The Geodesic Self-Organizing Map and Its Error Analysis

Yingxin Wu **Masahiro Takatsuka**

ViSLAB, School of Information Technologies
The University of Sydney
National ICT Australia*

chwu@it.usyd.edu.au

masa@vislab.net

Abstract

The Self-Organizing Map (SOM) is one of the popular Artificial Neural Networks which is a useful in clustering and visualizing complex high dimensional data. Conventional SOMs are based on the two-dimensional (2D) grid structure, which usually results in less accurate representation of the data. Several SOMs using spherical data structures have been proposed to remove the "border effect". In this paper, we compared our proposed Geodesic SOM (GeoSOM) with the 2D hexagonal SOM by experiments. The result shows that the GeoSOM not only runs as fast as the conventional 2D SOM, but also represents the data more accurately within fewer training epochs.

Keywords: Self-Organizing map, sphere tessellation, geodesic dome, error analysis

1 Introduction

1.1 Self-Organizing Map

The Self-Organizing Map (Kohonen 2001) is a type of the Artificial Neural Network and has been greatly appreciated in analysing and visualizing massively complex high-dimensional data. The applications of the SOM can be found not only in the fields of engineering but also in other areas such as the medical, agricultural and social science fields (Brown 1995, Tokutaka et. al. 1999, Lin 2001). This neural network attempts to map a high-dimensional real number space onto a low-dimensional (typically 2D or 3D) integer space. It exhibits the following characteristics during its non-linear mapping process: 1) multidimensional scaling, 2) topological mapping and 3) unsupervised clustering.

The neighbourhood relationship, which is used to describe the topological information, is usually defined by a 2D rectangular or hexagonal lattice such that every grid unit has 4 or 6 neighbours (Figure 1). Each grid unit is used as a neuron and is associated with a weight vector. The dimension of the weight vector is the same as the input data. During the training phase, all neurons compete with each other for the input signals. The winner r' and its neighbours r update their weights w by:

$$w_r(t) = w_r(t-1) + \alpha h_{rr'}(v - w_r(t-1)),$$

where t is the current training epoch, α is the learning rate, $h_{rr'}$ is the neighbourhood function and v is one of the input samples (Ritter 1992). The value of $h_{rr'}$ is maximized when r=r' and decreases to zero as the distance between r' and its neighbors approaches a predefined distance—the update radius. The most commonly used neighborhood function is Gaussian:

$$h_{rr'} = \exp(-(r - r')/2\sigma^2),$$

where σ is the update radius. The training will progress for several hundred epochs. In order for the SOM to converge, the learning rate and update radius will continually decrease as the training epoch increases.

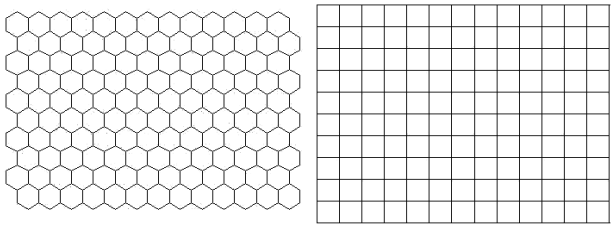

Figure 1 The hexagonal grid (left) and the rectangular grid (right).

1.2 The Border Effect

At the end of the training, all data samples are equally represented by the neurons, and the neighboring units in the grid tend to model similar regions of the data space. This topological mapping is achieved due to the neighborhood relationships. However, the grid units at the boundary of the SOM have fewer neighbours than the units inside the map so that they have fewer chances to be updated. The border neurons are therefore more similar to the neurons inside than the data they represent. This is known as the "border effect". Several people (Hsiang 2001, Kohonen 2001) pointed out that the border effect not only lowers the accuracy, but also makes the SOM harder to converge.

Some approaches have been proposed to overcome the problem such as the heuristic weighting rule method by Kohonen (2001) and the local-linear smoothing by Wand and Jones (1995). Besides the mathematical solutions,

*National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council

Ritter (1999) suggested that a SOM can be implemented on a spherical lattice (such as tessellated platonic polyhedra) so that all the boundaries are eliminated. He also pointed out that the spherical SOM would be very suitable for data with underlying directional structures. Since then, some SOMs based on different spherical grids have been implemented and applied to variety kinds of datasets.

2 Existing Spherical Lattices and the Spherical SOM

For the 2D SOM lattice, the hexagonal grid is considered to be preferable over the rectangular grid because every grid unit has the same number of direct neighbours, and the distances between a unit and its six direct neighbours are the same. However, it is a fact that such uniformity can not be achieved on the sphere except in the cases of the five platonic polyhedra — tetrahedron, cube, octahedron, icosahedron and dodecahedron (Pugh 1976).

