble discovery algorithm is very new and will find application in neural research and Deep Learning not only because of non-iterative and deterministic nature but also because of their efficiency and speed. It is strongly felt that this algorithm has the potential to supersede other classification methods which are iterative in nature and rely on error minimization.

Acknowledgment

The authors thank the management of Sreenidhi: Dr P. Narsimaha Reddy and Dr K.T Mahi for their sustained support; the HOD Dr Aruna Varanasi and their colleagues for creating a happy environment. Both the authors K.E. and K.D.R, thank their spouses Suhasini and Amani, resp. for their unwavering faith and encouragement.

References

[1] R.A. Fischer: The statistical utilization of multiple measurements, Annals of Eugenics, 8, 376-386 (1938); also Annals Eugenics, 7, 179-188 (1936)

[2] P.C. Mahalanobis: On the generalized distance in statistics, Proc. Nat. Inst. of Sc. India, 12, 49-55 (1936)

[3] McCullough, W. and Pitts, W. A: Logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 5:115133. (1943)

[4] A. N. Kolmogorov: On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition. Doklay Akademii Nauk USSR, 14(5):953 - 956,(1957). Translated in: Amer. Math Soc. Transl. 28, 55-59 (1963).

[5] PaulWerbos: Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, PhD thesis, Harvard University, 1974

[6] Rumelhart, D. E., Hinton, G. E., and R. J. Williams: Learning representations by back-propagating errors. Nature, 323, 533536 (1986).

[7] J. Schmidhuber: Deep Learning in Neural Networks: An Overview. 75 pages, www.arxiv.org-slashabs-slash1404.7828 (2014).

[8] Yoshua Bengio: Learning Deep Architectures for AI. Foundations and Trends in Machine Learning: Vol. 2: No. 1, pp 1-127 (2009). 7

[9] K.Eswaran: A system and method of classification etc. Indian Patents Nos.(1) 1256/CHE July 2006 and (2) 2669/CHE June 2015

[10] K.Eswaran: On a non-iterative algorithm for separation of points in n-dimension space by planes , 2015 arxiv.orgslash-abs-slash1509.08742

[11] K.Eswaran: Indian patents filed. (1) No. 2669/CHE/2015, A system and method for classification of data and (2) No. 201641015775, An Universal classifier for learning and classification

[12] K. Eswaran: Results of Separation of points in high dimension space from a computer program based on the non-iterative algorithm, Tech Report: www.researchgate.net(slash)publication(slash)285149835 (slash)Results

[13] R. A. Fisher (1936). "The use of multiple measurements in taxonomic problems" (PDF). Annals of Eugenics 7 (2): 179-188. doi:10.1111/j.1469-1809.1936.tb02137.x.

[14] Edgar Anderson (1936). "The species problem in Iris". Annals of the Missouri Botanical Garden 23 (3): 457-509. JSTOR 2394164. http: archive.ics.uci.edu-slash-ml-slash-machine-learning-databases-slash-iris

[15] Edgar Anderson (1935). "The irises of the Gasp Peninsula". Bulletin of the American Iris Society 59: 2-5.

[16] K Eswaran and Vishwajeet Singh. Article: Some Theorems for Feed Forward Neural Networks. International Journal of Computer Applications 130(7):1-17, November 2015.

[17] MNIST data were taken from http://yann.lecun.com/exdb/mnist/

[18] K Eswaran, D Hiranmayi, K Damodhar Rao, T Vishwajeet Singh, T Sneha and C Swetha: Application of Non Iterative classification algorithms for digit recognition on the MNIST dataset, being sent for publication.

[19] Extended yale data were taken from http://vision.ucsd.edu/ iskwak/ExtYaleDatabase/ExtYaleB.html

[20] K Eswaran, D Hiranmayi, K Damodhar Rao, T Vishwajeet Singh, T Sneha and C Swetha: Application of Non Iterative classification algorithms for face recognition on the Extended yale face dataset, being sent for publication.

V. APPENDIX PARTICULARS OF IRIS DATA AND PLANE COEFFICIENTS

Iris data were taken from http: (slash)(slash) archive.ics.uci.edu (slash) ml (slash) machine-learning-databases (slash) iris.

Points 102 and 143 had the same coordinates, so point 143 was removed. The order of points from 1 to 142 were same as before, but the points 144 to point 150 were renumbered as 143 to 149.

For testing 29 points were used which are: 5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100, 105,110,115,120,125,130,135,140 and 145 (ie is took every 5th point as a test point to avoid bias). For training 120 points were used which were the points other than testing points.

All the equations of the planes are of the form: $1+\alpha_1 x_1+ \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 = 0$.

