RESERVOIR COMPUTING AND DATA VISUALISATION

Wesam Ashour[1], Tzai Der Wang[2] and Colin Fyfe[3]

[1] *Islamic University of Gaza, Palestine*
washour@iugaza.edu.ps

[2] *Department of Industrial Engineering and Management,*
Cheng Shiu University, Taiwan, ROC
tw952276@gmail.com

[3] *University of the West of Scotland, UK*
Colin.Fyfe@uws.ac.uk

Abstract

We consider the problem of visualisation of high dimensional multivariate time series. A data analyst in creating a two dimensional projection of such a time series might hope to gain some intuition into the structure of the original high dimensional data set. We review a method for visualising time series data using an extension of Echo State Networks (ESNs).The method uses the multidimensional scaling criterion in order to create a visualisation of the time series after its representation in the reservoir of the ESN. We illustrate the method with two dimensional maps of a financial time series. The method is then compared with a mapping which uses a fixed latent space and a novel objective function.

1 Introduction

Identifying structure in high dimensional data spaces is a difficult problem. One method frequently used is to project the data onto a low dimensional manifold and allow a human investigator to search for structure in this manifold by eye. There are many artificial neural network methods e.g. [5, 8, 10, 13] for projecting data onto low dimensional manifolds. In this paper, we will review a visualisation method based on an artificial neural network which is specifically designed to display low dimensional projections of time series data.

Time series data presents opportunities for projection which other data sets may not have: in a typical time series, nearby (in time) samples often have values which are close to one another. We will develop a neural network method which captures the dynamical nature of such data but projects the data onto a 2 dimensional manifold on which we can view the original time series. We emphasise that because we are dealing with a time series in which present values depend on previous values, a static neural network would not be able to capture all the information which a recurrent neural network can since a recurrent network can capture the relationships between current and previous values.

We will base the visualisation property of the neural network on the neuroscale algorithm [21] which minimised the objective function of multidimensional scaling. However the neuroscale algorithm is based on radial basis networks which are static neural networks containing no feedback connections and are thus unable to satisfactorily represent dynamic information. We will continue to use the multidimensional scaling objective function however since we wish to capture dynamical information, we use a network that can retain a memory of the past - the echo state network.

It is well known that real biological neural networks have many feedback connections and it is recognised that such recurrent nets have information processing powers that feedforward neural networks do not have. However while we have efficient algorithms for training feedforward neural networks, no efficient algorithms have existed for recurrent neural networks. Reservoir computing is a relatively new type of artificial neural network which attempts to overcome this known difficulty in training recurrent neural networks. We will concentrate on a type of reservoir known as echo state networks [11, 15]. In this paper, we first combine a neuroscale-type algorithm with the echo state network for visualising time series data.

This paper presents an extension of work discussed in [22, 23]: in [23], we compared the new method with projections from principal component analysis and self-organising maps of various varieties[8, 9, 10, 13] and showed that the method using ESNs is much superior to those without. In this paper, we will compare the method with a different visualisation method based on an underlying latent space similar to that developed for the generative topographic mapping [5].

2 Echo state networks

Echo state networks (ESNs) consist of three layers of 'neurons': an input layer which is connected with random and fixed weights to the next layer which forms the reservoir. The neurons of the reservoir are connected to other neurons in the reservoir with a fixed, random, sparse matrix of weights. Typically only about 10% of the weights in the reservoir are non-zero. The reservoir is connected to the output neurons using weights which are trained using error descent. We emphasise that only the reservoir to output weights are trainable; the other sets of weights are fixed. It is this feature which gives the ESN the property of being easily and efficiently trained.

