The 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2014)
Nov. 12 - 15, 2014 at Double Tree Hotel by Hilton, Kuala Lumpur, Malaysia

Using Auto-Associative Neural Networks to Compress and Visualize Multidimensional Data

Zalhan Mohd Zin
Industrial Automation Section, Universiti Kuala Lumpur Malaysia France Institute
(Tel: +603-89132800; Email: zalhan@unikl.edu.my)

Abstract – To interpret the information hidden in multidimensional data can be considered as challenging and complicated task. Usually, dimension reduction or data compression is considered as the first step to data analysis and exploration of multidimensional data. Here, the focus is given to study Auto-Associative Neural Networks (AANNs) technique for data compression and visualization. AANNs have the ability to deal with linear and nonlinear correlation among variables. This technique is often referred to as nonlinear Principal Component Analysis (NLPCA) or could also be known as Bottleneck Neural Networks (BNNs) due to their specific structures that consist of combination of compression and decompression networks. The trained AANNs can reduce high dimensional data onto lower dimensional data by compressing them on its bottleneck layer that later can be used for data visualization. In this paper, the technique of AANNs are described, developed using high level computer language and applied on two different multidimensional datasets. The results have shown that AANNs are able to compress multidimensional data into only two non linear principal components at its bottleneck layer and these compressed data can provide visualization of different clusters of data.

Keywords—Auto-Associative Neural Networks, Dimension Reduction, Data Clustering, Iris flowers, Olive oils

1. Introduction

The field of Artificial Neural Networks have lately gained a place in many engineering applications due to better disposal of computing power and many engineering tools applications such as MatLab™ [1] and SOM Toolbox [2] [3]. These toolboxes have become widely available and are considered as a working system in the world for various applications [4]. Interpreting and understanding the information in multidimensional data can be considered as challenging and complicated tasks. The compression, dimension reduction and the visualization of these multidimensional data could provide solutions to these tasks. As mentioned in [5] [6], dimension reduction has been considered as the first step to process understanding of multidimensional data in order to see the significant patterns in it. There are many dimension reduction techniques that have been used previously. One of them is Principal Component Analysis (PCA), a commonly used technique to find pattern of high dimensional data as described in [7]. This technique has been widely described, used, compared and evaluated in many research works such as in [8] [6] [9] [5]. However, the effectiveness of linear PCA technique is quite limited due to its inefficiently to deal with nonlinear correlated variables. For these problems, nonlinear dimension reduction or projection methods are used. Some examples of nonlinear dimension reduction techniques are Multidimensional Scaling (MDS), Locally Linear Embedding (LLE), Isomap, Kernel PCA [10], Self-Organizing Maps (SOMs) [11] [12] [13] and also Auto-Associative Neural Networks (AANN) [5] [6]. Here, the focus was given to the latter technique for multidimensional data clustering and visualization. The AANNs have been used in particular for data compression or dimension reduction in the field of chemical applications, missing data estimations and image compressions [6] [14] [15]. In this research, AANNs have been used to perform compression, clustering and visualization of multidimensional data. They have been re-developed using high level computer language, applied and analyzed by computing it on multidimensional data of a well-known Iris flowers and Italian olive oils datasets.

2. Methods

2.1 Auto-Associative Neural Networks (AANNs)

These AANNs have often been regarded as an alternative to PCA for unsupervised learning, clustering, and outlier detection [16]. As described in [5], it is also known as Non Linear Principal Component Analysis (NLPCA) or can be also known as Bottleneck Neural Networks (BNN) [6]. AANNs can be applied to the same problems as in conventional PCA such as dimension reduction and visualization, fault detection etc. AANNs are feed forward network and are trained to map approximations of input vectors to their corresponding outputs. As described in [17], they can be viewed as circuits of highly interconnected units with adjustable interconnection weights. They can also be considered as a particular class of neural networks in which the target output patterns is identical to its input patterns. Conceptually, AANNs works by trying to imitate human brain abilities such as learning and memorizing. Similarly to the biological neural system, a neuron is the smallest functional unit in AANNs. This neuron receives simultaneously a number of input signals, n, in which, each of these signals is associated with their respective weights, w. The weighted sum of signals becomes the argument of the activation function, and the function value is the output of the neuron. This activation function can be

linear or nonlinear function which allows neuron to perform linear or nonlinear calculation operations on its output.

