From:

Tattersall G.D and Limb P.R. *Visualisation techniques for data mining*. BTTJ Vol.12, No.4, pp23-31, October 1994.

<u>**Introduction**</u>

Data mining is a term which has become popular to describe a number of techniques for the exploration and exploitation of data. In particular, a large part of data mining involves the visualisation of data and subsequent utilisation of machine learning techniques for data classification. This paper describes some techniques for data visualisation which enable the user to enhance understanding of the structure and properties of data. Such insight into the nature of a data set is very useful when deciding what type of pre-processing should be applied prior to automatic classification [1],[2], or prior to application of machine learning techniques for further analysis and exploration. The latter techniques are covered in other papers in this issue.

BT collects and stores large quantities of data from a variety of sources. These large data sets typically describe different states of a system and are difficult to interpret because there is no obvious way of abstracting and presenting data features in a meaningful way for a human observer. Examples of such data range from credit status of customers of a company, to acoustic data generated from speech. The use of computers has made it relatively easy to collect and store such data, but this has not been accompanied by a corresponding development in methods of displaying and interpreting the data.

A particular entry in a database typically consists of the values of a number of measurements by which the data is described. For example, a single entry in a database relating to BT customers might provide values for the number of lines rented by the customer, the last bill value, and the customer's title. The number of lines, bill value and title are called attributes and may have numerical or symbolic values such as £347 or "Mr". It is common practice to imagine individual data examples, described by N attribute values, as points in an N-dimensional space. The co-ordinates of the point in the space are the attribute values. This way of imagining data leads to the idea of visualising the distribution and relationship between data examples by "taking a walk through the N-space". In general many more than three attributes are used to describe a data example and it is difficult conceive of such a dimensional space.

Visualisation of the data examples in a high dimensional space can be made much easier by forming a two-dimensional map of the N-dimensional space. This is not a new idea and an algorithm developed by Sammon [3] in 1967 has been widely used. However, this algorithm is severely limited in its application to large data sets because its computational complexity rises as the square of the number of data points to be mapped.

The work described here is a novel approach to mapping which has been developed by the authors. This approach, which is based upon a neural network, produces a map very rapidly compared to other more conventional techniques, and its speed of operation allows the user to interact with the data whilst looking at the map. For example, the distribution of the data in the map can be observed as the user interactively modifies various parameters such as the way in which the distance between data examples is measured in the high dimensional space. Interactive mapping enables the user to discover natural clusters of examples in a data set, and also find features which can be used in automatic classification of the data.

The paper also shows how different types of two dimensional data map can be generated to emphasise different aspects of the data. For example, it is possible to generate maps which simply reflect the point to point distances between data examples in their N-space, or which emphasise the separation between clusters of data points in the N-space. Perhaps most importantly, the map can be generated to show the degree of separability of the data points in terms of their assigned class. The latter map can be used to assess how well the data could be automatically classified as well as highlighting those specific examples which are liable to be mis-classified.

All of these interactive data mapping facilities, which have been made possible by the new algorithm, have been embodied in a data visualisation system called the HTM tool which enables the user to interact with the data map displayed on the screen via the computer's mouse. This tool is being further developed to make it part of a suite of AVS data mining tools.

Before describing the HTM interactive visualisation tool in detail, we review the operation of the Sammon mapping algorithm in order to highlight some of the problems of mapping N-dimensional data into a two dimensional map.

The Sammon Map

The Sammon Mapping Algorithm

Successful interaction with the data visualisation tool requires fast mapping of multi-dimensional data into a two dimensional map for display with as little distortion as possible of the apparent distances between examples in the data set. A well established approach to such mapping was first proposed by Sammon [3], but unfortunately the practical use of the Sammon Mapping algorithm is limited because its computational complexity rises as the square of the number of data points to be mapped. Typically, if the algorithm is run on a PC, a map of 20 data points can be generated in a few seconds, but would take hours for a thousand data points. The long time to generate a map makes it impossible to use the algorithm in an interactive manner and effectively mine the data.

However, the Sammon algorithm exhibits many of the important aspects of data mapping and is described in detail here to illustrate general problems and show the state of the art prior to the development of the mapping tool

The basis of the Sammon Mapping Algorithm technique is to map the set of data points in N-dimensional space to an equal number of corresponding points in a 2-dimensional space. The relative positions of the 2-dimensional points are iteratively modified until their relative distances mirror as closely as possible the *relative* distances between pairs of points in the N-dimensional space.

The iterative adjustment of the positions of the points in 2-dimensional space is done using gradient descent minimisation of a mapping error, E, which is defined as the average of the squared difference between the distance of each pair of points in the N-dimensional space and the corresponding pair of points in the 2-dimensional space.

Let the i^{th} N-dimensional vector in the set of patterns to be visualised be denoted as X_i and the corresponding 2-dimensional vector be Y_i. Assuming that there are a total of M examples in the data set, there are M points in the N-dimensional space and 2-dimensional space. The squared distances between the i^{th} and j^{th} vectors in the N and 2-dimensional spaces are $d_p(X_i,X_j)$ and $d_m(Y_i,Y_j)$ respectively where $x_{i,k}$ and $y_{i,k}$ are the values along the k^{th} dimension of X_i and Y_i respectively:

$$d_p(X_i,X_j)=\sqrt[2]{\sum_{k=1}^{N}(x_{ik}-x_{jk})^2} \qquad (1)$$

$$d_m(Y_i,Y_j)=\sqrt[2]{\sum_{k=1}^{2}(y_{ik}-y_{jk})^2} \qquad (2)$$

A measure of the total difference between the distances of pairs of points in the N-dimensional and 2-dimensional spaces is therefore:

$$E=\sum_{i=1}^{M}\sum_{j=1}^{M}\left\{d_p(X_i,X_j)-d_m(Y_i,Y_j)\right\}^2 \tag{3}$$

Usually the initial values of Y_k are set randomly and are iteratively modified using the steepest descent algorithm defined in equation (4) in which k_s is a small constant which determines the learning rate. The gradient term is evaluated by differentiation of equation (3):

$$y_{ik}^{n+1}=y_{ik}^{n}-k_s\cdot\frac{\partial E}{\partial y_{ik}^{n}} \tag{4}$$

Problems of The Sammon Mapping Algorithm

Computational Complexity: Any mapping algorithm work sufficiently fast that a user can make changes interactively to the data whilst observing the map. This means that the mapping must converge within a fraction of a second after any changes have been made. The computational complexity of the standard Sammon Algorithm makes this impossible unless the number of examples being displayed in the map is very small.

