(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
09 November 2017 (09.11.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/191648 A1

(51) International Patent Classification:
G06N 3/00 (2006.01) *G06F 15/18* (2006.01)
G06N 99/00 (2010.01) *G06F 17/30* (2006.01)

(21) International Application Number:
PCT/IN2017/000096

(22) International Filing Date:
28 April 2017 (28.04.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
201641015775 05 May 2016 (05.05.2016) IN

(72) Inventor; and
(71) Applicant: **ESWARAN, Kumar** [IN/IN]; C 91, Second Crescent Road, Sainikpuri, Secunderabad, Telangana 500094 (IN).

(74) Agent: **RAVI, S**; IN P/A:- 853, No.10, Ramalinga Nagar, II CrossII Layout, Saibaba Colony, Saibaba Mission(P.O), Coimbatore, Tamil Nadu 641011 (IN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

(54) **Title:** AN UNIVERSAL CLASSIFIER FOR LEARNING AND CLASSIFICATION OF DATA WITH USES IN MACHINE LEARNING

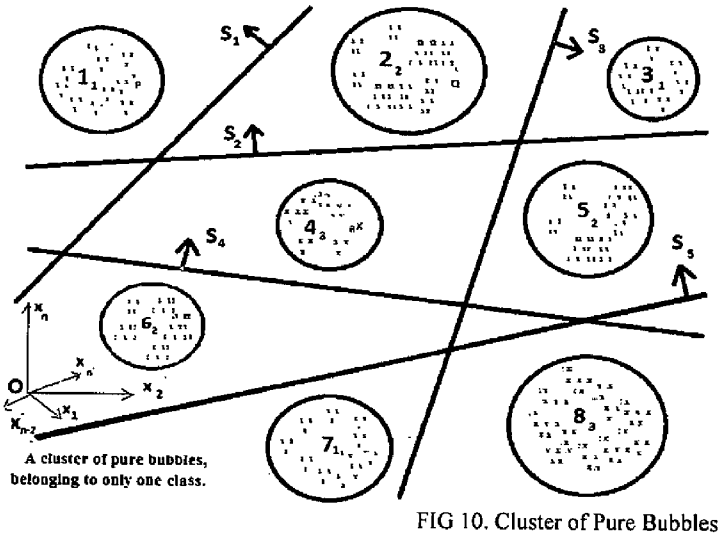


FIG 10. Cluster of Pure Bubbles

(57) **Abstract:** The present invention discloses a new methododlogy which can be used to classify specific problems and describes a product such as the "classification engine" which is implementable in hardware for specific problems. The specific problem could be such as face Recognition Systems, Disease diagnostic systems, Robotic inspection systems for use in the factory shop floor etc. The system and method given in this invention examines any form of data for the purposes of learning and classification so that any Machine such as a robot or machine will be able to handle the data and classify the data for automatic decision making. While analyzing the data it performs, basically THREE main steps (I) The Partition Process and (II) The Reduction of Dimension (of data) Process, (III) The Cluster Discovery Process and (IV) The classification Process.

[Continued on next page]

WO 2017/191648 A1

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*
— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

TITLE OF THE INVENTION

An Universal Classifier for Learning and Classification of data with uses in Machine Learning

FIELD OF INVENTION:

The present invention discloses a new and novel methodology which can be used to classify specific problems and describes a product such as the "classification engine" which is implementable in hardware for specific problems.

BACKGROUND OF THE INVENTION INCLUDING PRIOR ART:

This invention is about An Universal Classifier for Learning and Classification of data with uses in Machine Learning. A typical classification or pattern recognition problem involves a multi-dimensional feature space. Features are variables: they can be, for example in a medical data, blood pressure, cholesterol, sugar content etc. of a patient, so each data point in feature space represents a patient. Data points will be normally grouped into various clusters in feature space, each cluster will normally belong to a particular class (disease), the problem is further complicated by the fact that more than one cluster may belong to the same class (disease). The problem in pattern recognition is how to make a computer recognize patterns and classify them. The problem in the real world can be extremely complex as there are thousands of clusters and hundreds of classes in a space of a hundred or more dimensions. Normally computers are used to detect patterns in such data and recognize classes for decision making by using Artificial Neural Network technique and an iterative algorithm called the Backpropagation technique to determine the architecture of a given problem.

OBJECTIVES OF THE INVENTION:

This inventions is based on the previous work done by the Inventor and as reported in above para. The inventor has found that the methods developed in the previous work logically leads to the discovery of very new algorithms which makes the classification technique deterministic. This is the finding that is being presently reported and is a major advancement in the field of Neural Research.

This invention of the MLCE rests on two crucial discoveries which were hitherto completely unknown and which made this innovation possible.

(i) The first, is the that it was discovered that if one is given a set of N points in d-dimensional space then it is possible to discover hyper planes (say q in number), such that every one of the N points is partitioned from all the others by these q planes. In fact, when the dimension, d, of space is large then the number of planes q, required to do this is only O(Log(N)). A non–iterative algorithm was discovered, which can be adopted to perform this task of partitioning, when given the N points and the coefficients of the hyper planes are then calculated using only O(NlogN) computations.

(ii) The second crucial discovery was that if, once again, one is given a set of N points in d-dimensional space this time each one of the points are labelled by say k labels; then it is possible to discover d-dimensional spheres, of various sizes, such that each of the spheres will contain only those points which belong to the same label. In other words the spheres

separate points belonging to one label from points belonging to another label. A non-iterative algorithm which perform this task using $O(N^2 \log N)$ computations was discovered.

Every classifier performs the task of partitioning of labelled points using various techniques ways e.g. statistical techniques, nearest neighbour methods, rule based methods and techniques used in "Deep Leaming, all of which involve numerous computations and iterations.

The two discoveries which partition points in a much more efficient manner, has led to an entirely new invention: The Machine Learning and Classification Engine (MLCE). This new Classification Engine is deterministic, non-iterative and will always run successfully, completing its task in polynomial time

In view of the above, one of the objective of the present invention is to develop the Machine Learning and Classification Engine (MLCE), which is an analytical engine that consists of a processor, devices for accepting data which may consist of image data, or, voice data or numerical data and a resident memory which contains machine coded algorithms which are executed by the CPU. The MLCE learns patterns in the data so that (after the Training Phase) it may classify new incoming data, of similar nature in order to decide as to which pattern or class the present incoming data belongs to. While running the MLCE will be basically always be in one of two states viz. (i) The Learning State (Phase) and (ii) The Decision Making State (Phase).

Another objective is to see that, when the MLCE is in the Learning State it analyses all the input data and tries to detect patterns in the data which it associates with known data. The analysis is done using four main processes: (I) The Partition Process and (II) The Reduction of Dimension (of data) Process, (III) The Cluster Discovery Process and (IV) The Classification Process. In some cases (II) may not be required in such a case there are only three processes. During the Learning State the analysis of incoming data is performed following the aforementioned processes and the incoming data has been "learnt" by the MLCE so that this knowledge can then be used to classify new incoming data in the Decision Making Phase.

Yet another objective is to make the processes during the Learning State involve the execution of new algorithms (as explained in detail in Para/Section 6 below), all these algorithms are non-iterative in nature so that the tasks in the Learning Phase is always completed successfully. A provision to learn new data and for retraining is also provided.

In summary, this invention is an entirely new method of classification, which partitions data belonging to different categories. In order to do this every sample is treated as a point in feature space which has as many dimensions as the number of features (parameters). For example, if the incoming data is medical data, the parameters belonging to a particular person such as: systolic pressure, diastolic pressure, white blood corpuscular count, red blood corpuscular count etc. are considered as distinct features and if there are say, 10 such parameters for each person then the data belonging to a particular person could be considered as a point in 10 dimensional space, whose 10 coordinates are nothing but the actual values of the said 10 features. The analysis in the Learning Phase is then done in this 10 dimensional feature space. The method is explained in detail in the section containing the detailed description of the invention. The method is not just confined to medical data but the MLCE can be trained to classify image and voice patterns too.

The system in the present innovated software-hardware system consists of a CPU (Arm Processor), which will be executing the processes indicated, viz: The Training Process which determines a Neural

Architecture which then is used for The Classification and the Decision Making Process. The input to the device could be data coming in various forms, either through camera feeds, audio feeds or as numerical data. The incoming data is stored in memory locations during the Training stages. After the Training phase a Neural Architecture is determined which classifies the data in the Classification Phase..
In the Decision Making Phase, new data which arrives in real time is examined by the MLCE to arrive at a classification/ decision of the input data. The MLCE can thus be used for classification of images eg. Face recognition, classification of voice data eg. Speech recognition, classification of numerical data eg. Disease diagnosis, etc. all done in real time. Thus the MLCE can be considered as powerful Intelligent Decision Making Analytical Engine which can learn as well as decide upon new data.

The computational complexity of the MLCE for the entire process is such that all the processes are completed in polynomial time. In the present invention, given a data set of sufficient samples it is possible to deterministically classify the labelled data with near 100 percent accuracy (provided the data is consistent).

DESCRIPTION OF THE INVENTION

This invention is about An Universal Classifier for Learning and Classification of data with uses in Machine Learning.
A typical classification or pattern recognition problem involves a multi-dimensional feature space. .Features are variables: they can be, for example in a medical data, blood pressure, cholesterol, sugar content etc. of a patient, so each data point in feature space represents a patient. Data points will be normally grouped into various clusters in feature space, each cluster will normally belong to a particular class (disease), the problem is further complicated by the fact that more than one cluster may belong to the same class (disease). The problem in pattern recognition is how to make a computer recognise patterns and classify them. The problem in the real world can be extremely complex as there are thousands of clusters and hundreds of classes in a space of a hundred or more dimensions. Normally computers are used to detect patterns in such data and recognise classes for decision making by using Artificial Neural Network technique and an iterative algorithm called the Backpropagation technique to determine the architecture of a given problem. This invention describes an entirely new methodology which can be used to classify specific problems and describes a product: the classification engine which is implementable in hardware for specific problems. The specific problems could be: Face Recognition Systems, Disease diagnostic systems, Robotic Inspection systems for use in the factory shop floor etc.

The method given in this invention examines any form of data for the purposes of learning and classification so that any Machine such as a robot or machine will be able to handle the data and classify the data for automatic decision making. While analysing the data it performs , **basically THREE main steps (I) The Partition Process and (II) The Reduction of Dimension (of data) Process, (III) The Cluster Discovery Process and (IV) The Classification Process**

We explain each one of the processes as follows:

Partition Process

(a) Use the Partition Algorithm to part each point from the other by hyper planes. So that no two points are un-parted by at least one plane. Each point will belong to its own quadrant. This step takes

approximately NLogN multiplications This algorithm shown in Fig. 1 to partition points by hyper planes uses the method of orientation vectors which was used in our earlier invention (disclosure No: 2669/CHE/2015, dated 28/5/2015). For purposes of clarity the definition of an orientation vector is again given below in the form of a figure.The Orientation Vector 2 as shown in Figure 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) 6= Ov(R) automatically implies that P and R are separated by at least one of the hyperplanes.