The structure of AANNs usually consists of several layers: input layer, map layer, bottleneck layer, de-map layer and output layer as shown in Figure 1. In this figure too, the structure has three nodes in input and output layers, five nodes in map and de-map layers, and two nodes in bottleneck layer. This structure could also be seen as a combination of two different networks, compression network and de-compression network, as stated in [5] [6] [17]. These two networks "join" in the middle at bottleneck layer. In compression network, the input is known but the output is not known while the opposite situation happens in de-compression network. During training of AANNs, the dataset is compressed to few potential variables, number of which corresponds to the number of nodes in the bottleneck layer. This number must be lesser than the number of nodes at input or output layers. Then the output of the bottleneck is decompressed at the de-mapping layer. Hence, the bottleneck outputs of AANNs represent a type of nonlinear principal components, which are frequently more relevant than PCA for analyzing nonlinear, real-world datasets.

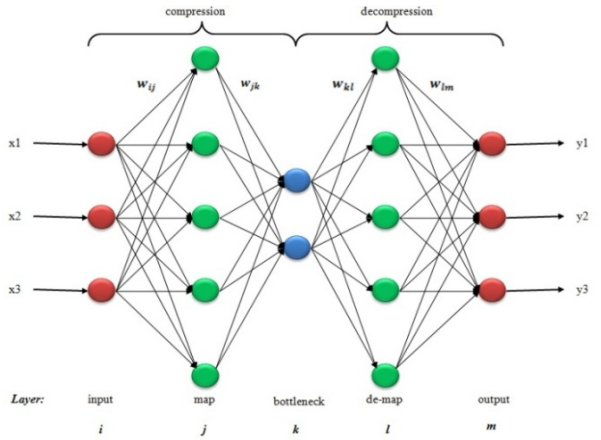

Fig. 1: Structure of AANN consists of 3 nodes at input and output layers, 5 nodes at map and de-map layers and 2 nodes at bottleneck layer.

In AANNs, its ability to deal with linear and nonlinear correlation between variables is related to the type of the transfer function used in each of its layer. According to [17] [5], the neurons in map and de-map layers must use non linear or sigmoid transfer functions. This is necessary to ensure proper functioning of AANNs [6]. On the other hand, the nodes in bottleneck and output layers could use either linear transfer functions or sigmoid transfer functions. In our case, the sigmoid binary function being continuous and monotonically increasing has been used. This function take input values X and scale them into the range $\{0,..,1\}$. It has been defined as follows:

$$\sigma(X) = \frac{1}{1+e^{-X}} \qquad \text{(Eq-1)}$$

In brief, the learning process of AANNs start with random initialization of all weights w which link all neurons in all layers in the network (Figure 1). The values at nodes at input layer are the original input values. The number of nodes in this layer represents the dimension of the data and the process starts at iteration p. At map layer, the algorithm continues to calculate the values at all nodes. Assuming $M_{(i,j)}$ function for this operation at map layer, it can be defined as:

$$M_{(i,j)} = \sigma(\sum_{i=1}^{nbmp} \sum_{j=1}^{nbinput} x_j w_{ji}) \qquad \text{(Eq-2)}$$

with *nbmp* and *nbinput* are number of nodes in map and input layers respectively. The same processes continue at bottleneck, de-map and output layers with the following equations:

$$B_{(i,j,k)} = \sigma(\sum_{k=1}^{nbtl} \sum_{i=1}^{nbmp} M_{(i,j)} w_{ik}) \qquad \text{(Eq-3)}$$

$$D_{(i,j,k,l)} = \sigma(\sum_{l=1}^{nbdmp} \sum_{k=1}^{nbtl} B_{(i,j,k)} w_{kl}) \qquad \text{(Eq-4)}$$