Placing New Data in The Map: If a new data example is to be added to an existing Sammon Map, the entire data set with the new data example needs to be re-mapped. It is difficult to simply place the new example in an existing map.

Map Storage: Maps of data need to be stored for future reference in exactly the same way as normal geographical maps. The Sammon Map requires storage of the co-ordinates of every single point in the map and the map cannot be represented in a compact form.

Topological Order: A map is said to be topologically ordered if the position ordering of points in the map reflects their position ordering in the original space. This kind of topological order can be *global* , in which case the ordering of all points is correct, or *local*, in which case only the ordering of points which are close together is correct.

The standard Sammon Map attempts to reflect global topological order and local topological order of the space in which the data examples are described. This leads to a conflict in the mapping unless the data examples actually lie on a plane in their space. Experiments have indicated that the Sammon Map generally minimises the conflict by effectively projecting the data points onto a suitably oriented plane rather than the surface of a complex manifold. Projection onto a plane is probably a useful solution, but it can be achieved much more simply by techniques other than Sammon Mapping.

The Hidden Target MLP Mapping Algorithm

Basis of The New Mapping Algorithm

The new mapping algorithm described in this paper is based on the multi-layer perceptron (MLP) [4]. The MLP is iteratively trained using a modified form of error backpropagation called the Hidden Target Mapping Algorithm (HTM). This algorithm effectively projects the examples of N-dimensional data onto a curved surface or plane within the N-dimensional space, and then displays the "flattened out" surface with the projected data points as a 2-dimensional map. The surface is automatically positioned to maximise the accuracy of the map and the curviness of the surface is controlled by selecting an appropriate MLP architecture.

How The Algorithm Solves The Computation, New data, and Map Storage Problems

The key to the algorithm's reduced computational load is that it places sensible constraints on the form of the mapping generated and in exchange requires much less computation than other techniques such as the Sammon Map. The degree of mapping constraint is determined by the complexity of the MLP and can range from forcing all points to lie on a plane, to placing points on a highly complex curved surface. In the former case very little computation is required to form a map whilst in the latter, considerable computation is needed. In comparison, the Sammon Mapping algorithm places no constraints on its mapping and as a result requires still more computation to form a map.

The algorithm also solves some of the other problems raised in connection with the Sammon Mapping algorithm: in particular the problems of *new data* and *map storage*. If the HTM algorithm has been already been used to generate a data map and it is desired to place a new data example in the existing map, the new data example is simply applied as a pattern to the input of the MLP. The output of the MLP will be the co-ordinates of the position of the new example in the existing map.

The HTM mapping algorithm also solves the potential problem of map storage associated with the Sammon Mapping. The entire HTM Map can be re-generated from the weight values of the MLP used to form the mapping. A typical MLP used in this application might use 250 connection weights, each represented by a four byte floating point number. Thus only 1kbyte of memory is required to store the map, regardless of the number of points in the data base which are to be mapped.

Structure of The Hidden Target Mapping (HTM) MLP

The MLP used to form the map has N input units to which the N-dimensional data example is applied, a number of hidden units, and two output units whose output values are the co-ordinates to which the input is mapped in the two dimensional map. The output units are linear and the hidden units use sigmoidal activation functions.

The function of the MLP is to develop a linear or non-linear transform of the input N-dimensional pattern to an output 2-dimensional pattern which can be plotted as a point in a 2-dimensional map. The form of the transform is determined by the weight values of the MLP which are iteratively adjusted to minimise an error measure for the mapping. This is shown in the next section.

The HTM Training Algorithm

It is usual to train an MLP using error backpropagation [4] in which the derivative of the error between the MLP's outputs and specified target values are evaluated and used to adjust the weights of the MLP using the least mean squares (LMS) algorithm. However, in the mapping application, no explicit output target values are available, and a different form of error must be evaluated.

The mapping error measure chosen for the HTM algorithm is essentially the same as that used in the Sammon Mapping. The distance, $d_p(X_i,X_j)$, between each pair of data examples, X_i and X_j in the N-dimensional pattern space is compared with the distance, $d_m(Y_i,Y_j)$, between the corresponding pair of points Y_i and Y_j in the map space. The weights of the MLP are iteratively adjusted to minimise the square of the difference between $d_p(X_i,X_j)$ and $d_m(Y_i,Y_j)$ over all the examples in the data set.

For practical reasons, the MLP weights are updated after the evaluation of the mapping error for a randomly selected pair of data examples instead of after the mean square mapping error for the entire data set has been evaluated. This is equivalent to "per example MLP training" as opposed to "epoch training" of MLPs. The algorithm for the iterative modification of the weights therefore proceeds as follows:

1) Initialise all the weights in the MLP with small random values.

2) Randomly select two N-dimensional pattern examples, X_i and X_j from the data set.

3) Record the responses of the MLP, Y_i and Y_j to the inputs X_i and X_j.

4) Evaluate the distances $d_p(X_i,X_j)$, and $d_m(Y_i,Y_j)$,

5) Evaluate the the squared error, $e^2(i,j)$, between $d_p(X_i,X_j)$, and $d_m(Y_i,Y_j)$ and then evaluate the derivitives of the squared error with respect to each weight in the MLP.

6) Use the gradient descent algorithm to update each weight value.

7) GoTo (2)

Evaluating The Error Derivatives

The squared error derivatives used in the gradient descent adaptation of the weights of the MLP are evaluated by differentiation of the expression for the squared error between the N-space distance, $d_p(X_i,X_j)$, and the distance between the corresponding two dimensional vector outputs from the MLP, $d_m(Y_i,Y_j)$, with respect to the ouput from the MLP. i.e.

$$e^2(i,j) = \left(d_m(Y_i,Y_j) - d_p(X_i,X_j)\right)^2 \tag{5}$$

and:

$$\frac{\partial e^2(i,j)}{\partial y_{ik}} = \frac{2\left\{d_p(X_i,X_j) - d_m(Y_i,Y_j)\right\}.(y_{ik} - y_{jk})}{d_m(Y_i,Y_j)} \tag{6}$$

It is then simple to evaluate the required derivatives of the squared error with respect to the weight values of the MLP by using the derivative given in equation (7) in conjunction with the standard backpropagation rule [4] in which the term $\delta_{1,k}$ is set equal to the derivative of the squared error with respect to the kth output of the MLP.