We first formalise the idea of reservoir. W_{in} denotes the weights from the N_u inputs $\mathbf{u}$ to the N_x reservoir units $\mathbf{x}$, W denotes the $N_x \times N_x$ reservoir weight matrix, and W_{out} denotes the $(N_x + 1) \times N_y$ weight matrix linking the reservoir units to the output units, denoted $\mathbf{y}$. Typically $N_x \gg N_u$. W_{in} is fully connected and fixed (i.e. the weights are non-trainable). In the standard echo state network W is fixed but provides sparse connections: in this work only 10% of the weights in W are non-zero. W_{out} is a fully connected set of trainable weights (the "readout weights"). It is often stated that the W weights should be such that the spectral radius (its greatest eigenvalue) is less than 1 to ensure stability in the reservoir *when there is no input* (see (1)). However a more useful heuristic for the more usual conditions (in which there is a non-zero input) is that there should be a playoff between the magnitude of the reservoir-reservoir weights, W, and those from the inputs, W_{in}: the larger W is, the more memory of previous values can be retained but of course we cannot ignore the inputs entirely.

The network's dynamics are governed by

$$\mathbf{x}(t) = f(W_{in}\mathbf{u}(t) + W\mathbf{x}(t-1)) \tag{1}$$

where typically $f(.) = tanh(.)$ and t is the time index. The feed forward stage is given by

$$\mathbf{y} = W_{out}\mathbf{x} \tag{2}$$

This is followed by a supervised learning of the output weights, W_{out}. If we are using online learning, a simple least mean square rule gives

$$W_{out}(t+1) = W_{out}(t) + \eta(\mathbf{y}_{target}(t) - \mathbf{y}(t))\mathbf{x}^T(t) \tag{3}$$

where η is a learning rate (step size) and $\mathbf{y}_{target}(t)$ is the target output corresponding to the current input.

Most research effort has gone into giving the reservoir weights some structure, either by pre-training with, for example, a self-organising map [4, 14] or by fixing the topology of the reservoir [17]. In this paper, we will leave the input to reservoir and the reservoir to reservoir weights fixed in their standard form (i.e. exactly as described in the previous section) but investigate training the output weights by optimising the multidimensional scaling criterion exactly as neuroscale did for static data [16, 21].

3 Neuroscale

Multidimensional scaling (MDS) is a set of methods used for visualisation: it identifies a set of projections of data points such that the distances between the points' projections are as close as possible to the distances between the original data points. If the projections are to a two dimensional space, we can hopefully gain some intuition about the relative positions of the original high dimensional data by viewing the two dimensional projections.

Classical MDS minimises a "stress" or objective function given by

$$E_{CMDS} = \sum_{i,j} (L_{ij} - D_{ij})^2 \tag{4}$$

where D_{ij} is the distance, often Euclidean, between two data samples, $\mathbf{x}_i$ and $\mathbf{x}_j$, and L_{ij} is the distance between their respective projections, $\mathbf{y}_i$ and $\mathbf{y}_j$. There are a number of variations on this basic theme, the most popular of which is the Sammon mapping which minimises

$$E_{Sammon} = \sum_{i,j} \frac{(L_{ij} - D_{ij})^2}{D_{ij}} \tag{5}$$

This concentrates on getting the local distances correct by giving less weight to being accurate with distances which are large in data space. Recently [19, 20, 25] extended MDS.

However MDS suffers from a set of problems based on the fact that there is one latent point for every data point; therefore

- The training time for the algorithm rises with the number of data points: for N points, we must recalculate $N(N-1)/2$ distances in latent space.
- There is no inherent generalisation: for each new sample, we must recalculate the whole new mapping.

Therefore a new mapping, Neuroscale, based on radial basis functions was created: a full account of this mapping can be found in [21].

The neuroscale mapping is based on a radial basis network: let the input vector be $\mathbf{x}$; then the response of the i^{th} hidden neuron is given by

$$h_i = \exp\left(-\frac{\| \mathbf{x} - \mathbf{c}_i \|}{\sigma}\right) \tag{6}$$

and the output of the network, y_j is given by

$$y_j = \sum_i w_{ji} h_i \tag{7}$$

Typically only the w_{ji} parameters are changed during learning which is generally supervised so that each input $\mathbf{x}$ must be associated with a target value, $\mathbf{t}$.