$$O_{(i,j,k,l,m)} = \sigma(\sum_{m=1}^{nboutput} \sum_{l=1}^{nbdmp} D_{(i,j,k,l)} w_{lm}) \qquad \text{(Eq-5)}$$

with *nbtl, nbdmp* and *nboutput* represent the number of nodes and $B_{(i,j.k)}, D_{(i,j,k.l)}$, and $O_{(i,j,k,l,m)}$ represent functions at bottleneck, de-map and output layers respectively. So, at iteration p, the values obtained at a particular neuron m, $y_m(p)$ at output layer is equalled to $O_{(i,j,k,l,m)}(p)$. In fact, through this process, the input signals are propagated through network from left to right (input layer to output layer). During AANN training, the weights of the nodes are updated using back-propagation technique as described in [18]. Once the signals arrive at all nodes m in output layer, the error signals are then calculated. The error signal at node m in output layer is calculated as following:

$$e_m = y_{d,m}(p) - y_m(p) \qquad \text{(Eq-6)}$$

where $y_{d,m}(p)$ is the desired output of node m in output layer at iteration p. These errors are back-propagated from output layer back to input layer. The rule used to update weights between output layer and de-map layer is:

$$w_{lm}(p+1) = w_{lm}(p) + \Delta w_{lm}(p) \qquad \text{(Eq-7)}$$

where l represent the nodes at de-map layer and weight correction $\Delta w_{lm}(p) = \alpha \times y_l(p) \times \delta_m(p)$ where $\delta_m(p)$ was the gradient error at neuron m in the output layer at iteration p and α is the learning rate. In these AANNs multilayer neural network, gradient error has to be calculated as the derivative of the transfer function multiplied by the error at the neuron output. Example for each neuron m at output layer, the gradient error $\delta_m(p)$ is:

$$\delta_m(p) = \frac{\partial y_m(p)}{\partial X_m(p)} \times e_m(p) \qquad \text{(Eq-8)}$$

with $y_m(p)$ is the output of neuron m at output layer and $D_{(i,j,k,l)}(p)$ is the net weighted input to neuron m. Both are at iteration p. Since binary sigmoid activation function as defined in (eq-1) has been used, so the derivative become as follow:

$$\delta_m(p) = \frac{\partial\left\{\frac{1}{1+e^{(-X_m(p))}}\right\}}{\partial X_m(p)} \times e_m(p)$$

$$\delta_m(p) = \frac{e^{(-X_m(p))}}{\{1+e^{(-X_m(p))}\}^2} \times e_m(p)$$

thus, (eq-8) can be written as:

$$\delta_m(p) = y_m(p) \times [1 - y_m(p)] \times e_m(p) \qquad \text{(Eq-9)}$$

where

$$y_m(p) = \frac{1}{1+e^{(-X_m(p))}}$$

Then, all the weights w_{lm} which link de-map and output layers are updated using eq-7. For calculation of weight corrections between de-map layer and bottleneck layer, the same equation has been applied as follows:

$$w_{kl}(p+1) = w_{kl}(p) + \Delta w_{kl}(p) \qquad \text{(Eq-10)}$$

$$\Delta w_{kl}(p) = \alpha \times y_k(p) \times \delta_l(p) \qquad \text{(Eq-11)}$$

where $\delta_l(p)$ represents the error gradient at neuron l at the de-map layer. It is defined as:

$$\delta_l(p) = y_l(p) \times [1 - y_l(p)] \times \sum_{m=1}^{nboutput} \delta_m(p) w_{lm}(p) \qquad \text{(Eq-12)}$$

where m is the number of neurons in output layer and:

$$y_l(p) = \frac{1}{1+e^{(-X_l(p))}}$$

$$X_l(p) = \sum_{k=1}^{nbtl} B_{(i,j,k)}(p) \times w_{kl} - \theta_l$$

where *nbtl* is the number of neurons in the bottleneck layer, $B_{(i,j,k)}(p)$ represent the values of nodes at the bottleneck layer at iteration p and θ_l is threshold at layer l of de-map layer. The process of updating all the weights, w_{jk} between bottleneck and map layers and also w_{ij}, weights between map and input layers have been done using the same equations as in previous update of weights. When all the weights w_{ij} have been updated through back-propagation error signals, the first iteration was considered as complete. The Mean Square Error (MSE) between all nodes *i* at input layer and all nodes *m* at output layer is calculated as follow:

$$MSE = \frac{1}{N} \sum_{i=1;m=1}^{nbinput;nboutput} (X_i - X_m)^2$$

As mentioned previously, AANNs have been trained to map input vectors to their corresponding output vectors, in other word, each value at input nodes should be identical to its respective value at output nodes. During AANNs training, the algorithm map input vectors to their corresponding output vectors. This training is an iterative process where at each iteration; another input data is randomly selected from the same training dataset to become the input signals for the networks. The process loops until certain condition of MSE value is fulfilled. The lower the MSE value, the more identical input-output nodes could have been obtained. In our case, the training was stopped when the MSE was lower than 0.001.All these details structure, equations and rules of AANN have been developed and written using C/C++ language.

2.2 The Datasets

Two different datasets have been used to observe the performance of data clustering and visualization using our developed AANNs algorithm. The first dataset was the iris flowers dataset. It consists of 150 flowers, divided equally into three different classes: Setosa, Versicolor and Virginica. Each of them has 50 samples of iris flowers. This commonly known dataset can be obtained in [19]. In each sample or data, four features were measured (in cm) which are: the length and width of sepal; and the length and width of petal.

Fig. 2: The geographical locations of Italian olive oils samples.

The second dataset that has been used was Italian olive oils dataset. As described in [6], this dataset contains the concentrations of eight fatty acids namely palmitic, palmitoleic, stearic, oleic, linoleic, eicosanoic, linolenic, eicosenoic. These fatty acids are distributed in 572 samples or data of olive oils in Italy. The data was taken from nine different growing regions in Italy which are North Apulia, Calabria, South Apulia, Sicily, Inland Sardinia, Coastal Sardinia, East Liguria, West Liguria and Umbria. Figure 2 shows the geographical locations of these samples taken from nine different regions in Italy.

3. Experimental Frameworks and Results

Two types of experimental works have been conducted in this research. The first one was Iris flowers data clustering and secondly, Italian olive oils data clustering. All these experiments have been conducted using our developed AANN algorithm based on the algorithm's description in Section 2. The experiments were conducted using Intel Core2Quad 2.5 GHz with 2.0 Gb memory.

3.1 Data clustering using AANNs

Firstly, the experiments of AANNs to cluster data have been conducted using total of five layers which consisted of two layers of input and output and three hidden layers of map, bottleneck and de-map layers. These AANNs structure were similar to the one shown in Figure 1. As mentioned in the study on NLPCA using AANN [5], three hidden layers were essential to achieve optimal non linear feature extraction. The input was also auto-scaled or normalized as in [6]. For Iris flowers dataset, the structure of AANNs consisted of four nodes at input and output layers, ten nodes at map and de-map layers, and two nodes at bottleneck layer.

The result of this dataset's projection by AANN's two activation nodes at bottleneck layer is shown in Figure 3. The high dimensionality Iris flowers data has been reduced by AANNs to only two dimensions represented by its two bottleneck nodes. The flowers from Setosa and Virginica classes have been appropriately clustered far from each other's, while the flowers from Versicolor class have been clustered appropriately between these two classes, and closer to Virginica class. This distribution of Iris flowers data was found to be consistent with the nature of the Iris flowers dataset itself where Setosa class is linearly separable from the other two classes, and these two other classes are not linearly separable from each other as described in [19].