Oyabu et. al (2003) developed a spherical SOM based on the latitude/longitude grid system which is commonly used in representations of the Earth's geometry. The latitude/longitude grid system has the advantage that each grid unit can be indexed by spherical coordinates. The positions of the grid units can thus be easily stored in a 2D array which makes searching for neighbours as easy as in the normal rectangular grid. But the distance between each grid unit and its direct neighbours varies greatly from the equator to the two poles. This would exaggerate the distances between the weight vectors along the equator and compress them at the two poles.

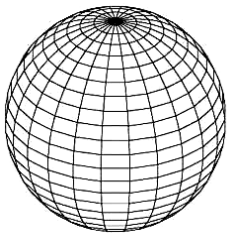

Figure 2 The latitude/longitude grid system

Sangole and Knopf (2003) implemented another spherical SOM using the lattice of tessellated platonic polyhedra. However, we would like to point out that this work considered the grid units on the tessellated platonic polyhedra to be uniformly distributed on the surface of the sphere. As mentioned above, only the five platonic polyhedra can achieve the uniformity. Further tessellating the polyhedra will come up with triangles with different sizes and edge lengths.

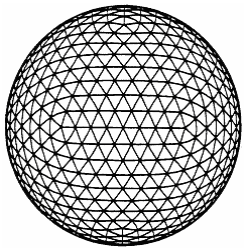

Figure 3 The tessellated platonic polyhedra

Uniformly distributing a large number of points on the sphere has been a problem of great interest to the mathematical researchers because it has important applications in the field of computation. However, their definition of uniformity is to minimize the extremal energies (Saff 1997, Rakhmanov et. al. 1994) which makes their solutions unsuitable to our purpose. Firstly, whether the generated points have same number of neighbours or not is not taken into consideration under these mathematical solutions; secondly, the nearest neighbours' distances vary greatly when only a few hundred points are needed.

3 The Geodesic SOM

In searching for the most appropriate spherical lattice, we have investigated various results from the research of mathematicians and the cartographers (Kevin Sahr et.al. 2003) and concluded that Icosahedron-based geodesic dome is very suitable for the spherical SOM (Wu and Takatsuka 2005). We also introduced a 2D data structure which supports fast neighbourhood searching on the geodesic dome and only use O(n) space. A directional GeoSOM was implemented based on the proposed data structure.

In this paper, we analyse the error distributions on the GeoSOM and compare it with the conventional 2D SOM. In this section, we will briefly review our proposed data structure and the neighbourhood searching algorithm. Section 4 discusses the error analysis and section 5 presents the experiment results of the GeoSOM.

3.1 Tessellate the Icosahedron

The tessellation process was introduced by Fuller (1975). Each triangular face of the Icosahedron is subdivided into a series of smaller triangles by lines running parallel to the original edges of the triangles. For example, to create a two frequency triangle face, firstly calculate the mid point of each edge. Then connect the 3 mid points using straight lines which subdivide the triangle into 4 smaller triangles. The spherical polyhedron obtained from this method is called the geodesic dome.

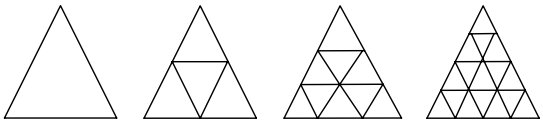

Figure 4 From left to right: one-frequency, two frequency, three frequency and four frequency tessellation.

3.2 The Data Structure and Neighbourhood searching

Finding the neighbours within a certain distance from a winning neuron is a very important step in the training of the SOM. Using the rectangular /hexagonal lattice, the grid units can be stored in a 2D array. This means that the neighbours of the winning neuron can be found using simple addition/subtraction arithmetic, see Figure 5. Storing the lattice of the geodesic dome is more complex compared to the 2D grid. The data structure should support

fast vertex indexing otherwise the neighbourhood searching would become the bottleneck in the SOM training process.