The algorithm that finds the planes which partitions is a direct application of the above properties of Orientation Vectors and is given subsequently in the following paragraphs of this specification. After the points are partitioned and the planes that have performed the partitioning are known then the method to process the classification of new points is easily devised as we shall see. We use the information obtained from the planes that partition the training-sample points each of which are assumed to have a class label and then try to classify the new test-sample.

The Reduction of Dimension of Data Process

One of the main difficulties of classification is the "curse of dimension". In most practical cases the data has too many dimensions. For example a medical history of a patient may consist of 50 to 100 parameters (ie 50 to 100 dimensions) and for images involving say a 30X 30 pixels involve 900 dimensions. In general a d –dimension space will have 2^d quadrants.
This is the "curse of dimensions".

So whenever it is required to reduce the dimension of the input data (this may or may be necessary in some cases), this Invention has provided a very ingenious method of dimension reduction, which rests on the partition algorithm described in the previous section, which will be presently described.

The procedure is as follows: We use the sample points provided to us as train data as the basis of a mapping which we explain as follows: Suppose we are given N sample points each of d-dimension and forming say k classes. We then use the partition algorithm to find the planes which part all the N points, suppose it turns out that we have q planes. In some cases especially when d, the dimension of the space is large q will be smaller than d, (q is the O(logN)), in such cases it is worthwhile to reduce the dimension of data. We then use the q planes as a basis to for a mapping which maps every sample point P from its original space of d-dimensions to a point P' in q dimensions.
The Fig. 3 given below shows the concept: Originally there are say N points in d-dimensional space, and suppose they are parted by using our partition by using q planes then then by taking the perpendicular distances s_1 $s_1,s_2,s_3,\ldots\ldots\ldots s_q$, of any point P from the planes (1,2,3.....q), we see that these distances serve as "coordinates" of the point P. Hence we can denote P not by its d-dimensional coordinate $(x_1,x_2,x_3,\ldots x_d)$ but by the q dimensional coordinate $(s_1\ s_1,s_2,s_3,\ldots,s_q)$, we will call this the "S-coordinate System" We are already assured that they are unique because all the points have been parted by the q planes, but the important point is that the S-coordinate sytsern is q-dimensional and in general for large dimension d, $q = O(\log_2(N))$ and in general $q << d$. Therefore since every sample point P,Q,R,..etc can be expressed in the S coordinate system using q- coordinates, we have essentially reduced the input dimension of data from d to q. For example: in the Example A it is shown the MNIST data which is d=784 dimensions and has around N=60000 points can be parted by just q=24 planes and hence the dimension of the input in this case is reduced from 784 to 24.

The Cluster Discovery Process

For the purpose of cluster discovery we use another **Bubble Algorithm.**
(a) The Bubble algorithm: In this algorithm examines every train point and sees its nearest neighbour, if it belongs to the same class it will include it in a sphere about the centroid. Then it searches for the next closest point, if the closest new point belongs to the same class as the other two it includes it in the same set, the centre is shifted to the centroid of 3 points. The process goes on till a closest point to the latest centroid does not belong to the same class then a sphere is drawn (about its previous centroid). By this process as shown in Fig. 4, all test data is partitioned into "pure bubbles", each bubble has points belonging to only one class. This takes approximately N^2 multiplications but is also non-iterative and will end successfully for every case.

(b) **The Cluster discovery algorithm.** In this algorithm if two bubbles belonging to the same class are neighbours they can be combined provided they can be parted from others by a hyper plane. We search from the possible hyper planes if such a plane exists, if not we do not combine the bubbles. In the worst case each bubble is one cluster as in (b) above. Actually, step (c), is not fundamental, it only serves to reduce the number of clusters and this step (c) is not strictly necessary, so we will not talk anymore about it. For our purposes the Bubble Algorithm will serve as our "Cluster Discovery" Algorithm, though it may give many numbers of bubbles (spherical clusters).

Classification Process

The classification can be done by two alternative techniques viz. method I and method 2 which are described in the following:
Method 1: This method only uses the partition result (a) above, and is a kind of nearest k-neighbour algorithm **but the search for the nearest neighbour is restricted to a very small subset of the train point. Whereas in the usual k-nearest neighbour method the search for the nearest point is exhaustive and is carried through the entire train data set involving many more computations of nearest Euclidean distance.** This subset contains those train points whose Orientation Vectors differ from that of the test point by only 6-8 components. **That is we search only those train points which are parted from the test point by only 6-8 planes. We then find that point which is closest (Euclidean distance) to the test point to classify the latter.**

Method 2: This method finds those planes which part the bubbles from those found in I(a) above and if needed add more planes. The process may require "disabling" some planes which pass through a particular candidate bubble. We finally end up with a smaller set of planes which can part each bubble and hence the
When each bubble is parted from the others by a plane, then a Neural Network architecture can be found whose weights are the coefficients of the planes without using Back Propagation.
Note in both the Methods 1 and 2 are non-iterative and end in a finite number of steps. The Neural Network for the above problem which consists of 25 train points belonging to three classes (red, blue and black) which form 11 clusters and parted by 8 planes assumed d-dimensional is d-8-11-3 is automatically and completely determined as shown in Fig. 5.
Software Implementation

The procedure for software implementation as shown in Fig. 6A is as follows, when data is first given, it is analysed by several algorithms and three main process are executed:

(I) The Partition Process and (II) The Reduction of Dimension (of data) Process, (III) The Cluster Discovery Process and (IV) The Classification Process.

This will give the number of clusters as shown in Fig 1. The processes when completed will give the number of partitioning hyperplanes separating all the train data (points), the number of pure bubbles and the number of pure clusters (if needed). Then by using the partitioning planes so obtained, new planes will be added if necessary, to partition all the bubbles (or clusters). The dimension of the input data will be reduced, if it is found advantageous to do so. After obtaining the final partitioning planes that partitions the bubbles that "cover" all the train data, a neural network architecture which classifies the entire data will be determined in a non-iterative manner by using all the information obtained by the analysis of the train data viz. the coefficients of all the hyper planes, the radii and locations of the centres of all the pure bubbles, and the class information of each training point which is given as input data. The details have been given in para 6 above and also in the details of each algorithm in paragraphs 7 and following below.

The above procedures which uses all the input information, are sufficient to find the architecture of the Neural Network classifier completely. The Neural Classifier is then implemented in a hardware device the latter could be termed as "A Machine Learning and Classification Engine" (MLCE) for the particular application.

Hardware Implementation

Hardware implementation of the Machine Learning and Classification Engine" (MLCE), will depend on the type of application. For example if one wants to use this method for face detection or terrorist detection we would need data from camera inputs. The algorithm will then be used to detect the person (ie classify different people). The output can come as a detected image or as an alarm. The hardware for this will be an embedded program working on a suitable process (such as ARM). The implementation of an MLCE, for this particular application will be as follows: A number of images of various personnel will be taken by cameras and stored as "Train Data", along with this collection a number of "Test Data", would also be stored. The 4 processes starting from the Partition Process and ending with the Classification Process will be implemented using the Train Data. This will determine the Neural Architecture, which will be tested using the Test Data. After successful testing the Neural Architecture will be implemented in Hardware using a suitable processor, say an Arm Processor. This Hardware implementation will consist of a camera which takes images of people entering say a factory (or protected area) and these input images will be sent to the Arm processor which will use the a Neural Architecture to classify the input image and provide an output on a screen or and initiate an alarm system in case of un-authorised entry. Similarly the input can also be voice data from microphones in case we wish to use the MLCE for voice identification or speech recognition. Of course the MLCE need to be trained separately for each of the different applications viz. face recognition or voice recognition.

Figure Description: The Fig. 6B shows a schematic of the Hardware implementation of the "Machine Learning and Classification Engine" (MLCE). The MLE consists of a Central Processing Unit which could be an ARM processor, several input devices such as camera feeds through CCTV cameras, voice lines from microphones, data buses from other devices or computers, or from remote wireless lines etc,

all these are not shown but may exist. There are on board storage disks and on board memory devices which contain the machine code of the classification programs. Basically the MLCE function in two modes, that is it will be in two states (i) the Learning State and (ii) the decision making state. While it is in te learning state it acquires input from the input devices such as camera feeds or microphones. During the learning phase the CPU will be running the classification process to learn the input images (say faces for recognition), all the inputs are learnt and classified by the classification process. The output of this process is a Neural Network Architecture (the details for each layer of processing elements, their weights etc), which is stored and then used for the Decision making process to recognize the images that arrive in real time. In the Decision making process the input is analysed by the CPU which then uses the Neural Network Architecture to recognize each of the input data which arrive in real time (say faces of people) image, and then a decision is made either to sound an alarm/ accept entry of the said person or simply record the data.

As said before, for another type of application say speaker or voice recognition the input will be through a microphone and the data will be previously recorded voice samples. Even here embedded systems which are stand alone can be built. For very large classification problems involving Big Data e.g. from the internet or from Youtube will mean the deployment of our algorithms on numerous parallel machines or putting the application on a Cloud. Many players like Google and Microsoft, and Face book are using trying to further such research in Deep Learning. However, the principle will remain the same and the algorithms and techniques evolved from this present invention can be used to advantage to handle the huge amounts of data by working on many computers and CPUs.

Details of Algorithms

I. Algorithm of Partition of Points in D-Dimension by Hyperplanes

a. Setting up the Tasks at hand and necessary definitions

We give the necessary definitions and briefly prove the fundamental algorithm by which planes which part a given set of points could be found.

Our task is to determine the equations of the planes that can part the points in such a way that every point is parted from another by at least one plane.