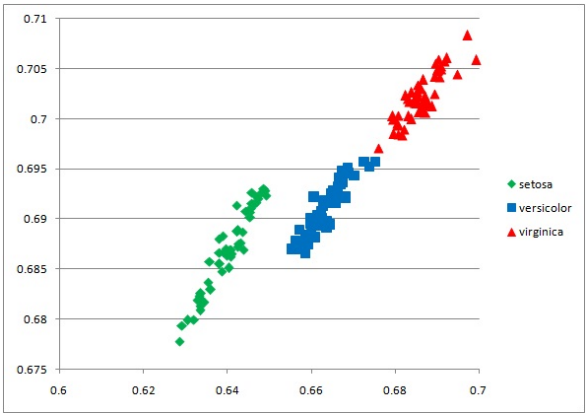

Fig. 3: The projection of iris flowers dataset defined by the two activation nodes at bottleneck layer in AANNs.

For the second dataset, the structure of AANNs had eights nodes at input and output layers, twenty nodes at map and de-map layers, and two nodes at bottleneck layer. In this experimental work, the eight dimensional data of Italian olive oils dataset have been reduced into only two dimensional data situated at the two activation nodes at bottleneck layer. The result of this dataset's projection is shown in Figure 4. In this figure too, it can be noticed that samples taken from four south-east regions of Calabria, Sicily and North Apulia have formed a group on the right side. On the other hand, another group was formed on the left side that contained samples taken from the three north-west regions of Umbria, West Liguria, East Liguria and island of Sardinia (Inland and Coastal). Meanwhile, the largest samples that were taken from South Apulia were projected at the top from left to right side above the two groups. Similarly as in [6], these samples were far from North Apulia and were closer to Inland and Coastal Sardinia. In the left group, these Inland and Coastal Sardinia samples were closer to each other and have formed a subgroup which were different from East Liguria, West Liguria and Umbria.

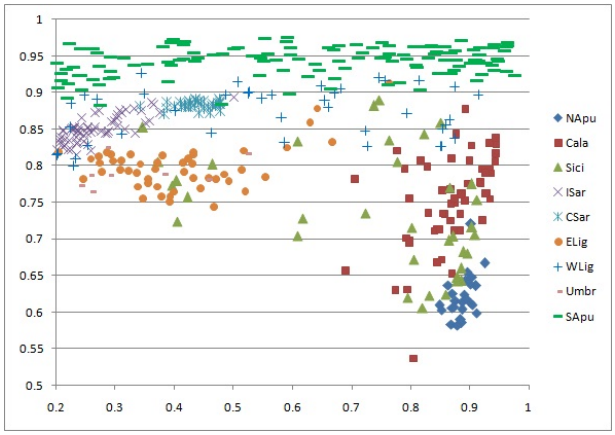

Fig. 4: The projection of Italian olive oils dataset defined by the two activation nodes at bottleneck layer in AANNs.

Through the experimental results of clustering Iiris flowers and Italian olive oils datasets, AANNs algorithm have been able to cluster and visualize both datasets into their appropriate classes. From these results too, the nature of the data clusters was found similar to the characteristics of the datasets themselves. However, it can be observed that the visualization of AANNs was better and clearer in Iris flowers than in Italian olive oils case. This might due to the fact that more complexity exists in the second dataset such as higher number of data, more similar data, higher features or dimensionality of data and also higher number of classes.

4. Conclusions

The AANNs can be considered as a very powerful tool in exploratory data analysis with its ability to deal with linear and nonlinear correlation among variables. Through this study, AANNs can perform dimension reduction and data visualization usually using its two bottleneck activation nodes. In this paper, the technique of AANNs has been developed using high level language of C/C++, based on the structure, equations and rules described in

Section 2. Our developed AANNs algorithm has been later used to perform compression, clustering and visualization of multidimensional data using two datasets: Iris flowers and Italian olive oils. The experimental results have shown that AANNs algorithm has been able to cluster and visualize these two high-dimensionality data appropriately according to their respective classes. This could give us the way to explore the potentiality to enhance this technique by integrating it with another technique such as SOM with the purpose to provide solutions to data clustering of very complex multidimensional data ie: gene expression data with multidimensional variability [20].