The neuroscale algorithm replaces this supervised learning with a new training process known as 'relative supervision': instead of error descent with respect to the target values, neuroscale uses the classical MDS stress function (4) as the error to be minimised which it does by updating the weights using gradient descent.

$$\Delta w_{ij} \propto -\frac{\partial E_{CMDS}}{\partial w_{ij}} \tag{8}$$

in which only the terms in L_{ij} are functions of the weights.

4 The MDS-reservoir model

We will maintain the first two sets of weights of an echo state network in the standard format discussed in Section 2. However we will change the reservoir to output weights using the MDS criterion (4). Since we are interested in using the resulting algorithm for visualisation, we will have two dimensional outputs, $N_y = 2$ There exist two possibilities for calculating D_{ij}:

1. $D_{ij} = \| \mathbf{u}_i - \mathbf{u}_j \|$. This is in line with the original neuroscale algorithm [21] and simply substitutes the non-linearity of the radial basis functions with that of the reservoir. This method has the advantage that the D_{ij} values need only be calculated once before training commences since they will not subsequently change.
2. $D_{ij} = \| \mathbf{x}_i - \mathbf{x}_j \|$. This has the advantage that D_{ij} then takes into account the history of the time series which is echoing in the reservoir. This is something that the first formulation would not allow.

If we denote the value of the reservoir activations in response to input $\mathbf{u}_t$ to be $\mathbf{x}_t$, i.e. t defines the time index of the input and corresponding reservoir activation, then we can develop a recursive relationship between any two reservoir activations. Thus

$$\begin{aligned}
(\mathbf{x}_i - \mathbf{x}_j) &= W_{in}\mathbf{u}_i + W\mathbf{x}_{i-1} - W_{in}\mathbf{u}_j - W\mathbf{x}_{j-1} \\
&= W_{in}(\mathbf{u}_i - \mathbf{u}_j) + WW_{in}\mathbf{u}_{i-1} \\
&\quad + W^2\mathbf{x}_{i-2} - WW_{in}\mathbf{u}_{j-1} - W^2\mathbf{x}_{j-2} \\
&= \dots \\
&= W_{in}(\mathbf{u}_i - \mathbf{u}_j) \\
&\quad + WW_{in}(\mathbf{u}_{i-1} - \mathbf{u}_{j-1}) \\
&\quad + W^2W_{in}(\mathbf{u}_{i-2} - \mathbf{u}_{j-2}) + \dots
\end{aligned} \tag{9}$$

Since the output of the reservoir contains memories of the past, it is preferable to use $D_{ij} = \| \mathbf{x}_i - \mathbf{x}_j \|$.

We can see that, provided the relationship between W and W_{in} is set appropriately (see Section 2), the effect of differences in the past data set is washed out with time and the most recent differences play the strongest role in D_{ij}. However the other differences do have an effect as our simulations have shown. This is based on the echo state property which means that there still exists an echo of past inputs reverberating in the reservoir and this property still exists in the new training mechanism based on the MDS criterion. In fact this form of training which is based on gradient descent of (4), utilises the echo of differences between training patterns corresponding to inputs at different time instances.

Thus the method is essentially a batch algorithm: we apply the inputs $\mathbf{u}_i, i = 1, ..., N$ where N is the number of samples, sequentially in order and note the corresponding reservoir values $\mathbf{x}_i$. Then the differences $(\mathbf{x}_i - \mathbf{x}_j)$ are used in the neuroscale algorithm to update the W_{out} parameters which were initialised to small values from a uniform distribution in $[0, 0.001]$. Typically in the experiments in the next section, we use $N_x = 300$, $N = 1000$ and we iterate 200 times over the data set.

We report on experiments using (1) but we also investigated leaky integrator neurons which used

$$\mathbf{x}(t) = \lambda \mathbf{x}(t-1) + (1-\lambda) f(W_{in}\mathbf{u}(t) + W\mathbf{x}(t-1)) \tag{10}$$

but found qualitatively similar results for the simple data sets though perhaps a slight improvement for chaotic data sets.