$$\delta_{1k} = \frac{\partial e^2(i,j)}{\partial y_{ik}} \tag{7}$$

The derivative of the squared error with respect to a weight, $w_{n,k,s}$ which connects the output of the kth neuron in the (n+1)th layer to the sth unit in the nth layer is then given by the recursively evaluating expressions (8) and (9) in which S_n is the number of neurons in layer n, and the output of the kth neuron in layer n in response to input X_i , is denoted by $o_{i,n,k}$.

$$\delta_{n+1,k} = o_{i,n+1,k}(1 - o_{i,n+1,k}).\sum_{s=1}^{S_n} w_{n,k,s}.\delta_{n,s} \tag{8}$$

$$\frac{\partial e^2(i,j)}{\partial w_{n,k,s}} = o_{i,n+1,k}.\delta_{n,s} \tag{9}$$

The weights of the MLP can then be updated iteratively using gradient descent as shown in equation (10) in which p is the iteration index.

$$\omega^{p+1}_{n,k,s} = \omega^{p}_{n,k,s} - k_s.\frac{\partial e^2(i,j)}{\partial \omega_{n,k,s}} \tag{10}$$

Forcing Local Topological Order

The error measure, (equation 5), which is minimised by the HTM algorithm makes no distinction between the mapping error for pairs of points which are distant in the pattern space and pairs of points which are close together. This is precisely the problem mentioned in connection with the standard Sammon Map. Inevitably the mapping will attempt to simultaneously reflect the global and local topological order of the space in which the data examples are described, and this may lead to a conflict in the mapping unless the data examples actually lie on a plane.

The problem is ameliorated by the presence of the term, $d_m(Y_i,Y_j)$, in the denominator of the expression for the error derivative given in equation 6. If two points are close together in the map, $d_m(Y_i,Y_j)$ is very small and tends to amplify the value of the error derivative. Thus the mapping becomes much more sensitive to errors in mapping points which are close together than far apart and the map will tend to reflect local topological order at the expense of global order. This property has been exploited in the HTM algorithm by including a *locality control* which can be modified while the map is being displayed. The locality control is a numerical factor which is used to control the influence of the $d_m(Y_i,Y_j)$ term in the denominator of equation 6. The locality control, k_L is introduced into the modified expression for the error derivative given below. The range of k_L is zero to one. When it has zero value, the denominator of equation 18 becomes independent of $d_m(Y_i,Y_j)$ and global order will tend to be reflected in the mapping. When k_L is one, the $d_m(Y_i,Y_j)$ term will have full effect and local order will tend to be reflected in the mapping at the expense of global order.

$$\frac{\partial e^2(i,j)}{\partial y_{i,k}} = \frac{2\{d_p(X_i,X_j) - d_m(Y_i,Y_j)\}.(y_{i,k} - y_{j,k})}{d_m(Y_i,Y_j).k_L + 1 - k_L} \tag{11}$$

Practical Considerations In The Use Of The HTM Algorithm

Absolute Values of Outputs: An MLP trained using the HTM algorithm is only provided with an error related to the relative positions of the points in the map space, and not their absolute values. This can lead the MLP to produce a set of points, {Y}, which have massive absolute value even though their relative values are correct. This may result in numerical overflow. A solution is to modify the expression (7) for δ_{ik} so that it includes a term which causes the values of the outputs to tend towards zero. This is achieved by including a weighted absolute target value for $y_{i,k}$ of zero as shown in equation (12) in which the tendency for the outputs to adapt to zero value is controlled by the constant k. Typically k has a value of 0.05.

$$\delta_{1k}^{'} = \delta_{1k} - k.\, y_{ik} \qquad (12)$$

Choice of Single versus Multi Layer HTM Algorithm: It is frequently found that the N-dimensional data can be mapped quite accurately onto a plane rather than a curved surface. To do this, the MLP is made to have just a single layer of linear units. This has a number of beneficial effects: the speed of learning is high, convergence to a global minimum is certain, and a map is immediately visible even before weight adaptation has started. The latter property arises because any set of weights in a linear perceptron will define a surface onto which the data can be projected.

Using a single layer MLP will not always allow a satisfactory map to be generated. If it is found that the normalised mean square mapping error does not converge to a small value when using a single layer MLP, it will be necessary to introduce a second layer of units. A practical approach is to start by using a small number of hidden units in the second layer. If this fails to provide a satisfactorily low mean square mapping error, the number of hidden units should be increased.

Pattern Space Distance Metrics and The HTM Mapping System

The Effect of Changing The Distance Metric

The HTM mapping is iteratively modified until the relative values of the 2-dimensional output of the HTM's MLP mirror as closely as possible the relative distances between pairs of points in the N-dimensional space. It makes sense to plot points in the map space using simple Euclidean distance (equation 1) because humans intuitively understand this type of distance. However, there is no reason why some other metric should not be used in the N-dimensional pattern space. If this is done, the Euclidean distances in the map space become proportional to distances in the N-space according to the other chosen metric. There are numerous pattern space distance metrics which could be used in conjunction with the mapping, and some examples which have been built into the prototype visualisation tool are presented in the following section.

Examples of Pattern Space Distance Metrics

Vari-Power Metric: This metric is a generalisation of the Euclidean distance metric. The distance between two vectors is found from the summation of the distances along each dimension, raised to the power p, which can be any real positive value. If p is made large then the distance between two vectors tends to be dominated by the largest distance along any particular dimension. Conversely, using a value of p which is much less than one tends to make the distances measured along each dimension have equal significance, regardless of their actual value. The metric is defined as follows:

$$D(X_i, X_j) = \sqrt[p]{\frac{1}{N}\sum_{k=1}^{N}\left|x_{ik} - x_{jk}\right|^p} \quad (13)$$

City Block Metric: The City Block metric is a special case of the vari-power metric in which the power, p, is equal to one. When applied to binary data, this metric effectively returns the Hamming distance. The metric is defined as follows:

$$D(X_i, X_j) = \frac{1}{N}\sum_{k=1}^{N}\left|x_{ik} - x_{jk}\right| \quad (14)$$

Normalised Dot Product Metric: The normalised dot product metric returns a similarity, $S(X_i,X_j)$, value which is proportional to the cosine of the angle between a pair of vectors, independent of their magnitudes.

$$S(X_i, X_j) = \frac{1}{\left|X_i\right|}\cdot\frac{1}{\left|X_j\right|}\sum_{k=1}^{N} x_{ik}.x_{jk} \quad (15)$$

However, for the purposes of generating maps, a distance is required rather than a similarity value, and the following conversion, shown in equation 16, has been chosen which returns a distance value in the range 0-1.

$$D(X_i, X_j) = \frac{1}{2}(1 - S(X_i, X_j)) \quad (16)$$

Attribute Weighting in The Map

In addition to selecting a particular type of pattern space distance metric, it is possible to weight the importance of each attribute such that it contributes more or less to the calculated pattern space distances. The weighting of each attribute can be changed interactively whilst the map is being displayed and this facility enables the user to examine the significance of each attribute.