Assumption: It is assumed that we have specified a (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

$$I + \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n = 0 \qquad (1)$$

is the equation to some n- dimensional plane then a point P whose coordinates are $(p_1, p_2; \ldots, p_n)$ will be said to be on the positive side if $1 + \alpha_1 p_1 + \alpha_2 p_2 + \ldots + \alpha_n p_n > 0$ and in the negative side if $1 + \alpha_1 p_1 + \alpha_2 p_2 + \ldots + \alpha_n p_n < 0$

Each component of the Orientation Vector of a point P in X space, is on the positive side or negative side of each plane.(The 'Orientation Vector', is defined in the next section, below.)

a.1 Definitions and Terminology:

Orientation Vector: Suppose we have a point P in n-dimension space (also called X-space) and suppose we are given 3-planes. we define a Orientation Vector as a Hamming vector whose components precisely specify on which side the point P lies with respect to the 3 planes. Example: if the point P lies on the positive side of plane 1, negative side of plane 2 and positive side of plane 3 , then we define the Orientation Vector associated with P as Ov(P) and define Ov(P) (1; 1; +1). Similarly a point Q which lies on the negative side of plane 1, negative side of plane 2 and positive side of plane 3 will have an Orientation Vector Ov(Q) (1; 1; 1) . Points, whose Orientation Vector di□er from another by at least one component can be said to be parted by at least one plane.
Q Space or Hamming space: We will also call the space spanned by the Orientation Vector as Hamming Space or Q space. Since each point P in X-space has an Orientation (Hamming) Vector associated with it, we can imagine that all the points in X space are mapped to a point in Hamming space. Of course this mapping is many to one, but the fact to notice is the following: Let P be some point in X-Space, then all points, R, in X space which are not parted from P by planes will all have the same Orientation Vector as P i.e. Ov(P) = Ov(R) and we describe this by saying: "Points P and R belong to the same 'quadrant"' this statement is certainly true for the "images" of P and R in Q-Space, but we will loosely use this terminology for the points in X-space and say P and R are in the same "quadrant", what we really mean is that P and R are points in X-Space and that they are not parted by planes; Sometimes we will use the term neighbors and say P and R are neighbors. The point to remember is that P and R will be neighbors if and only if Ov(P) = Ov(R). Therefore if one wishes to find out if two points A and B are parted in X-Space which contains planes, all one needs to do is to compare their Orientation Vectors: Ov(A) with Ov(B).

A Saturated Plane: We consider a plane in n dimension space to be "saturated" if it has already been constrained to pass through n points and hence cannot be adjusted to pass through a new point, the coefficients of such a plane are completely determinable. Eg. A plane in 3 dimension gets saturated if it is made to pass through three points.

We will suppose we are given all the data containing N_f points, and that all all their coordinates in n dimension space are known to us. Our task is then to find the planes numbering q_f , that can part all these points from one another and also to find all the coefficients of the equations which define each one of these planes.

a.2 Input Output Requirements

Input to the Algorithm: We are given a set G containing a total of N_f points in n dimensional space: That is we know the coordinates of all these points. These points may also belong to different classes, so they may be given a label or 'color'. However, since we are isolating all points from one another regardless of their class, the labels do not matter here, but will be useful in certain applications.

Without, too much of a loss of generality we will assume that the coordinates of all the N points in G are rational numbers, i.e. they are either integers or fractions. We will see that this assumptions makes the coefficients of all the q planes that part the N in G, also rational numbers.

Output Items of the Algorithm: The equations of the planes which divide each point in such a way that every point is parted from another. And an integer q_f which is equal to the total number of planes required. The equations of the q^{th} plane will be of the form:

$$1 + \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n = 0 \qquad (q = 1; 2; ...; q_f) \quad (2)$$

Where all the α_{qj} are coefficients of the plane all these coe□cients are determined by the algorithm and is a known number nearly equal to unity.

Storage Requirements for running the algorithm: A set Set S which will contain a list of N points and q planes. S will contain the Orientation Vectors of all the points in S and the identification numbers (labels) for the planes, along with the coe□cients which define each plane. S contains an array V for storing Orientation Vectors; and 'Counter': An integer number.

In all stages of the algorithm S, will contain only those points, (N in number) which are completely

parted from one another by the q planes contained in S. This condition of seprability of the N points in S is never violated even though N and q change as the algorithm progresses. When the Algorithm ends S will contain the necessary output.

Another Set T which contains points and their Orientation Vectors and an array M :"List of Midpoints" which are the coordinates of the mid points of certain line segments. At each stage of the algorithm this set T, will contain pairs of points one not in S and the other in S. We are sure that each such point has only one such neighbor (in S), because the points in S are always part from one another and hence two points in the same quadrant cannot both belong to S. Each such point stored in T will also store the coordinate of the point which is the midpoint of the line joining it to its neighbor and the line segment with its neighbor. The points in T are the candidates waiting to be put in S but they first have to be parted from all the points in S. We shall see that as soon as T has a collection of n pairs of such points, then a new plane is drawn which parts all these points so that they become eligible to be incorporated in S. (Later on, we will permit a given point in S to have two or even three neighbors in T, these too will be candidates to be put in S, but for a first reading of this algorithm, it is simpler to assume that a point in S can have at most one neighbor in T.)

We also need another set D, this is the 'Dust-Bin' set, it removes all 'accumulation' points when detected from set G, (as explained later this is done so that the algorithm does not go into an infinite loop; which may happen if G contains a sequence of points converging to an 'accumulation' point or 'limit' point - we assume that such points do not exist in G).

b. Steps of the Algorithm

Step 1: Initially collect a small number of initial points numbering N_0 and choose a set of q_0 planes that part each one of these N_0 points and put these planes in S. The coe□cients defining the equations of these q_0 planes should be determined (or randomly chosen but ensuring that they part the N_0), call $N = N_0$ and $q = q_0$. Store all Orientation Vectors of points in S in array V. In addition we need a set T which will contain points which cannot become immediate members of S but are prospective members and will become members of S eventually. Set T will contain points which are neighbors of a point which already is a member of S. To make things simple we will assume that at the start, all the q_0 planes are saturated. (We do not want to adjust any of these planes)[1]. Put Counter = 0.

Step 2: If no more points in G go to step 7, else: Randomly choose a new point from G, which could be a candidate point to be put in S, from the remaining points not in S, go to Step 3.

Step 3: Check if this new point is in a new 'quadrant', this involves finding its Orientation Vector and comparing with those in S. If the point is in a new quadrant, put this point in S and store its Orientation Vector in V, put N = N + 1 go to Step 2, if not, it means it has a neighbor in its quadrant. Go to step 4.

Step 4: (You will come here only if the current point has a neighbor in S. Notation and procedure for this Step: We keep count of the number of members of S which have first neighbors. Such points are called a_i, its first neighbor will be called b_i, if a_i has a second neighbour this will be called c_i and the third will be called d_i .) First: Find the 'distance' of this new point from its neighbor, call

this If $<_{th}$ then remove the new point from G itself and put it in set D, and Go To Step 2; else Put this new point in Set T. Check if this new point is in a quadrant which has already a pair in T, if not, put Counter = Counter +1 define i = Counter, call the point with which this new point is a neighbor as a_i. However, if this new point already is a neighbor of some point a(j) already in S and if b(j)&c(j) exist call this new point d(j), if only b(j) exists call the new point c(j) go to Step 2. However, if b_j , c_j as well as d_j exists this means the new point will become a 4th neighbor of a(j), a situation which we do not permit, so put back the new point in G and go to step 2. If c_i and if d_i do not exist, it means this point a_i has only the present point as neighbor.

Calculate the midpoint of the "segment"(a_i; b_i), if not already calculated, and call it m_i, add the

[1] To avoid 'starting troubles', choose q_0 and N_0 s.t. $n < 2^{q0}$ and $N_0 > n$

coordinates m_i, in a "List of Midpoints".(Note Counter only keeps track of the first neighbors of the a^0 s.)

If Counter = n go to Step 5, if less than n, go to Step 2.

Step 5: Since Counter = n, this means you have collected n points in T, each of which have to be parted from their neighbors. However we, first re-check if all the n points collected in T are in di□erent quadrants (This step is only cautionary, and not necessary if step 4 has been done properly),if say b_i and b_k in T are in the same quadrant, then mark one of them say b_k as c_i, if the latter doesn't exist, else as d_i (If both c_i and d_i both exist there is no place for b_k put it back in G , restore the count of points in G as well as restore in either case Counter = Counter -1, check similarly for other b^0 s and then go to Step 2.)[2]

Now in this step we will part the n pairs collected in T by introducing a new plane which passes through all the n mid points, whose coordinates are available in "List of Midpoints" in the array M: "List of Midpoints" and determine the coefficients of the new plane, by solving then constraint eqs. to pass it through the n midpoints[3] ; call this plane as plane number q + 1 and put q = q + 1, and include this new plane in S along with all the n points labeled b_j; j = 1; 2::; n) and their coordinates, put N = N + n,

Step 6: Now check the neighbors of all the a_i, i = 1; 2; ::; Counter, in T, if only c_i exists then it gets promoted to first neighbor. But there is a slight complication after a_i and b_i are parted c_i can be a neighbor either of a_i or b_i so if it is the former call c_i as b_i else call b_i as a_i and c_i as b_i the same kind of investigation needs to be done for d_i because it can now belong to the old a_i quadrant or belong to the new b_i quadrant, in either case the d_i gets promoted as second neighbor and it will be called c_i. Calculate the new mid point m_i for the just created segment (a_i; b_i) and store.

After finishing this task, the number, r of second neighbors that were promoted as first neighbors will be known, after drawing the plane q +1, then call Counter = r.

Update V, the Orientation Vectors of points now stored in S wrt. to this new plane (their dimensions become q + 1); Clear the data in M: "List of Midpoints" of all points T you have transferred to S and remove data, of these k points that have been transferred to S, from set T (most of the time k = n) and go to Step 2.

Step7: You will come here only if no more points are left in G and $N_f = 0$ and Counter is not yet = n). If Counter = 0, Stop else introduce a new plane passing through the midpoints of segments collected so far, (T will contain as many points as the value of Counter), since Counter < n, some of the coe□cients of this plane can be randomly chosen; (take action like step 6 to check all these new points are in their own quadrants), update V, the Orientation Vectors of points now stored in S wrt to this new plane (their dimensions become q + 1); call this plane q = q + 1, Counter = 0, Clear the data in M: "List of Midpoints" then if now $N_f = 0$ i.e. G is empty, Stop; else go back to Step 2. At the end of section 6.2, we mention that the case when Counter < n and G is empty which is one of the possibilities in Step 7; the situation can be dealt with in an easier manner.

END OF THE ALGORITHM

[2] We have permitted three points in T to belong to the same quadrant but not 4 or more. This is just to simplify the algorithm, anyway, such events are extremely rare. Since, for large n, if there are q planes there are 2^q quadrants, hence the chances of two points randomly chosen to be in the same quadrant is $O(1=2^q)$. We permit max. no of neighbors,3, just to ensure that even in the freakiest conditions the algorithm comes to a halt successfully. Of course the other possibility is the presence of points of 'accumulation', however this does not happen when points represent integers.

[3]The coefficients of the plane containing the n midpoints, can be found by using Gauss elimination, Gram-Schmidt evaluation or by using the QR algorithm, the last is more useful just in case the n mid points fall in a plane of n-1 dimension or less, then the rank of the coe□cient matrix will become less than n. This will happen rarely, and even if it does, it only means we can accommodate another point (or points) in T with a neighbor in S, thus making the rank n. In such a case all the points in T (even though they are now more than n) can be parted by a single plane - the algorithm can proceed.