4.1 Financial Data

The book "Irrational Exuberance" [18] uses a stock market dataset which represents the closing price each month of the S & P Composite to illustrate its thesis and the author has made this dataset available to all. We have used this closing price, the dividend, earnings, consumer price index and the long term interest rate to form a 5 dimensional data set. We illustrate the method with the first 1000 samples of this data set. Note that the data in the book contains the date and some columns containing calculations both of which we omitted so as to use solely the raw financial data.

Results from the method of this paper using the reservoir of 300 neurons are shown in Figure 1. The top diagram shows a projection of all of the data while the bottom shows the projection of each of the years numbered from 1871 so that for example 1901 is year 30 on the diagram.

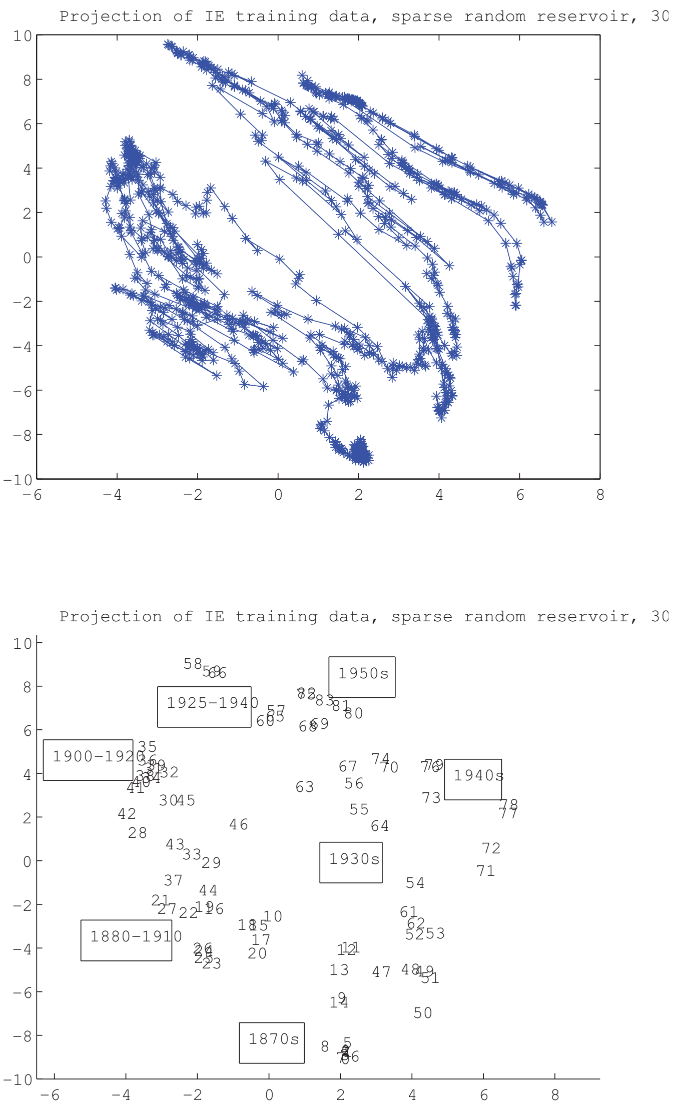

Figure 1. Projections of IE data [18] using the multidimensional scaling criterion. Top:1000 samples. Bottom:showing the years, 1871= year 0.

We can readily distinguish a great deal of information: the projection of all the data shows two distinct regions in the map: the region to the bottom left contains the projections of the data from 1870 to 1917 while that in the top right of the diagram shows the projections of 1918 onwards. The first region shows little or no structure however the second does appear to show distinctly quasi-periodic behaviour which is very suggestive of the boom-bust cycles we know so well. This type of projection can be seen with this method and many different structures of reservoir and many different divergences.

We find that with the random reservoirs the resulting projections are remarkably stable with only the number of neurons being a determining factor [24]. We have tried networks larger than 300 and found no improvements; thus we feel confident in stating that, for the irrational exuberance data, the random reservoir-MDS method with 300 neurons gives the best projections and is quite reliably the same projection for any random reservoir with the stated level of sparsity.