Typically, the examination would be done by setting all attribute weightings to zero except the weighting of the attribute being tested. If the resulting map shows class specific clusters it is clear that the particular attribute is important. Conversely, if increasing the weight of a particular attribute does not cause greater class or cluster separation in the map, it indicates that the attribute is not useful.

The attribute weighting can also be used to control the viewing aspect of the map. This happens because the HTM mapping error is minimised by primarily projecting the values of the heavily weighted attributes onto the map. Thus changing the attribute weighting function changes the "angle" of the data projection onto the map surface.

<u>Product Features and The HTM Mapping</u>

The Need For Product Features

Different categories of data are often defined by the co-occurrence of pairs of attributes. One approach to numerically indicating the co-occurrence of a pair of attributes is by forming a new feature which is the product of the values of the chosen pair. If the product has a positive value, it indicates that the attributes simultaneously have the same polarity and vice versa. (The product must be evaluated after the chosen attributes have been normalised by setting their mean values to zero.) The product features from each pattern in the data set can be evaluated and then displayed in a map using the HTM algorithm. This may reveal structure in the data which is not apparent when the original attributes are mapped.

An interesting and potentially useful case arises when a product feature is formed by squaring a single attribute value. Regions in the original attribute space which are enclosed by a convex boundary will be mapped to a region which can be separated by a single hyper-plane in the feature space.

The arithmetic process of forming product features can be applied equally well to both real valued and binary symbolic data. In the latter case it is equivalent to forming a new feature which is the exclusive-OR of the two chosen attributes. This may, or may not be useful, but it suggests that future work should allow new features to be generated interactively which are *any* user defined logical function of the original binary symbolic attributes.

Using The HTM Mapping To Emphasise Different Aspects of The Data

Class Distance Based Maps

The HTM mapping algorithm attempts to find a linear or non-linear projection of the N-dimensional patterns onto a 2 dimensional surface which is displayed as the data map. The projection is adjusted until the point to point distances in the N-space are reflected as nearly as possible in the 2-dimensional map space.

This idea can be extended to force the projection to reflect some externally defined point to point distances in the N-space. In particular, if the user arbitrarily defines the distances between each pair of points in the N-space, the HTM algorithm can use these distance values to adjust the projection until they are reflected as nearly as possible in the map. The only change to the HTM algorithm is that the user defined point-to-point distances are substituted for the pattern space point-to-point distances.

An important application of this idea arises when the user specifies the class of each example in the data set and the distance between each class type. The HTM algorithm will then attempt to form a map which separates the examples of each class in a way which is reflects the specified inter-class distances.

The inter-class distances are defined in the form of a matrix of values which are typically set to unity. This causes the HTM to attempt to place all examples of the same class at the same position in the map and make all classes equidistant. Using a 2-dimensional map, this is only possible if the number of classes is less than or equal to three. In practice it has been found that using a 2-dimensional map in conjunction with four or more classes still works satisfactorily, although the mapping error increases as the number of specified classes increases.

Natural Cluster Distance Based Maps

The aim of this type of mapping is to cause the N-dimensional data to be projected into the 2-dimensional map in such a way that natural clusters in the data are displayed with maximum separation. The chosen approach is to perform a cluster analysis on the data in its N-space. The N-space distances between the centroids of each cluster are then measured and the HTM mapping is adjusted so that the map space reflects the inter-centroid distances as nearly as possible. The entire data set is then subjected to this mapping and displayed in the map.

There are several possible clustering algorithms which could be used in this application, but the most suitable candidate appears to be a simplified form of the algorithm used in the Kohonen Network [5]. The basis of the clustering is as follows:

i) Before the mapping commences, assign random positions to the centroids of the k clusters into which the data is to be grouped. Note that the value of k can be adjusted interactively whilst the program is running. It would be usual to start with a low value then progressively increase the value of k, all the while observing to see if widely separated clusters are being generated.

ii) Randomly select a pair of data examples from the data set. This is the same pair as used in each iteration of the HTM algorithm. For the purposes of clustering, only one example is needed. However, since two are selected for the HTM algorithm, it is computationally efficient to use both at each iteration of the clustering algorithm.

iii) Measure the distances from both of the selected examples to each of the current data centroids.

iv) Assign each of the two selected examples to the clusters with the nearest centroids.

v) Update the centroids of the selected clusters, such that the centroid of each cluster moves fractionally closer to the corresponding data example.
ie.

$$G^i = (1-\alpha).G^i + \alpha.X^i \qquad (17)$$

Where G^i is the centroid of the i^{th} cluster to which the selected data example X^i has been found to belong, and α is a small constant which determines the rate at which the centroid is updated.

v) Repeat the process starting at (ii).

Examples of The Use of The HTM Mapping Tool

A simple example which illustrates the basic operation of the HTM mapping tool is shown in Fig.1. The figure shows the map formed by the HTM of the points on lines of "longitude" and "latitude" on a sphere. Points in the northern hemisphere are labelled as class 1 and points in the south, class 2. Examples of the two class are represented by the symbols '+' and 'x'. It is evident that there is no way of unambiguously mapping a three dimensional distribution into a two dimensional space and so the map is simply a projection of the sphere onto a plane. The type of mapping used in this example is based upon the pattern space distances.

A more interesting example is shown in Figs. 2a, 2b, and 2c which show HTM maps for artificial nine dimensional data which contains nine gaussian distributed clusters at each vertex of a nine dimensional hypercube. Three different classes have been arbitrarily defined, with three clusters being associated with each class.

The map in Fig 2a is based upon class distances. i.e The map has been generated to maximise the distance between examples of different classes. The map in Fig. 2b has been generated to maximise the distances between natural clusters in the data and the map in Fig. 2c has been generated to reflect the point to point pattern space distances in the nine dimensional space.