The algorithm works swiftly most of the time, since in our case all the point are discrete and there are no limit points or there is no "accumulation points" in G the algorithm will always come to a halt successfully. The general case when the input data is say prepared by another program , then the above conditions need to be checked by the data preparation program in order to ensure the correct halting of the algorithm

c. Discussion

The following crucial concepts will immediately clarify the steps of the whole algorithm:: In n dimension space, if we are given n line segments; $(a_1; b_1)$; $(a_2; b_2)$; $(a_3; b_3)$; ...;$(a_n; b_n)$,

then a single plane passing through the mid points of each segment will part the n points a_1, a_2, a_3;...,a_n from the n points b_1, b_2, b_3, ..., b_n that is the a^0 s and the b^0 s will lie on opposite sides of the plane. But this does not ensure that the a^0 s are part from each other and the b^0 s are part from each other. In order to ensure this we have imposed the condition that each of the a^0 s belong to Set S and are already part from each other and have all have different Orientation Vectors (in the space of q planes which are in S), Similarly by imposing the condition that each b , (each of which belong to T), has only one neighbor in S, (for the moment ignore the second and third neighboursc^0 s and d^0 s) we are making sure that each b_j is part from other b^0 s though, each b_j is not part from its neighbor in S namely a_j. Thus each pair of points $(a_j; b_j)$ have the same Orientation Vector. Now if the (q + 1)st plane is drawn then it will ensure that all the 2n points i.e. the set of all the a^0 s and the set of all the b^0 s are not only part but are parted from each other. We can then include plane q + 1 and the n points viz. the b^0 s in S. This intuitive result is mathematically easily demonstrable:

PROOF: Since we have included the (q+1)st plane, all the Orientation Vectors have gained one more dimension and have q + 1 dimensions, the last, (q + 1)st component of the Orientation Vector of point a_j will differ from the last component of the Orientation Vector of its ex-neighbor b_j; because they are now on either sides of plane (q + 1). Thus proving that all the 2n points are now part. **QED**

The rare cases of three points belonging to the same quadrant (viz b_i; c_i; d_i) is permitted but not four, we do this only to avoid unnecessary return of points to G, once randomly chosen from G. However, one can simplify the algorithm if one just returns the second point to G and then choose a new point at random.[4]

d. PROOF OF ALGORITHM:

At the start S only contains points which are separable from each other, because that is the way they have been selected. Since $q = q_0$ have been selected initially each of the Orientation Vectors of the points in S, are q dimension vectors. They are all different.

Now till the time we come to step 5 we are just collecting points which are separable and putting them S or in case they have a neighbor in S then we are putting the points in T. This can go on till there are n points in T and we arrive at Step 5.

At this point there are n points in T and N points in S. Firstly, there can never be two points in S which are neighbors to a single point in T. This is because every point in S has a different Orientation (Hamming) Vectors so they belong to di□erent quadrants in Hamming space and one point in T cannot belong to two quadrants.

There are now only two possibilities (i) Each point in T has one distinct neighbor in S or (ii) there are two or three points in T which have the same neighbor in S. We will consider the first possibility (i): Since by choice, every point in T must have one neighbor in S, they all belong to different quadrants, since there are n points which can be parted by the q + 1 st plane after implementing Step 5. Then in the new q + 1 space all the points (the old points in S and the n new points in T) are all

[4]This simplification of the algorithm will work most of the time work. But there may be some freakish cases when (say) the last few points left in G all belong to the same quadrant! This would mean introducing a few unnecessary number of planes in the end.

separable form one another and hence all the n points in T which were first neighbors the b_i's can now be included in S, we must add the q + 1 st component to each Orientation Vector in S to make them into Orientation Vector wrt to the q + 1 st plane.

Now coming to the second possibility, if there are two or three points in T which have the same neighbor in S then after introducing the new (q + 1) plane only one of the points in T namely the one which is the first neighbor of a_i say b_i can be transferred to S and the other points c_i and d_i need to be kept in T with one of the points a_i or b_i as its neighbor and the newly calculated midpoint.(Because the q+1 plane has parted the a_i and b_i they are on opposite sides of plane q + 1, so we have to now determine on which sides c_i and d_i are; the seemingly complicated arguments are only to ensure that we make the right decision about c_i and d_i). However, the situation (ii) is rare in high dimension n space, since we are choosing points in random, and can best be avoided by not choosing two points in T which are neighbors to the same point in S (this means ejecting the second point from T and putting it back among N_f in G, or by keeping it in T and use it, later, only after you have introduced a new plane or even after a few new planes.) In brief, the situation (ii) can be avoided though it just causes difficulties in programming.

So we see after step 5 and Step 6 we will have an enhanced set S with more points and one more plane and all of which are parted by these q + 1 planes. So we can call (q + 1) as q and then re-do steps 2 to 6 till all points N_f in G are exhausted. Then S will contain all the points and all the planes which part them.

Step 7 will happen in the end when all the points are exhausted but there are less than n points in T and these must be parted. The logic of Step 7 is similar to step 6.

It may be mentioned here that Step 7 can be completely avoided.

If we find that there are say less than n points in T. Let us say r is a number such Counter + r = n. All we have to do is choose r random points (these should not have been in G) each of which has as a neighbor in its quadrant, one of the points which was drawn from G but now in S. Then put each of these r points along with its neighbor as a pair in T. Since you have already Counter number of pairs in T, with the addition of these r pairs, we have n pairs. Now since Counter = n, we can choose the last plane separating all the n pairs.[5]

So we see at the end of the algorithm S will contain all the points along with the equations of the q planes that part them, which are the needed outputs.

QED[6]

In the above algorithm it is assumed that every new plane does not 'split' some other point (pass through). However if this happens (it is a very rare event), to some point, we will have to shift the 'mid points' of the segments slightly to one side (because to cut a segment into two parts it is not necessary that the plane passes exactly though its mid point) so that the plane passes to one side of the o□ending point,[7] and then proceed with the algorithm.

---

[5] These are just 'End-Game', tricks that can make programming simpler.

[6] It may be noted that this algorithm is non iterative, and every point is mostly "looked at" only once.

[7] Instead of shifting a point already in S, we could shift all the mid points by a small distance in the direction of the new normal and recalculate the plane, thus moving it to one side of the point upon which it was formerly incident. We do this so the algorithm won't fail.

Another neat trick which, pure mathematicians who are disciples of Cantor, may admire, is to replace the constant 1 by τ (this 1 occurs in every equation which defines the q planes in Eqs., (1), (2),..., (q) below); and choose τ to be a transcendental number which is almost unit, say choose $\tau = \pi_{20}$, where π_{20} is the value of π correct to 20 decimal places. If now, we are careful to choose the coordinates of all the points to be rational numbers (which is easily arranged if we denote each coordinate up to a fixed number of decimal places say 15), then we can be assured that the plane can never pass through a point whose coordinates are represented by rational numbers, this is especially true since all the coefficients α_{ij} are rational, being obtained by solving a finite set of equations whose coefficients are rational (because all the mid points are means of two rational numbers). This replacement of 1 by τ (which is very close to unity) should be done only after we have obtained all the α_{ij}, we thus doubly ensure that all points which represent rational numbers will never lie exactly on any of the q planes. Strange as this may seem we have, succeeded in separating all points whose coordinates are represented by rational numbers in n-dimensional space by using q planes, which act as boundaries which contain points

e. Conditions for the Algorithm to Stop

Before completing the proof of the algorithm, it better to briefly discuss the conditions when the algorithm will stop and when it will not and to make sure that it will always stop.

In order to better understand what is happening: We will pretend that the first part of the If...then statement in Step 4 is removed. ie the 'distance' is not calculated and all new points which arrive at this step are never removed and put in D even if it is close to its neighbor. (This analysis will then reveal the need of set D).

It is then clear from the algorithm that as we randomly pick points from G each point picked finds a new quadrant or finds itself in T. If it finds itself in a new quadrant there is no problem: It is put in S and the algorithm goes to Step 2, and a new point is picked from G. But if it lands itself in T, then it is paired with another point which is already in S. Let us say that such a point is Q and it has been paired with P which is in S and we pair (P,Q). Now the algorithm will not attempt to transfer Q to S from T until it has collected n such pairs. Then the algorithm draws a plane through the mid points of n pairs. Everything is fine if all the pairs are in di□erent quadrants the new plane parts all the 2n points and the n points which were collected in T (and now parted) are transferred to S. The algorithm ensures that in the set of n pairs if there are at most 3 (r = 3) points which are neighbors of a point in S (in the same quadrant), for example b(i); c(i); d(i) can be neighbor of a(i) which is in S, then things would be fine.

Things will not be fine if every new point that is subsequently picked will all belong to the same quadrant as Q! This strange phenomena will happen when:

1. All the Q's which are chosen all belong to a sequence which has an "accumulation" point. That is the remaining points in G which are left after all the others are parted are only those points which tend to be close to some unknown accumulation point (or limit point) close to P. This will happen since we are treating X-Space as real space and all the coordinates (x_1; x_2; x_3; :::; x_n) are either rational numbers (or even real numbers which are uncountable as opposed to integers which are countable). And then we are trying to part an infinity of points or uncountable points in a sequence by a countable (or finite) set of planes an impossibility!

2. When all the points in G are not distinct and there are repetitions in data.

They way to get rid of both the situations is to take a bit of care in preparing the input data. In real life situations the input set G may be points generated by another computer program. The following are obvious suggestions:

1. We have characterized all the components of the n space as real numbers, that is all the x^0 s in the coordinate array of point P : (x_1; x_2; x_3; :::; x_n), are considered real. It is better that one of the components is an unique integer number. For example, in the case of the medical data which we describe in section 3.2 (below) it is better to label each point with an unique integer number ie every patient P is given a unique integer label L_P , and we put this value as (say) the 10^{th} component of the coordinate of P. That is we put $x_{10} = L_P$. If this happens no two points will ever have the same coordinate, (because they di□er in their 10^{th} component) and the number of points will always be an integer number, and be kept finite hence countable. They will always be separable because the minimum Euclidean (or Manhattan) distance would be 1.

2. For every prospective candidate for T, we check its 'distance' from it neighbor and if $< _{th}$, we remove the point from G and put it into the 'Dust-Bin' set D. We use a small threshold value for $_{th}$, we need only calculate the 'Manhattan distance' to perform the task. This process removes all

---

whose coordinates are not all rational but have algebraic and transcendental components!

'accumulation' points from the data and thus explains the need of the first part of If...then statement in Step 4.