Also these results were achieved with a Java program running for 2 minutes 28 seconds on a standard PC, 2.99 GHz, 3.46 GB RAM; this time is comparable with the other methods.

4.2 Adding distance information

[21] suggests incorporating class information into the neuroscale algorithm by importing this information into the definition of D_{ij}. Thus $D_{ij} = \alpha \parallel \mathbf{x}_i - \mathbf{x}_j \parallel + (1-\alpha)s_{ij}$ where s_{ij} contains information about whether sample $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same class or not. We do not have class information however we can use the fact that the order of presentation of the data contains information. Thus we can use the same method but create a function $s(i, j)$ such that it is maximum when i and j represent nearby times and decreases as the time distance increases. We have used $s_{ij} = a(1-\exp(-(i-j)^2/b)$ in which the parameter a is data-dependent and typically chosen to be the standard deviation of the data and $b = 5N$ which gives a window of size approximately 100 of values close to 0 and rises to a away from this window. Of course we use wrap around when calculating s_{ij}.

With the financial data set discussed in this paper, we found a similar structure with this method as that found in Figure 1.

5 Inverse-weighted K-means Topology-preserving Mapping (IKToM)

The Generative Topographic Mapping (GTM) [5, 6, 7] was developed by Bishop as a probabilistic version of the SOM, in order to overcome some of the problems of this map, especially the lack of an objective function.

The Generative Topographic Map (GTM) is a mixture of experts model which treats the data as having been generated by a set of latent points. The GTM can be described as a non-linear latent variable model that defines a mapping from the latent space to the data space, generating a probability density within the latter.

We have recently used this idea of a latent space [1, 2, 3] but with an objective function which is not a probabilistic function and thus is not optimized using the Expectation-Maximization algorithm (EM). Instead, we have developed the Inverse Weighted K-means algorithm (IWK) as the learning process. IWK is more robust to the initial parameters than K-means and the EM algorithm. It is also provides better results regarding convergence to a local optimum.

By adding a latent space model to the IWK algorithm we have created the Inverse-weighted K-means Topology-preserving Mapping (IKToM) which we illustrate on the financial data used above.

The basis of our model is K latent points, $t_1, t_2, \cdots, t_K$, which are going to generate the K prototypes, $\mathbf{m}_k$. To allow local and non-linear modeling, we map those latent points through a set of M basis functions, $f_1(), f_2(), \cdots, f_M()$. This gives us a matrix Φ where $\phi_{kj} = f_j(t_k)$. Thus each row of Φ is the response of the basis functions to one latent point, or alternatively we may state that each column of Φ is the response of one of the basis functions to the set of latent points. One of the functions, $f_j()$, acts as a bias term and is set to one for every input. Typically the others are gaussians centered in the latent space. The output of these functions are then mapped by a set of weights, W, into data space. W is $M \times D$, where D is the dimensionality of the data space, and is the sole parameter which we change during training. We will use $\mathbf{w}_i$ to represent the i^{th} column of W and Φ_j to represent the row vector of the mapping of the j^{th} latent point. Thus each basis point is mapped to a point in data space, $\mathbf{m}_j = (\Phi_j W)^T$.

We may update W either in batch mode or with online learning. In IKToM we used the Inverse Weighted K-means algorithm to create a new topology preserving algorithm.