Finally, maps of some real 7-dimensional financial data are presented in Figs. 3a and 3b. Each example in this data set has been categorised into one of two classes. It can be seen from the pattern space distance based map in Fig. 3a that the classes generally occupy different class regions, and that there is some clustering into sub groups. (Investigation of the sub-groups revealed that they were caused by one of the attributes in the data only taking integer values.) It can be seen from the class distance based map in Fig. 3b that the classes would be rather easily separated using a single discriminant plane, although several examples are positioned ambiguously and would be mis-classified using such a classifier.

Summary

This paper has presented a new approach to visualising high dimensional data in the form of a two dimensional map. The mapping is performed iteratively by a MLP using a modified form of the backpropagation algorithm called the Hidden Target Mapping (HTM), which can be adapted to generate maps which reflect the N-space pattern distances, inter-class separation, or cluster separation in the data. A key property of the visualisation tool embodying the HTM mapping is that it is computationally much faster than the standard Sammon Map when using large data sets. This allows the user to interact with the map by changing parameters of the mapping as well as the form in which the data is presented to the HTM. When using data sets of up to several thousand examples, the user can see the map change within seconds and explore the data.

References

[1]Tattersall G.D, Chichlowski K, Limb R, *Pre-processing and visualisation of decision support data for enhanced machine classification*, Intelligent Systems Engineering Conf. Edinburgh, August 1992.

[2] Tattersall G.D, Chichlowski K, Limb R, Totton K.A.E, *Feature extraction and visualisation of decision support data*, BTTJ, July 1992.

[3] Sammon J.W, *A Non Linear Mapping For Data Structure Analysis*, IEEE Trans. on Computers, vol C-18, no 5, May 1969.

[4] Rumelhart D.E, Hinton G.E, Williams R.J, '*Learning Internal Representations by Error Propagation'*, in *Parallel Distributed Processing*, Rumelhart D.E, McLelland J.L, and the PDP Research Group, The MIT Press, Vol. 1, pp.318-362, 1986.

[5] Kohonen T., *Clustering, Taxonomy, and Topological Maps of Patterns*. Proc. Int. Conf. on Pattern Recognition, Oct. 1982.