3. Ensure that there are no repetitions in input data i.e. the points in G are all distinct.

Once it is ensured that the input data, set G, is prepared as above, the algorithm will run smoothly.

f. Calculation of coefficients of planes

As we have seen, as the algorithm progresses and points are being transferred from G to S and new pairs are being created, one will have to draw a new plane as soon as n pairs are collected. Let these n segments be called:

$(a_1; b_1); (a_2; b_2);...(a_j; b_j); ...; (a_n; b_n)$,

and let the mid points of each segment such as $(a_j; b_j)$ be denoted as m_j thus we have the list of midpoints for all the n segments: m_{j1}

$(m_1, m_2, ..., m_j,..., m_n)$, where the coordinate of the j^{th} mid point is denoted as $x_1 = m_{j1}$; $x_2 = m_{j2}$; ...; $x_i = m_{ji}$; ...; $x_n = m_{jn}$

Now we require that the $q^0 = q + 1$ plane:

$$1 + \alpha q^0_1 X_1 + \alpha q^0_2 X_2 {}_q0 + \ldots\ldots + \alpha q^0_n X_n = 0 \qquad (3)$$

should pass through all the above n mid points, we thus have the n constraint equations, from (j = 1,2,3,...n) :

$$1 + \alpha q^0_1 m_{j1} + \alpha q^0_2 m_{j2} + \ldots\ldots + \alpha q^0_n m_{jn} = 0 \qquad (4)$$

The above Eqs.(4)represent n linear equations which can be solved by suit-able numerical techniques, eg. Gaussian elimination, to obtain the n unknown coefficients $\alpha_q 0_i$, for (i = 1,2,...n).

It may be noted that since all the m_j are midpoints of the segments $(a_j; b_j)$ and the coordinates of (a_j and b_j) are rational numbers the coordinates of m_j are also rational numbers, this makes all the coefficients ${}_q0_i$ rational numbers. This observation has profound implications as we have described in the footnote 9 at end of page 12.

In the following we work out a simple example on the working of the algorithm in great detail. It may be consulted by any-one who needs to quickly get an hands on experience of the algorithm and program it.

g. Calculation of Computational Complexity:

The computational complexity of the algorithm is easily determined.

Given: N_f : Total number of points; q_f : Total number of planes used; n; Dimension of X-Space.

I. To determine the coe□cients of each plane by Gaussian elimination we require 2/3 $O(n^3)$ multiplications, and 2/3$O(n^3)$ additions therefore for q_f planes we have 2/3q_f $O(n^3)$ multiplications and 2/3 q_f $O(n^3)$ additions. 2. To compute the Orientation Vector of each point with respect to q_f planes require per point per plane: n linear evaluations, each consisting of n multiplications and n additions. Therefore Total: N_f q_f n multiplications and N_f q_f n additions. 3. When every time a new point is put into S, it's Orientation vector is compared with the Orientation Vector of the points in S. Therefore there are N_f^2 Hamming Vector comparisons. But just to see if two Hamming Vectors of dimension q_f are not equal it does not require us to check all the q_f components, the moment the rth component differs the Hamming vectors are declared unequal. So we may assume that 99 percent of the time only about 8 components are checked on the average. So the number of bit comparisons made are: 8:N_f^2 bit comparisons.

4. We must calculate the Manhattan distance of every candidate member which arrives at T from its neighbour (Step 4). For every plane there will be n such points arriving at T. Each Manhattan distance calculation requires n subtractions and n additions, therefore for q planes we have q.2n additions. We

ignore the additions required to detect accumulation points which we had put in the Dust-Bin set D, assuming they are small in number.

Now we need to guess as to how many planes are require to part N_f points, we use the estimate of Boland and Urrutia (1995), Ref [2], also see [1],and assume that $q_f \log_2(N)$, hence we come to the conclusion that the order of computations involved by the algorithm are :

$N_f \log_2(N_f)n + 2/3 \log_2(N_f)O(n^3)$ multiplications;
$N_f \log_2(N_f)n + 2/3 \log_2(N_f)O(n^3) + \log_2(N_f).2n$ additions and
$8N_f^2$ bit comparisons.

II ALGORITHM FOR OBTAINING A NEURAL CLASSIFIER BY USING PARTITIONING OF INDIVIDUAL POINTS.

In this method the process is as follows:

(i) All the data which consists of points in d-dimensional sample space is divided into two sets (a) a set of points, say N in number which will be considered as the "Training 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 coordinates 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 partitioned from the other by at least one plane, let q be the number of planes that separate these points.

(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 than its orientation vector Ov(P) indicates the "Orientation" of the point P w.r.t. each of the q planes See the Figure above.

After the points are partitioned and the planes that have performed the separation are known then the method to process the classification of new points is easily devised. We now 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.

Data retrieval

We will now briefly describe 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. We will now show that 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 we will assume that each point N represents one d- 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 d pixels and represent a photograph of a person. Now we wish to store many such samples, data points, in such a manner that classification and retrieval becomes easy. The idea is simple: after we had run the algorithm which separates each of the N data points (samples) by q number of planes, we will 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 we have the Orientation Vector for each point stored in S. We can use this in-formation to (i) store and classify the data in such a manner that it is possible to (ii) retrieve it easily. What we mean is that after the storage is done and if we are given a fresh (approximate) data of a person whose data is stored in the storage receptacle in the repository, it possible to use this new (approximate) data to retrieve the stored data (image). In other words if we are given a new sample point, we can retrieve another example 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 we can retrieve another data point of a person L (stored in the data

base), which is closest to the present sample point P.
We show that if this new sample point P in X-space, is close to some other sample point L (say) stored in the receptacle then we 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. Obviously 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 we 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 Fig. 7.

In the Fig, 7, E1 represent the equation to the first plane stored in S; We assume that the equation to this plane is given by

$$1 + \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n = 0 \qquad (5)$$

we then define

$$z_1 = 1 + \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \qquad (6)$$

If we use the Sgn (Sign) function 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 our _Storage Plan_ for each point L in is to 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.

3. Method 2: Classification by using a Cluster Discovery algorithm in addition to the partitioning of points 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 $O(N^2 \cdot \log_2(N))$: 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.

Figure 8 shows 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 rowi = [di1;di2;:::;din]. Here dij is the distance from point 'i' to point 'j'. After sorting, rowi = [di1;di4;di7;di9;di2]. 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 rowi, count the first 'm' number of consecutive points, whose class is same as the class of point 'i'. Consider the rowi = [di1;di4;di7;di9;di2]. 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 rowi as four.

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

(ii). Consider rowi = [di1;di4;di7]. Create a new bubble by taking pointi as bubble center and add pointj(j=1, 4, 7) to the bubble, when pointj 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 sometimes 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 centres 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:).

In order to check if the bubbles are separable by planes, as shown in Figure 9, 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 described method planes are obtained which separate bubbles after they have been discovered by the "Bubble Discovery" algorithm. As shown in the methods described in literature, 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.

4.0 Final Step of Classification of feature space after the Bubble Algorithm and Partitioning of the pure bubbles.

After the feature space has been covered by the cluster discovery and bubble algorithm, we obtain a figure shown as below, wherein, each pure bubble consists of only those points belong ing to one class.(The figure is simplified for illustrations purposes). In the Figure 10, we show the bubbles in blue, green and red colours indicating the three classes. In the above figure, the bubbles have been portioned by planes, and hence by following the methods of the previous invention (May 2015)we can immediately write down the Neural Architecture that classifies the above configuration, the neural Architecture is shown below and the weights of each processing elements are known because the equations of the portioning planes are known.

Since there are five partitioning planes as shown in Fig. 11, the architecture contains five processing elements in the first layer, and since there are eight pure bubbles there are eight processing elements in the second layer and since there are only three classes there are three processing elements in the last layer. The above architecture completely solves the given classification problem. Note it has been obtained in a completely deterministic and non-iterative manner.

8. EXAMPLE PROBLEMS

In this section we just describe the results of the algorithms in two illustrative examples. The first example (a), describes the application of the computer program that was developed for the portioning of points, the results are discussed briefly. The second example (b), is the application of the entire portioning and classification process on the classical IRIS problem. We show how the computer program using the algorithms described in this invention automatically determines in a non-iterative manner the neural architecture to solve the IRIS problem.

a) Partitioning of Random points in a high dimension cube.

A computer program that solves a non-trivial problem by using the algorithm employing the method of Orientation Vectors, described in provisional patent disclosure No: 2669/CHE/2015, dated 28/5/2015 and detailed in Para 7.1 (above), has been written.

This program first generated 2000 random points inside a 15 dimensioned unit cube and then applies the invention to show that only 22 hyper planes can separate each of the 2000 points. The coefficients of the 22 hyperplanes are obtained by the program and the Orientation Vectors for each of the 2000 points are obtained and shown to be unique. Thus explicitly demonstrating that the 22 hyperplanes actually partition all the 2000 points in such a manner that every point is separated from another and if one chooses any pair of points there will be at least one hyper plane separating them. (The time taken to run the program was almost instantaneous in a Toshiba laptop).

The computer program was then successfully used for a larger sized problem involving 50,000 points which were similarly randomly generated this time in a 25 dimension space they could be separated by only 27 planes. The time taken was approximately 6 min 25 secs on a desktop.

Thus we have an explicit demonstration by using the computer program that the algorithm that has been proved in the above paper, is not only correct but also a practical algorithm which is efficient and can be put to practical use.

b) Example of Application of above methods on the IRIS FLOWER Classification Problem.

We applied the above method for the same IRIS Flower data. And did as follows:

This classical problem, which has been a bench mark test for many researchers involved in pattern recognition was first introduced by R. A. Fisher (1936,[13]) and Edgar Anderson [14]. It consists of 4-dimensional data of 3 classes of IRIS flowers. The problem is to consider some of them as "train data" and then to train our classification engine to correctly classify the rest of the data. The data as available at https://archive.ics.uci.edu/ml/machine-learning-databases/iris is utilised.

We had applied the above processes described in our invention for partitioning of points and then bubbles to obtain the architecture of the neural network shown below:

FIG 12

The IRIS data set as available at the above cited website is a data set which was first collected by Edgar Anderson and has data regarding three species of flowers: Iris setosa, Iris virginica and Iris

versicolor data. Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres, thus the data is 4-dimensional. Based on the data the task is to classify any other sample flower into one of the three classes if just 4 measurements of a sample is given (i.e. one can find the flower-species of the given sample). The data consists of 4 numbers each for 150 flowers (50 samples for each class), therefore the data is four dimensional n = 4 and our task is to separate every point from the other by using suitable 4-dimensional hyperplanes.

This method was applied on the IRIS data set in the following manner:

(i) The data was separated into two sets (a) the training set and (b) the test set. The test set was obtained from the IRIS data by choosing every 5th flower, so there were 120 flowers left in the training set and 29 in test set (it was discovered that two flowers had the same dimensions so one of them had to be removed, leaving 149 samples).

(ii) A training set containing only 120 samples were used to build the classification - retrieval system.

Training Process:

The classification system was built in the following steps:

Step 1: Use the separation algorithm to separate the 120 samples each of which can be represented by a point in 4-dimension space, by planes.

To start the algorithm initially one can choose in set S, 4 planes whose equations were of the form: - $[\bar{x}]^{1} x_j + 1 = 0$ (j = 1 to 4) where x is the average value of x_j of the data set.