Each data point is visualized as residing at the prototype on the map which would win the competition for that data point. However we can do rather better by defining the responsibility that the j^{th} prototype has for the i^{th} data point as

$$r_{ji} = \frac{\exp(-\gamma \parallel \mathbf{x}_i - \mathbf{w}_j \parallel^2)}{\sum_k \exp(-\gamma \parallel \mathbf{x}_i - \mathbf{w}_k \parallel^2)} \tag{11}$$

We then project points taking into account these responsiblities: let y_{ij} be the projection of the i^{th} data point onto the j^{th} dimension of the latent space; then

$$y_{ij} = \sum_k t_{kj} r_{ki} \tag{12}$$

where t_{kj} is the j^{th} coordinate of the k^{th} latent point. When we use these algorithms for visualisation purposes, it is these y-values (which are typically two dimensional coordinates) which we use. Note that this method represents each data point $\mathbf{x}_i$ by a value $\mathbf{y}_i$ where $\mathbf{y}_i$ is a weighted sum of the coordinates of the original latent points. An alternative (which is typically used the SOM) is to find the latent point with greatest responsibility for the data point and allocate its $\mathbf{y}_i$ value at this latent point.

5.1 Simulations

Experiment1:
In this experiment, we have fed the reservoir (300 units) with the first 1000 samples of the IE dataset. Then we have projected the reservoir (1000 x 300) onto a two-dimensional latent space using IKToM. Although IKToM is more robust than GTM to the initial parameters, it is still sensitive to the initial conditions, thus we run the algorithm more than once and explore the results. Figure 2 and 3, show the results after running IKToM twice.

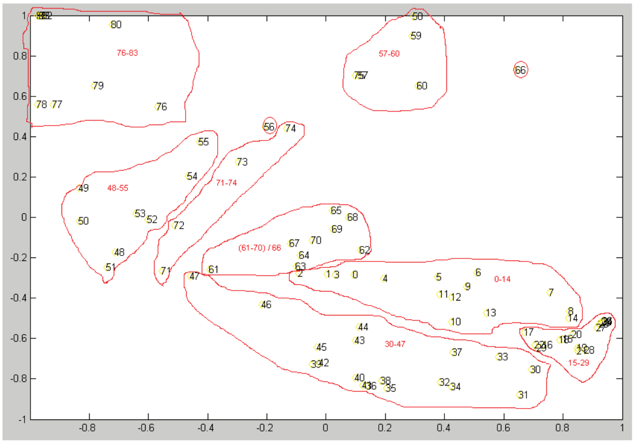

Figure 2. Projection of IE dataset (1000 samples) using Reservoir-IKToM.

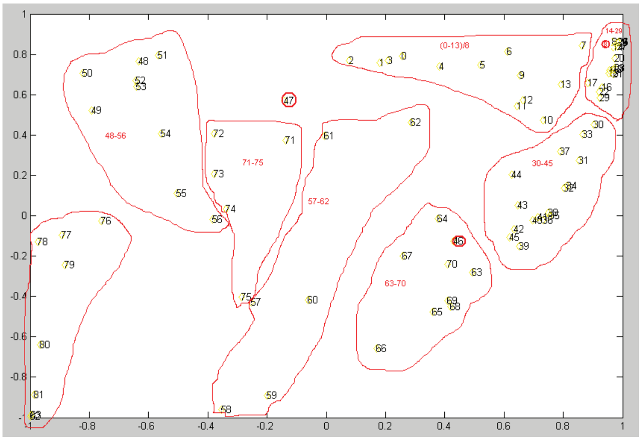

Figure 3. Projection of IE dataset (1000 samples) using Reservoir-IKToM.

In Figures 2 and 3, we can see an interesting projection; consecutive years have been projected close to each other, e.g. 0-13, 15-29 and 76-83. From the two Figures 2 and 3, we can notice the following:

1. We got approximately the same groups of years; the reservoir makes the projection more robust to the initial parameters.
2. Years from (14-29) are the farthest from years (76-83).
3. Years from (47-60) are the closest to years (75-83).
4. We can combine groups of years to get bigger group e.g. we can combine years 0-30 in one group.
5. It is possible to divide the whole years into 3 big groups - (0-45), (46-75) and (76-83).

6. Years 57 and 75 are projected close to each other.

Experiment 2:
This is similar to experiment 1, except we used only IKToM to project the data directly and without using a reservoir. Figure 4 and 5 show the results of projection.