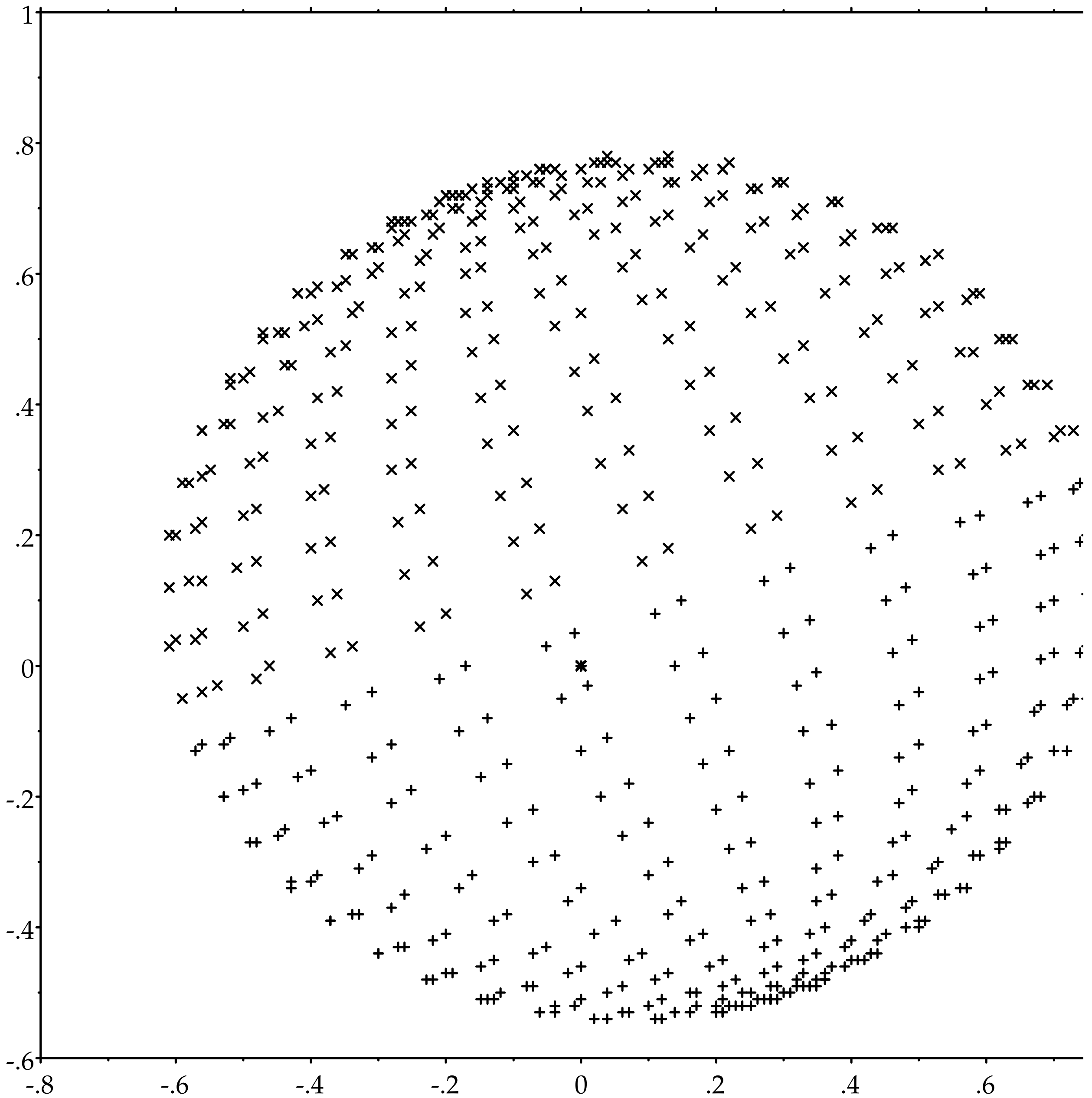

Fig. 1: HTM Mapping of Points on a Sphere

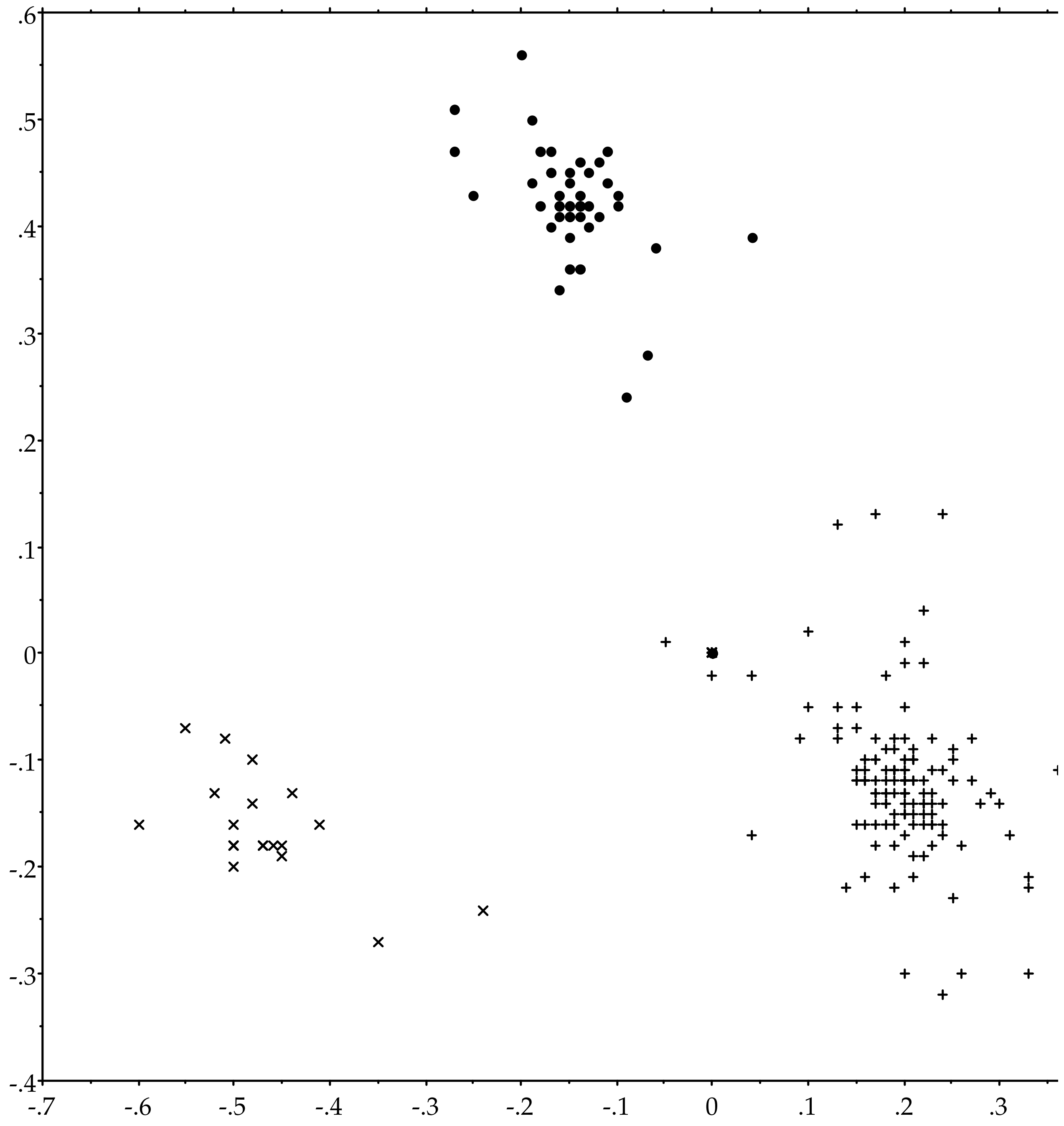

Fig. 2a Class Distance HTM Map of Nine Dimensional Data

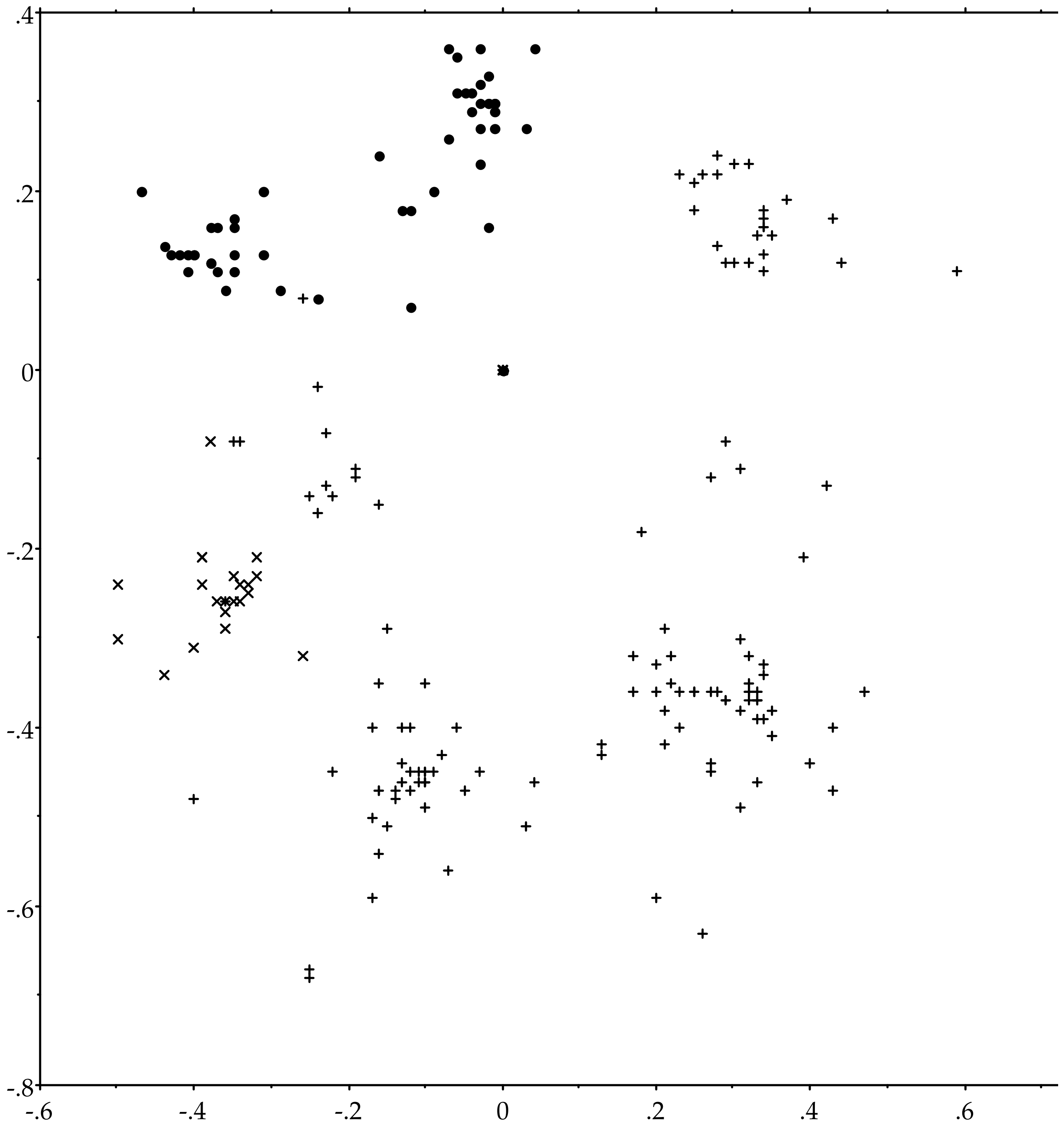