Step 2: Run the algorithm which transfers points from initial set G to set S such that each point is in its own 'quadrant' drawing planes whenever necessary. It was found that 24 planes separate all the 120 points.
Step 3: By the means of the coefficients of the 24 planes the classification system shown in the figure can be built, this system basically finds the OV's of any incoming 4-dimensional point P. In our case there will be 24 processing elements $E1; E2; :::::; E24$ in the first layer because there are 24 planes.

Testing Process:

After this for the testing process each of the 29 points were presented and the OV of each point was found, then the OV's of each point was compared to the OV's of the other 120 points and the closest match was found. By this manner:

It was found that: (i) Nine points (point numbers 5, 10, 25, 30, 35, 40, 70, 85 and 110) from the test data had the same OV as some point in the train data (ii) another nine points (point numbers 15, 20, 45, 50, 75,100, 105, 140 and 145) from the test data had their OV's differing from their closest match with the train data by only one component (iii) Eight points (point numbers 55, 60, 65, 80, 95, 120, 125 and 130) from the test data had their OV's differing from their closest match with the train data by only two components (iv) Two points (point numbers 90 and 115) from the test data had their OV's differing from their closest match with the train data by only three components (v) Only one point (point number 135) is wrongly classified as class1 and class2.

Hence, this simple method of classification worked very well for this IRIS data correctly classifying 28 out of 29 test points.

Then the Bubble Discovery algorithm (Method 2) was applied for the same IRIS Flower data set [14-15] (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. 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.

9. Additional Example Problems: Digit Recognition and Face Recognition

We now briefly describe our results of applying Method 1 and Method 2 to two other problems, see ,namely the Digit recognition problem (MNIST database) and the Face Recognition Problem (the Yale Data Base).

Application of Method 1: this algorithm was recently applied on the digit recognition data set MNIST and the Face recognition data set viz. the Extended Yale dataset. In MNIST, the original dimensions of each sample was 28 X 28. It was reduced to 6X6 dimensions. It was found that it took 65 planes to separate the 60,000 training points and the time taken (on a Lap Top) was 4.8 minutes and the time taken to classify the 10,000 test points was 3.9 minutes. The classification accuracy was 92.2 %. In Yale dataset, the original dimensions of the each sample was 30 X 30. To separate approx. 11,200 train points (faces), 25 planes were needed and the time taken was 1.36 minutes and the time taken to classify the test points was 5 minutes. approximately 81.3% of test points were correctly classified out of 3744 test points, this is a very good result because no dark images were removed and it involved many poses with more than 64 lightning conditions. Just the raw data was fed and no pre-processing of the images was done and no face-dependent segmentation was used.

Application of Method 2: this algorithm was recently applied on the digit recognition data set MNIST. 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. 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. 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 the literature cited above. This Method 2 was also applied on the Face recognition Extended Yale Database, the results were around 81%.

It is pointed out that the time taken by these methods were only minutes whereas in the conventional neural network can take place for several hours and even days, because they are iterative, and they have the further disadvantage of the lack of knowledge of the number of processing elements and number of layers for each problem and these have to be guessed at to start with a neural architecture. Whereas our algorithm actually determines the neural architecture.

Conclusions from above Examples:

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

CLAIMS:-

What is claimed is:

1. A system and method with Artificial Intelligence (AI), Machine Learning comprising:
partitioning given data points, building neural classifier and data retrieval system for classification of new data points; classify large amounts of data, building artificial neural network specific to the data space; use of invented algorithm to separate individual data points in large n-dimensional space from one another in polynomial time with hyper planes; a method capable of using this information to perform classification, recognition and decision making at optimum level and to classify large amounts of data, classification of diseases from medical data.

2. The system and method of claim 1, with Artificial Intelligence (AI), and Machine Learning comprising: partitioning of said data points, to train available data, classify the said patterns using number of processing elements and its weights, forming automatic machine learning and classification system, and use of method of orientation vector to classify said data in non-iterative manner.

3. The system and method of claim 1, further comprising:
partitioning of all data points, building up of a Neural Classifier and Data Retrieval system done for the classification of new data points.

4. The system and method of claim 1, further comprising:
ensured availability of sufficient amount of trained data and then use separation algorithm to completely determine the architecture of an artificial neural network (ANN), along with the number of processing elements and its weights, to classify the said patterns; which then forms the basis of an automatic machine learning/classification system, the architecture of which performs the classification of the trained data to almost 100% and the classification of the unseen (test) data at a very high success rate.

5. The system and method of claim 1, further comprising:
the Orientation Vector method leading to newer inventive steps to suitably and efficiently classify the given data points in a non-iterative manner and capable of learning new data without starting ab-initio.

6. The system and method of claim 1, further comprising:
usage of the distance information to form the "Pure Bubbles", which contain only train-points belonging to the same class, and the utilisation of the planes obtained in the process of partitioning of points, which results in the cluster discovery algorithm wherein each bubble is separated by planes, the same being used as planes defining a Neural Network to determine and classify new test data.

7. The system and method of claim 1, further comprising:
use of separation of planes algorithm to reduce the dimension of the input data, using this non-iterative and deterministic method which is qualitatively as well as much more efficient than the trial-and-error method of finding an auto-encoder neural network used in Deep Learning; the reduced dimension input

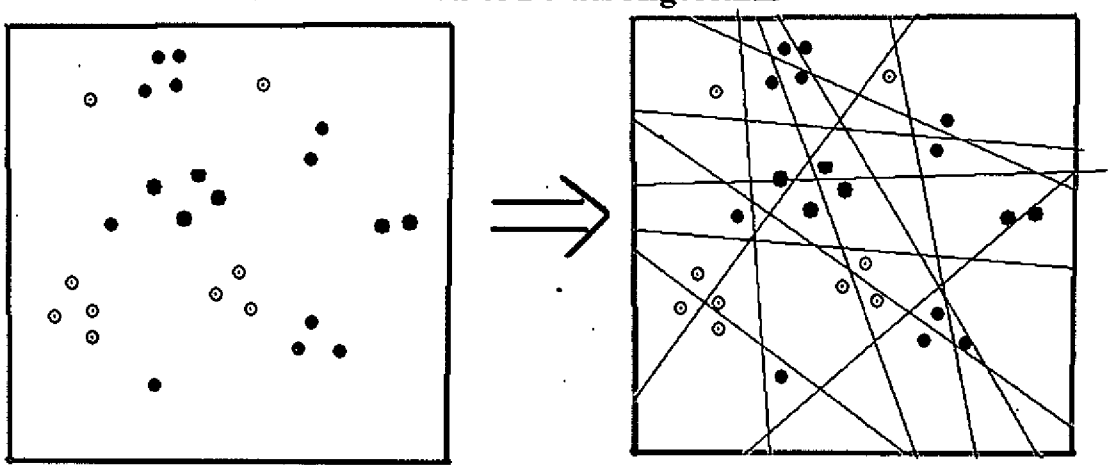
so obtained is used for classification problems which involve large dimension inputs of images or other forms of data.

8. The system and method of claim 1, further comprising:
the non-iterative and deterministic system and method used to determine the underlying Neural Network in terms of number of processing elements and number of layers of processing elements.

9. The system and method of claim 1, further comprising:
achieving water-tight proofs which guarantee success whenever and wherever patterns are recognizable by given features and the algorithms, and being deterministic and non-iterative, wherein the algorithm determines the neural architecture by itself.

10. The system and method of claim 1, further comprising:
resulting in high accuracy results while solving several particular problems such as (i) digit recognition MNIST, (ii) Face-recognition viz. Extended Yale data base, and (iii) Alphabet recognition (iv) Liver Disease Detection (v) Detection and Classification of diseases in Trees; in comparison with similar ones being attempted by the developing deep learning methods.

The Partition of Points Algorithm



ALL Points are partitioned by Hyperplanes regardless of which class the points belong to.

In the above figure 25 points are partitioned by 11 planes

This algorithm is NEW, fast and non iteratiive and always ends successfully.