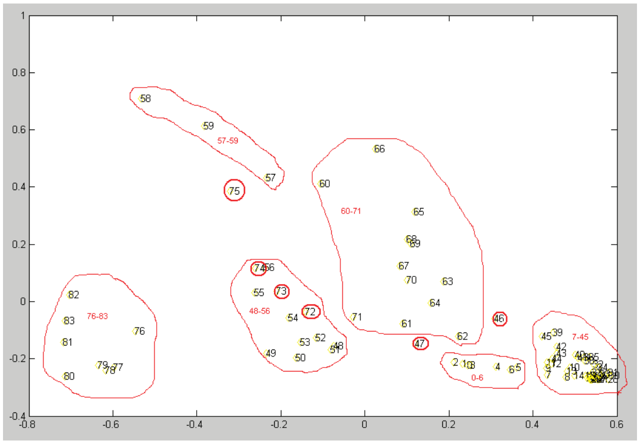

Figure 4. Projection of IE dataset (1000 samples) using IKToM.

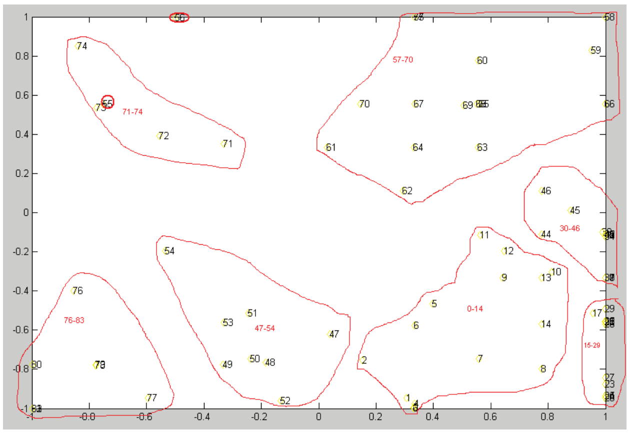

Figure 5. Projection of IE dataset (1000 samples) using IKToM.

From Figures 4 and 5, we can notice the following:

1. Projecting Reservoir units is more robust to the initial parameters than projecting the data directly. We can see that the Figures 2 and 3 are approximately similar and they provide the same characteristics of data, while in the Figures 4 and 5 there are differences.

2. In Figure 4, IKToM gave good results similar to Reservoir-IKToM. However, normally Reservoir-IKToM gave better results than IKToM results, see next experiment.

From Experiments 1 and 2, we can notice the following:

1. IKToM gave poor results with the whole IE dataset, while Reservoir-IKToM does not.

2. Reservoir-IKToM gave better results than IKToM.

6 Discussion

A feature of the SOM [12, 13] and IKToM [2, 3] is that they quantise data to specific points. This is an advantage in that they scale up to large data sets easily however note that when we wish to visualise individual points we must create some method for assessing the responsibility of each prototype for each data sample. Typically we calculate $d(\mathbf{x},\mathbf{p}_i)$ and create responsibilities

$$r_i = \frac{\exp(-d(\mathbf{x},\mathbf{p}_i))}{\sum_j \exp(-d(\mathbf{x},\mathbf{p}_j))} \tag{13}$$

and use this to get a position for the projection of $\mathbf{x}$:

$$\mathbf{y} = P(\mathbf{x}) = \sum_i r_i \mathbf{t}_i \tag{14}$$

where $\mathbf{t}_i$ is the two dimensional position in feature space corresponding to prototype $\mathbf{p}_i$.

Multidimensional scaling, on the other hand, does not require this but does however not scale so well with size of sample. The neuroscale and MDS–reservoir methods do scale up well and do not require special arrangements for visualisation. However the method does require keeping a record of the projections.

Also, while the other methods can be used in either batch or online paradigms, the multidimensional scaling methods can only be used in batch form: we need to store the projection of each set of reservoir values and only subsequently can we calculate the distance between pairs of projections. It is worth stating too that we must calculate the distance between the pairs of reservoir values too with this method.