Fig. 2b HTM Map based Upon Natural Cluster Distances

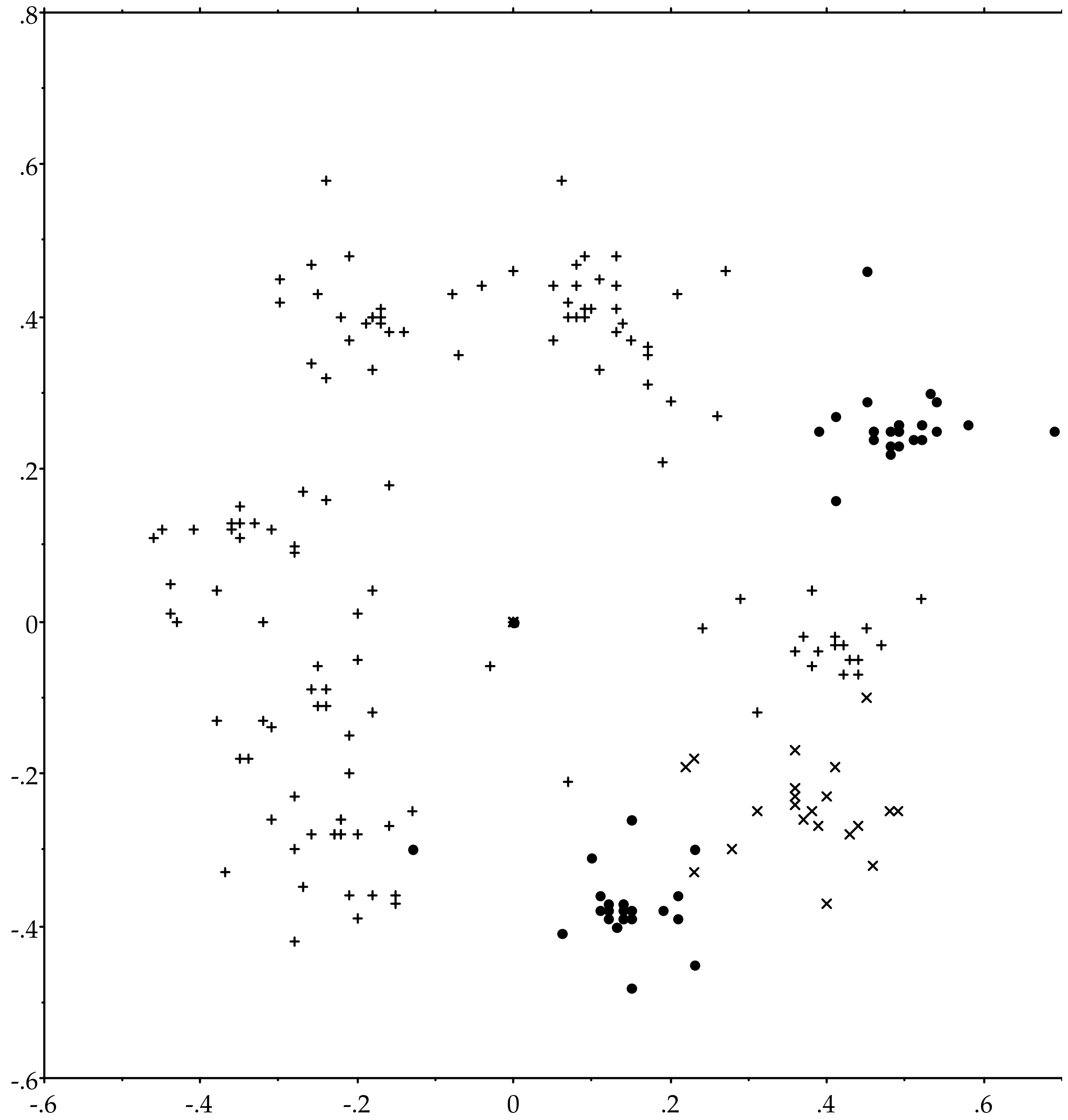

Fig. 2c HTM Map based Upon Pattern Space Distances

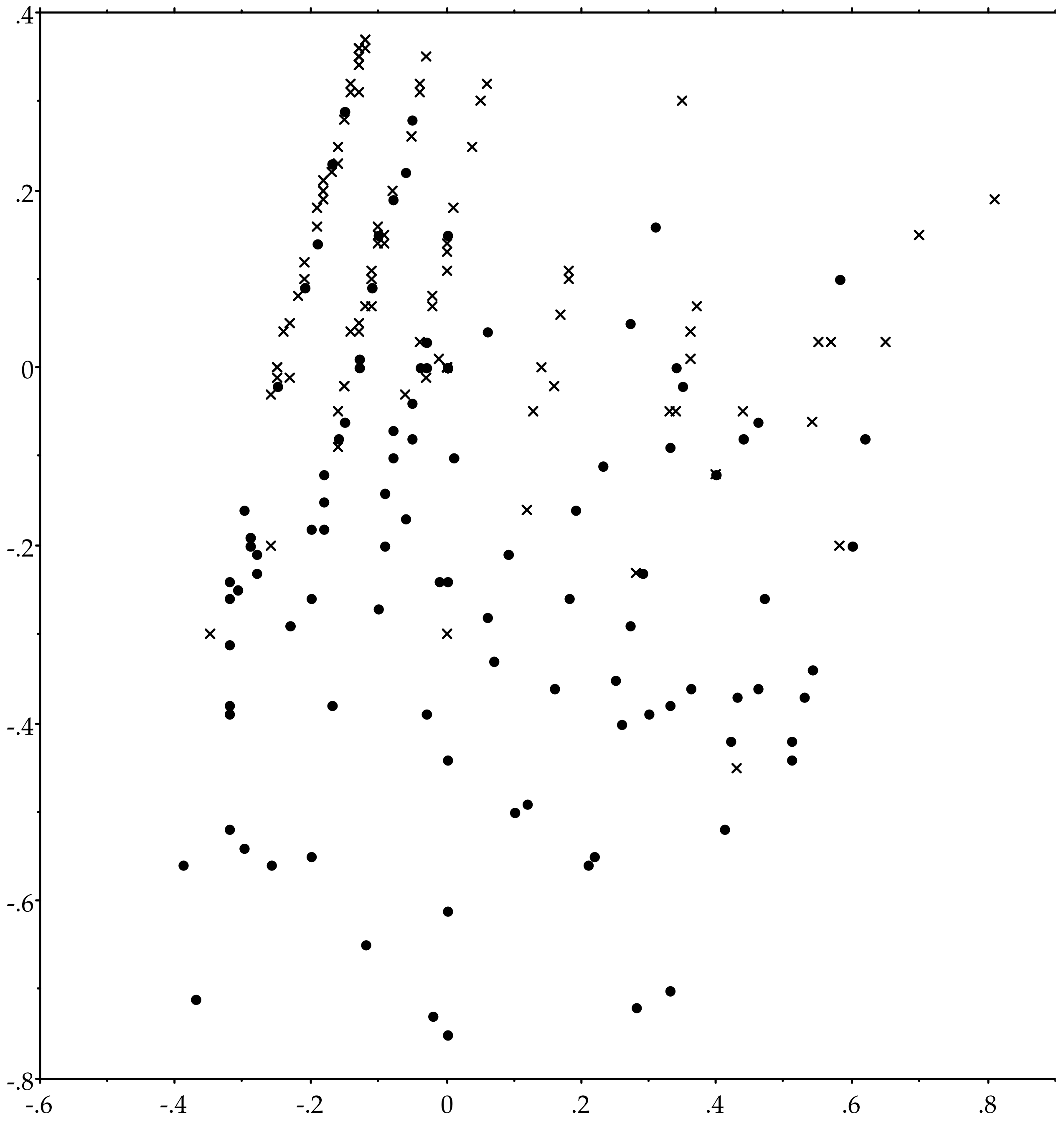