References

[1] Mathworks Incorporation, [Online]. Available: http://www.mathworks.com/products/neural-network/index.html. [Accessed 10 March 2012].

[2] J. Vesanto, "SOMToolbox Homepage," 2008. [Online]. Available: http://www.cis.hut.fi/somtoolbox/. [Accessed 20 January 2011].

[3] J. H. E. A. &. J. P. Juha Vesanto, "SOM Toolbox for Matlab 5," 2000.

[4] V. M. Stone, "The Autoassociative Neural Network-A Network Worth Considering," in *World Automation Congress (WAC)*, Hawaii, 2008.

[5] M. A. Kramer, "Nonlinear Principle Component Analysis Using Autoassociative Neural Networks," *AIChE Journal,* vol. 37, pp. 233-243, 1991.

[6] B. W. D. L. M. M. Daszykowski, "A Journey Into Low-dimensional Spaces With Autoassociative Neural Networks," *International Journal of Pure and Applied Analytical Chemistry Talanta,* vol. 59, pp. 1095-1105, 2003.

[7] L. I. Smith, 2002. [Online]. Available: http://neurobot.bio.auth.gr/2005/a-tutorial-on-principal-components-analysis/. [Accessed 05 February 2012].

[8] S. L. J. L. Giraudel, "A Comparison of Self Organizing Map Algorithm and Some Conventional Statistical Methods for Ecological Community Ordination," *Journal of Ecological Modelling,* vol. 146, pp. 329-339, 2001.

[9] F. V. N. T. M. Jaisheel Mistry, "Missing Data Estimation Using Principle Component Analysis and Autoassociative Neural Networks," *Journal of Systemics, Cybernatics and Informatics,* vol. 7, no. 3, pp. 72-79, 2009.

[10] P. J. K. M. G. Ali Anaissi, "A Framework for High Dimensional Data Reduction in the Microarray Domain," in *IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, 2010.

[11] T. Kohonen, "The Self-Organizing Maps," in *Proceedings of the IEEE*, 1990.

[12] A. Flexer, "On the Use of Self Organizing Maps for Clustering and Visualization," in *International Conference on Principle on Data Mining and Knowledge Discovery*, Prague, Czech Republic, 1999.

[13] E. A. Juha Vesanto, "Clustering of the Self Organizing Maps," *IEEE Transaction on Neural Networks,* vol. 11, no. 3, pp. 586-600, 2000.

[14] m. F. K. C. L. G. J. K. Matthias Scholz, "Non-linear PCA: A Missing Data Approach," *Journal of Bioinformatics,* vol. 21, no. 20, pp. 3887-3895, 2005.

[15] S. Y. B. C. M. A. Abidi, "Image Compression Using Hybrid Neural Networks Combining The Auto-Associative Multilayer Perceptron and The Self Organizing Feature Map," *IEEE Transaction on Consumer Electronics,* vol. 40, no. 4, pp. 796-811, 1994.

[16] B. J. H. J. D. L. Mark J. Embrechts, "Augmented Efficient BackProp for Backpropagation Learning in Deep Autoassociative Neural Networks," in *International Joint Conference on Neural Networks (IJCNN)*, Barcelona, 2010.

[17] J.-C. G. Gaetan Kerschen, "Feature Extraction Using Auto-Associative Neural Networks," *Smart Materials and Structures,* vol. 13, no. 1, 2004.

[18] M. Negnevitsky, Artificial Intelligence: A Guide to Intelligent Systems 2nd Edition, Pearson Education Limited, 2005.

[19] R. A. Fisher, "UCI Machine Learning Repository," 2006. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Iris. [Accessed 31 January 2011].

[20] E. Mesbahi, "Cryptic codes in non-coding DNA: Autoassociative Neural Networks and multidimensional Self Organising Maps (SOM) mediated prediction of positional significance of cis-elements in co-regulated expression systems," 29 March 2012. [Online]. Available: http://www.ncl.ac.uk/marine/research/project/1997. [Accessed 15 July 2012].