However even with these caveats, we believe that we have demonstrated that the MDS–reservoir method is much better than the standard visualisation techniques available in the artificial neural network literature for visualisation of temporal data. Clearly the method could be equally well used to other data which contain positional information.

Finally we have applied a second algorithm IKToM to the data and shown that it is the combination of reservoir plus the projection method which gives the most powerful results.

7 ACKNOWLEDGEMENTS

The first author would like to extend his sincerest thanks and appreciation to the Arab Fund for Economic and Social Development for their support and grant Fellowship-ID-734 to accomplish and complete this research.

References

[1] W. Barbakh, M. Crowe, and C. Fyfe. A family of novel clustering algorithms. In *7th international conference on intelligent data engineering and auto- mated learning, IDEAL2006*, pages 283-290, Springer. ISSN 0302-9743, 2006.

[2] W. Barbakh, and C. Fyfe. Local vs global interactions in clustering algorithms: Advances over k-means. *International Journal of Knowledge-based and Intelligent Engineering Systems*, ISSN 1327-2314, 12(2):83–99, 2008.

[3] W. Barbakh, Y. Wu, and C. Fyfe. *Non-standard exploratory data analysis.* Springer, 2009.

[4] S. Basterrech, C. Fyfe, and G. Rubino. Initializing echo state networks with topographic maps. In *2nd International Conference on Morphological Computation (ICMC 2011).*, 2011.

[5] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

[6] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. Developments of the generative topographic mapping. *Neurocomputing*, 21(1):203–224, 1998.

[7] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. Gtm: A principle alternative to the self-organizing map. *In Advances in neural information processing systems*, 5:354–360, 1997.

[8] C. Fyfe. A scale invariant feature map. *Network: Computation in Neural Systems*, 7:269–275, 1996.

[9] C. Fyfe. *Hebbian Learning and Negative Feedback Artificial Neural Networks*. Springer, 2004.

[10] C. Fyfe. Two topographic maps for data visualization. *Data Mining and Knowledge Discovery*, 14:207–224, 2007. ISSN 1384-5810.

[11] Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks. Technical Report 148, German National Research Center for Information Technology, 2001.

[12] T. Kohonen. *Self-Organising Maps*. Springer, 1995.

[13] Teuvo Kohonen. *Self-Organizing Maps*, volume 30. Springer Series in Information Sciences, third edition, 2001.

[14] Mantas Lukoševičius. On self-organizing reservoirs and their hierarchies. Technical Report 25, Jacobs University, Bremen, 2010.

[15] Mantas Lukoševičius and Hebert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3:127–149, 2009.

[16] Ian Nabney. *Netlab*. Springer, 2001.

[17] Ali Rodan and Peter Tiňo. Minimum complexity echo state network. *IEEE Transactions on Neural Networks*, 22:131–44, 2011.

[18] R. J. Shiller. *Irrational Exuberance*. Princeton University Press, 2000,2005.

[19] J.. Sun. *Extending metric multidimensional scaling with Bregman divergences*. PhD thesis, School of Computing, University of the West of Scotland, 2011.

[20] J. Sun, M. Crowe, and C. Fyfe. Extending metric multidimensional scaling with bregman divergences. *Pattern Recognition*, (44):1137–1154, 2011.

[21] Michael E. Tipping. *Topographic mappings and feed-forward neural networks*. PhD thesis, The University of Aston in Birmingham, 1996.

[22] T. D. Wang and C. Fyfe, Training Echo State Networks with Neuroscale *Proceedings of the International Conference on Technologies and Applications of Artificial Intelligence*, 2011)

[23] T. D. Wang and X. Wu and C. Fyfe, Comparative study of visualisation methods for temporal data *2012 IEEE Congress on Evolutionary Computation* , 2012

[24] T. D. Wang and C. Fyfe. The role of structure size and sparsity in echo state networks for visualisation *The United Kingdom Conference on Computational Intelligence*, 2011.

[25] X. Wang, M. Crowe, and C. Fyfe. Dual stream data exploration. *International Journal of Data Mining, Modelling and Management*, 2011.