Fig. 3a HTM Map Based Upon Pattern Space Distances in Seven Dimensional Financial Data

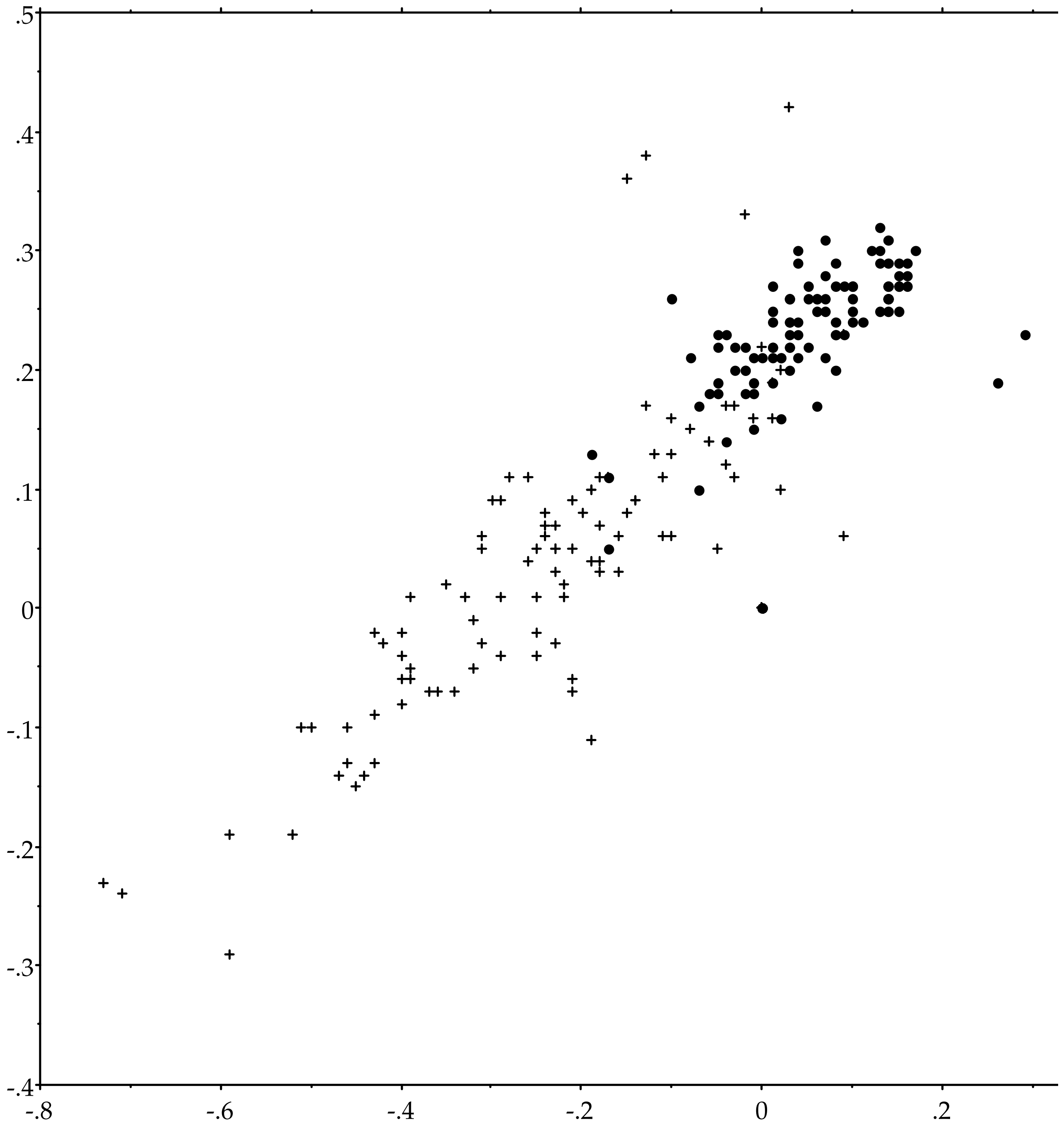

Fig. 3b HTM Map Based Upon Class Distances in Seven Dimensional Financial Data