FIG 1. The Partition of Points Algorithm

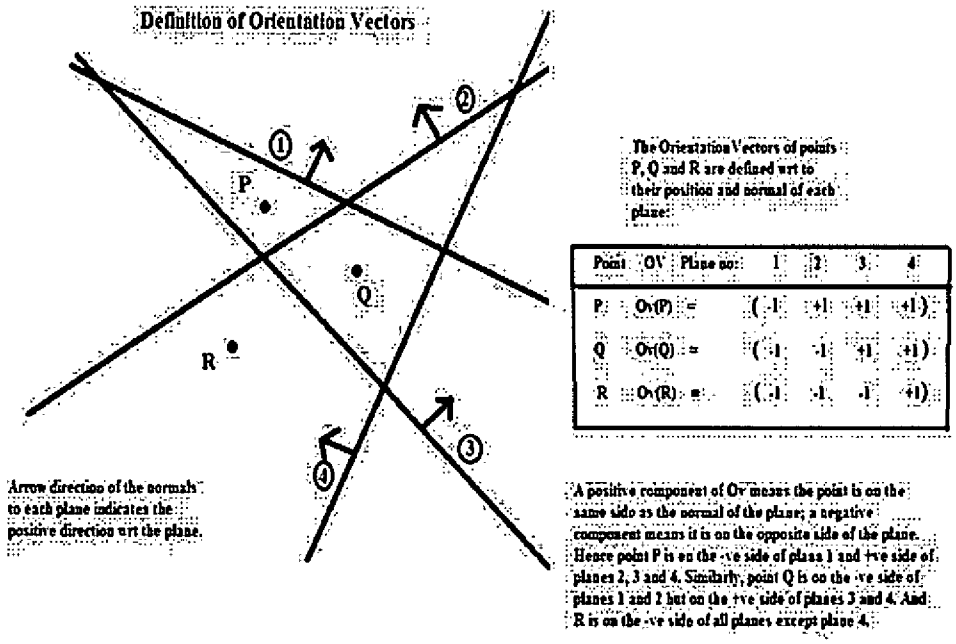


FIG 2. The Orientation Vector Method

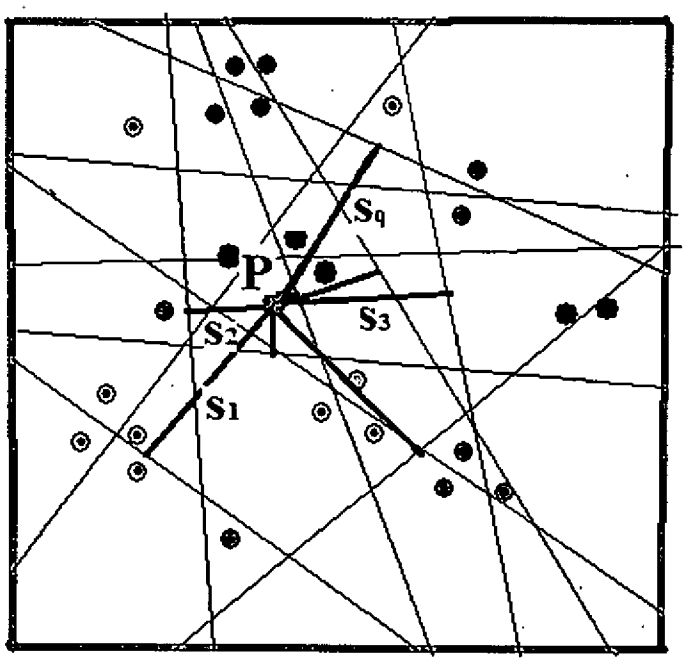


FIG 3. S – 'Coordinate' system

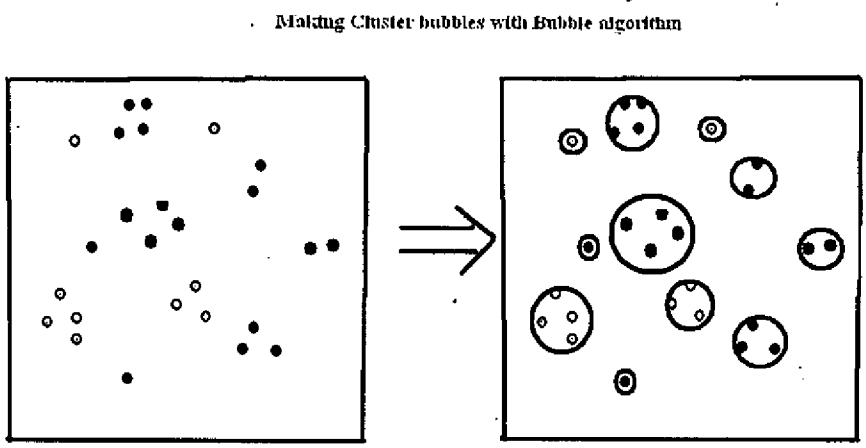
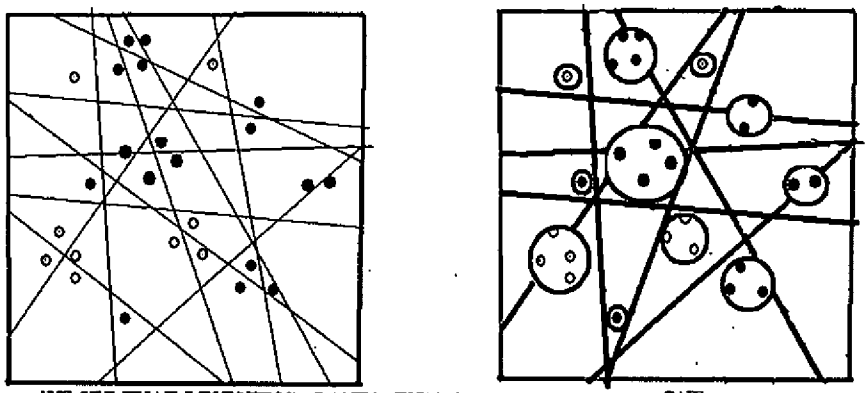


FIG 4. Making the Cluster bubbles with Bubble algorithm

BRIEF DESCRIPTION OF FIGURES (DRAWINGS)

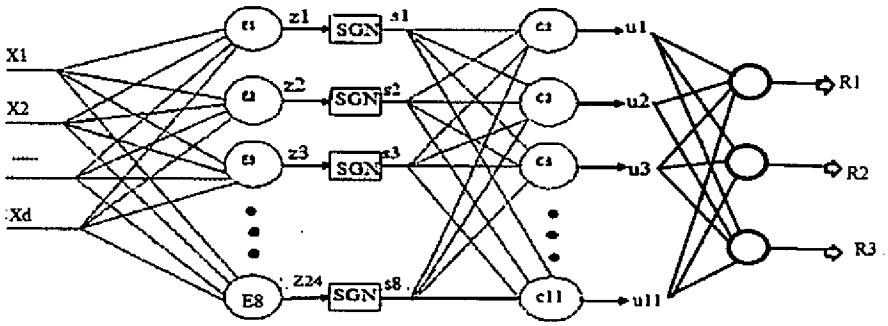
1. The Partition of Points Algorithm.
2 Definition of orientation Vectors (OV).
3. The S- Coordinate System, used for dimension reduction
4. Making Cluster Bubbles
5. Partitioning of Bubble clusters by Planes
6. Neural Network Architecture for Fig 5

SEPARATION OF BUBBLE CLUSTER BY PLANES



WE SEE THAT 7 EXISTING PLANES (THICK BLACK LINES) AND ONE ADDITIONAL PLANE (THICK RED LINE) SEPARATE ALL CLUSTERS. THE "BUBBLE OV's" FOR EACH BUBBLE ARE COMPUTED JUST LIKE POINT OV's EXCEPT THAT IF A PLANE PASSES THROUGH THAT PARTICUALAR BUBBLE THAT COMPONENT IS MADE 0 (PLANE DISABLED) FOR THAT BUBBLE). FOR CLASSIFICATION A DOT PRODUCT IS TAKEN OF A TEST POINT OV WITH ALL BUBBLE OVS.

FIG 5 Seperation of Bubble Cluster by Planes



This is the Neural Network Architecture, for the above 25 point (assumed d-dimensional), 8 plane, 11 cluster (11 bubbles) and three class problem (classes red, blue and black). Note the NN Architecture is completely determined with weights by the Algorithms in a unalterative manner and there is no need of Back-propagation. The weights in the 1st layer are nothing but the coefficient of the planes (known), the 2nd and 3rd layer weights are determined by the Bubble OVs and the class of each bubble (all known).

FIG 6A Neural Network Architecture

HARWARE IMPLEMENTATION OF MACHINE LEARNING AND CLASSIFICATION ENGINE (MLCE)

COMMUNICATION LINES ARE SHOWN AS ARROWS, GREEN LINES ARE USED WHILE TRAINING/UPDATES AND BLACK ARROWS FOR DECISIONS IN REAL TIME.