The coefficients $\alpha_1, \alpha_2, \alpha 3, \alpha_4$ which were found by the separation algorithm [10-11] are given below for all the 24 planes:

1 -0.1711351 0.0 0.0 0.0

2 0.0 -0.3270824 0.0 0.0

3 0.0 0.0 -0.2660989 0.0

4 0.0 0.0 0.0 -0.8337965

5 0.0915693 -0.1799810 -0.9820018 2.3997473

6 -3.01654425 3.39045923 3.27472364 -5.11462274

7 -1.06402164 -1.24436429 1.63510670 1.03997595

8 -0.0382210 -0.1219001 -0.0407087 -0.1519794

9 0.0229477 0.0500678 -0.2816313 -0.0083446

10 -0.1706478 -0.1565447 0.1526663 -0.1653591

11 0.0075358 -0.6014673 -0.0748200 0.6106179

12 0.1618563 -0.3735359 -0.1043258 -0.2218643

13 0.1836043 -0.9878048 -0.3770325 1.3102981

14 -0.2599430 0.0116093 0.2707517 -0.4116654

15 -0.2889498 -0.1879260 0.2721125 -0.0043451

16 -0.1352105 -0.1317516 0.0658758 -0.0248410

17 -0.5241379 0.7586206 -0.3172413 -2.5517241

18 -0.2380952 0.3846153 -0.5402930 -1.0256410

19 -0.0861793 -0.0127673 -0.3702521 0.1947015

20 -0.9523809 2.8571428 -4.7619047 1.9047619

21 -2.3953323 2.6854845 2.7375226 -4.1047070

22 -0.5644084 0.2080952 0.3685541 0.0940926

23 -0.0959491 -0.0793593 -0.3909716 0.8595098

24 -0.5855213 0.4343851 0.3979970 -0.3358711

(i) List of Bubbles

The following list contains the points within each bubble. The suffixes indicate the bubble number. For example bubble number 17, contains points:128,139. THE TOTAL NUMBER OF BUBBLES = 25.

$(1, 18, 28, 29, 41, 8, 38, 22, 49, 27, 47, 36, 12, 11, 32, 37, 21, 44,$
$24, 3, 7, 2, 31, 26, 17, 48, 13, 46, 6, 33, 4, 23, 19, 43, 34, 39, 9,$
$14, 16, 42)_1,$

$(63, 93, 68, 81, 83, 54, 88, 82, 72, 69, 91, 97, 98, 56, 96, 74, 89,$
$62, 79, 64, 92, 73, 61, 59, 67, 76, 94, 66, 52)_2,$

$(106, 123, 108, 136, 119, 131, 126, 118, 103, 132, 143, 121, 109, 1$
$113, 144, 141, 101, 129, 133, 117, 138, 104, 142, 137, 147, 116,$
$112, 111)_3,$

$(148)_4, (99)_5, (58)_6, (87)_7, (51)_8, (53)_9, (77)_{10}, (86)_{11}, (122)_{12},$
$(127)_{13}, (57)_{14}, (102)_{15}, (124)_{16}, (128, 139)_{17},$
$(146)_{18}, (149)_{19}, (78)_{20}, (114)_{21}, (71)_{22}, (84)_{23}, (107)_{24}, (134)_{25}$

The numbers calculated by the algorithm and given in this Appendix completely determined the NN architecture for the IRIS problem as shown in the last section of the paper and with results described therein.

Coordinates of the points: The coordinates of the first four points are

PointNo x1, x2, x3, x4

1 5.1 3.5 1.4 0.2

2 4.9 3.0 1.4 0.2

3 4.7 3.2 1.3 0.2

4 4.6 3.1 1.5 0.2

The coordinates of the remaining points can be found from the reference.

ORIENTATION VECTORS OF POINTS:

Orientation vector can be found once the coordinates of the point and the plane coefficients are known. The Orientation vectors of the first four points w.r.t the 24 planes are given below. (See foot note 2 on page 2, to recall how the OVs are defined.)

OV(1) =

1 -1 1 1 -1 1 -1 1 1 -1 -1 1 -1 1 -1 -1 1 1 1 -1 1 -1 -1 1

OV(2) =

1 1 1 1 1 -1 -1 1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 -1 -1

OV(3) =

1 -1 1 1 1 1 -1 1 1 -1 -1 1 -1 1 -1 1 1 1 1 -1 1 -1 -1 1

OV(4) =

1 -1 1 1 -1 1 -1 1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 -1 1

ORIENTATION VECTORS OF BUBBLES:

Orientation vector of first four bubbles are

Bubble 1:

1 0 1 1 0 0 -1 1 1 0 -1 1 -1 0 -1 0 0 0 0 0 0 -1 0 0

Bubble 2:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 -1 -1 -1 0 0 -1 0

Bubble 3:

-1 0 -1 -1 0 0 1 -1 -1 0 0 0 0 0 0 0 -1 -1 -1 -1 0 0 0 0

Bubble 4:

-1 -1 -1 -1 1 -1 1 -1 -1 -1 1 -1 -1 -1 1 1 -1 -1 -1 -1 1 1 1