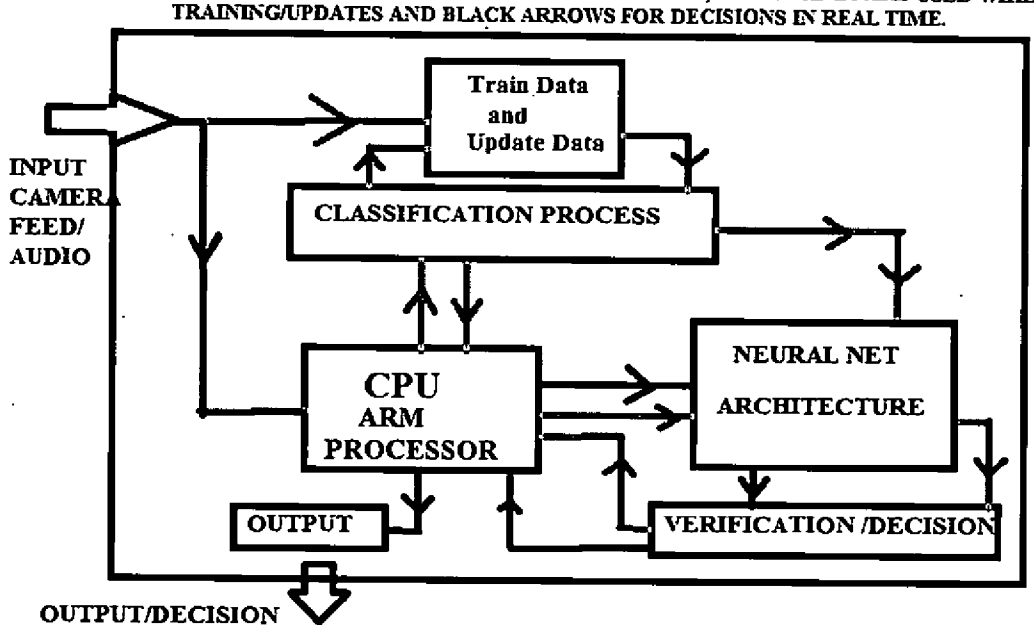


FIG 6B. Classification Engine

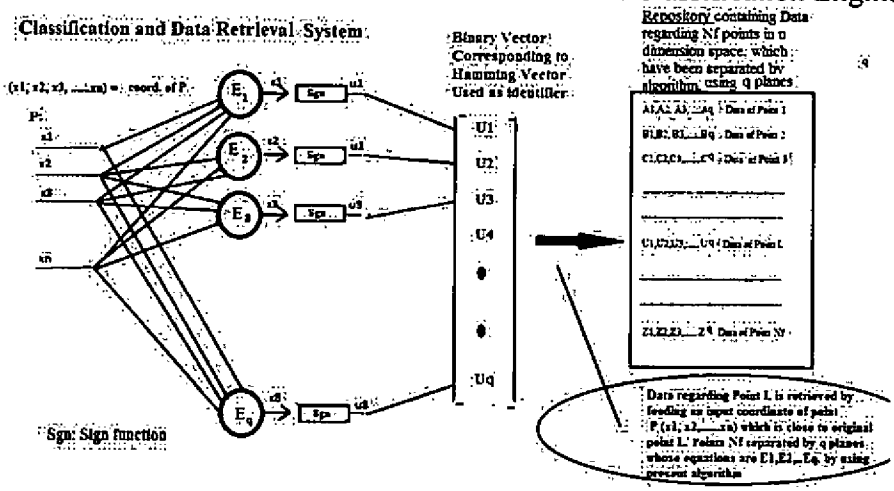


FIG 7. Classification and data retrieval system

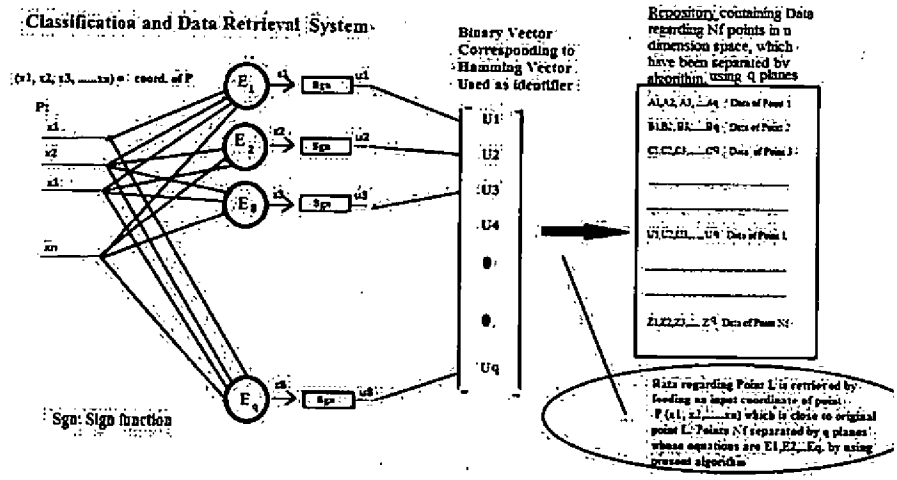


FIG 8. Classification and data retrieval system

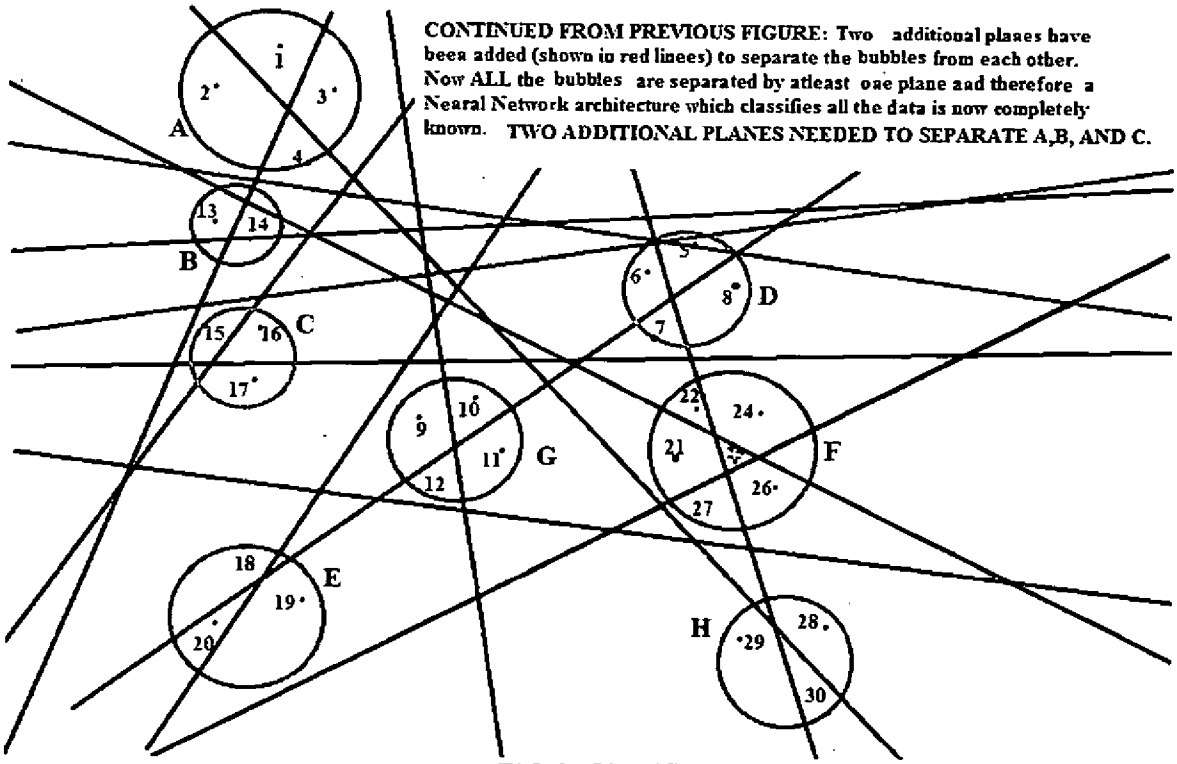


FIG 9. Classification and data retrieval system

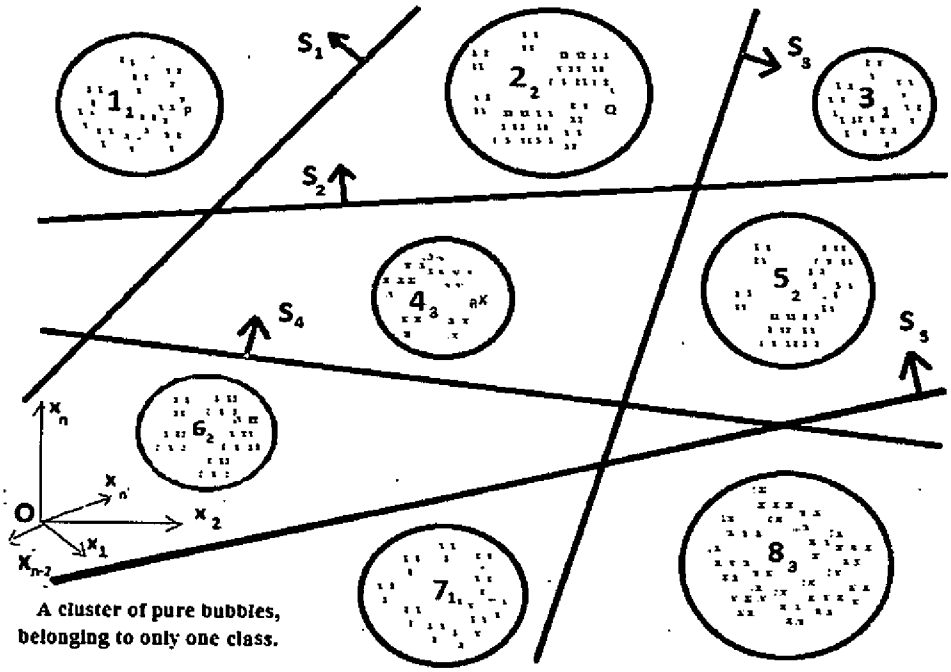


FIG 10. Cluster of Pure Bubbles

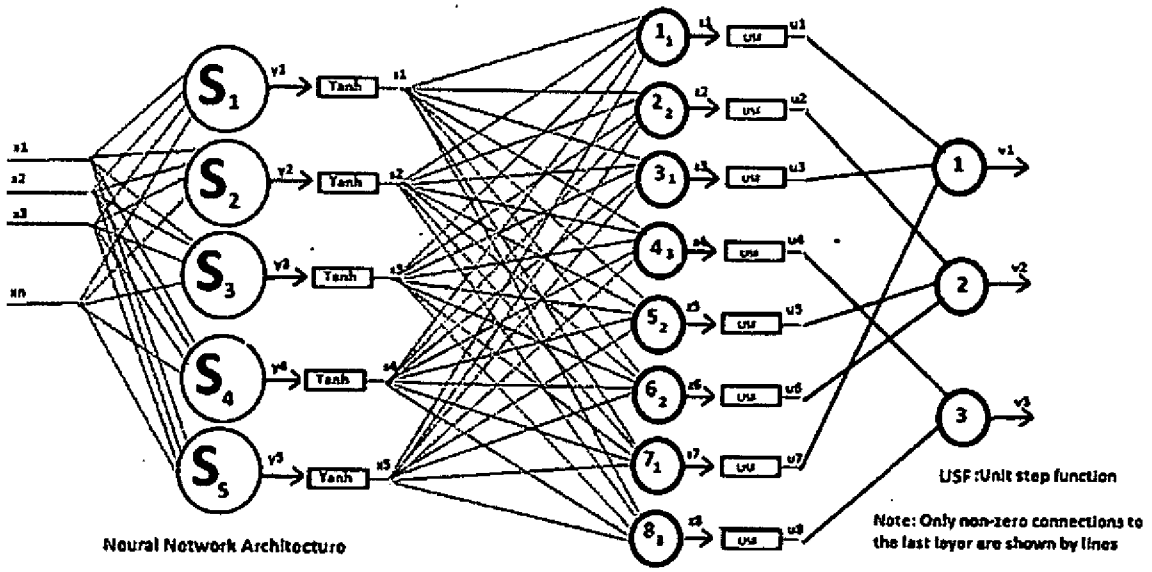


FIG 11. Neural Network Architecture

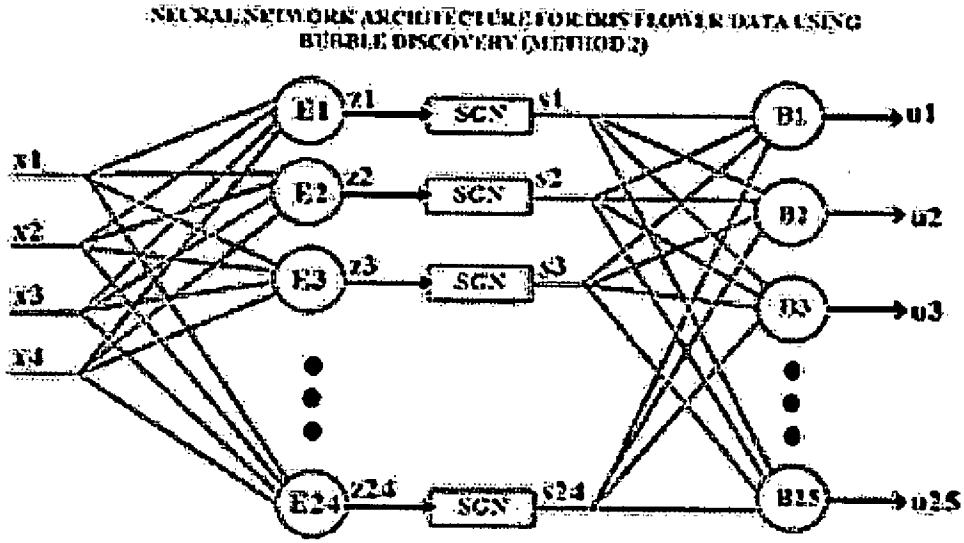


FIG 12. Bubble discovery method

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IN2017/000096

A. CLASSIFICATION OF SUBJECT MATTER
G06N3/00,G06N99/00,G06F15/18,G06F17/30 Version=2017.01

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06N, G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Databases: Patseer, IPO Internal Database
Keywords: Neural Network, Architecture, Classification, Learning

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	IN-CHE-2015-02669A(KUMAR ESWARAN) 02-12-2016 (02 December 2016) Abstract, Pages 3-9, 12-15	1-10
A	US20060074824 A1 (AGENCY SCIENCE TECHNOLOGY AND RESEARCH) 06-04-2006 (06 April 2006) Whole Document	1-10
A	US7792770 B1 (LOUISIANA TECH RESEARCH CORP) 07-09-2010 (07 September 2010) Whole Document	1-10

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19-07-2017	19-07-2017
Name and mailing address of the ISA/ Indian Patent Office Plot No.32, Sector 14,Dwarka,New Delhi-110075 Facsimile No.	Authorized officer Shri Ram Kaunaujia Telephone No. +91-1125300200

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/IN2017/000096

Citation	Pub.Date	Family	Pub.Date
US 20060074824 A1	06-04-2006	EP 1550074 A1	06-07-2005
		CN 1689027 A	26-10-2005
		JP 2005538437 A	15-12-2005
		AU 2002330830 A1	11-03-2004
		WO 2004019264 A1	04-03-2004