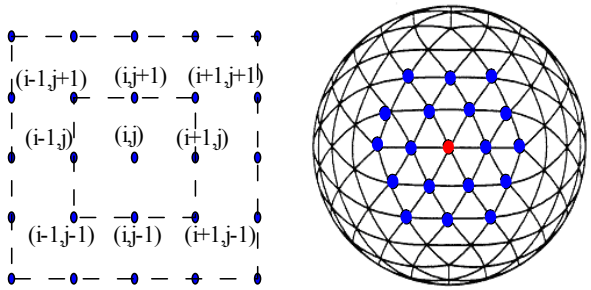

Figure 5 Left: the neighbourhood of a winning neuron on the rectangular lattice; Right: the neighbourhood of the winning neuron on the 4-frequency icosahedron

The basic idea is to treat the spherical lattice as a 3 dimensional graph. The distance between two vertices is defined to be the length of the shortest path. Given a specific vertex, we first find out its directly connected neighbours. Then go on to search the further away neighbours based on the direct neighbours and so on, up to a specified distance parameter (the update radius).

There are two common methods to store a graph: one is to calculate the adjacency matrix of all the vertices. The alternative is to store all the vertices into an array with each array element containing pointers to its direct neighbours. The first method takes $O(n^2)$ space, where n is the number of grid units. Therefore this method is not suitable for SOMs with a large number of neurons. The second method is clumsy because we need to carefully maintain a large number of pointers. The main drawback of these two methods is that if we need to further tessellate the geodesic dome to create more neurons, the adjacency matrix or all the direct neighbours' pointers need to be updated. The data structure we introduced can address the above problems and only uses O(n) space.

3.2.1 Data structure for an icosahedron based geodesic dome

The lattice of the geodesic dome can be unwrapped into a two-dimensional map, as shown in Figure 6. We created two axes U and V in the map such that each vertex in the map can be uniquely denoted by the two-dimensional coordinate pair (u, v).

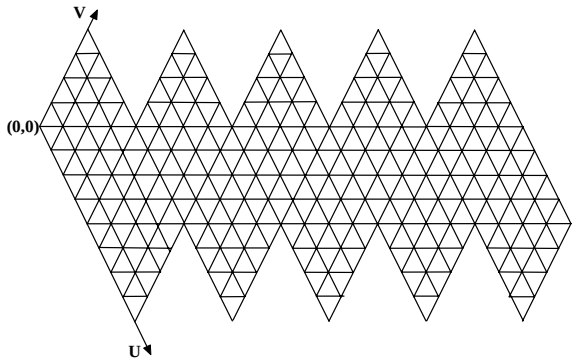

Figure 6 Unwrapped 4-frequency Icosahedron

By rotating all the vertices in the map such that the two axes U, V are orthogonal to each other, we can see that all the vertices of an icosahedron could be stored in a two-dimensional matrix (Figure 7)

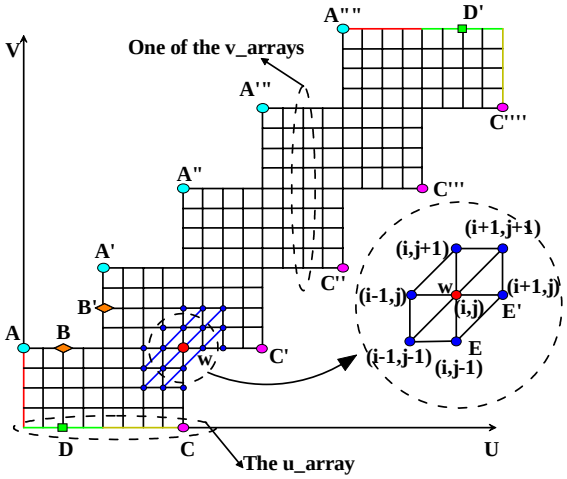

Figure 7 The data structure for the GeoSOM

Note that all the boundaries of the 2D map are joined on the sphere. Therefore there would be some duplicated points in the map. For example, in Figure 7, point A and point C are two poles on the geodesic dome. They each have four duplicated points (A' to A'''', C' to C''''). Our data structure maintains a list of the duplicated points. Only the points at the boundary may be duplicated, therefore the number of duplicated points will be very small compared to the total number of grid units. Therefore, only O(n) space is needed to store the geodesic dome, where n is the number of vertices on the dome.

3.2.2 Searching for neighbours in the data structure

The neighbours of a grid unit are grouped into levels. For the point W in Figure 7, its first level neighbours are the vertices that connect to it by only one edge. The second level neighbours are the vertices that connect to W by 2 edges and so on. In our algorithm, in order to get the n^{th} level neighbours, we need to search from the 1^{st} level neighbours until the target level is reached. When updating a wining neuron, all the neighbours from the 1^{st} level to the target level need to be updated. Therefore using this algorithm, searching for neighbours can be done efficiently.

Each point in the 2D geodesic dome map has 6 first level neighbours. The first level neighbours of W(i, j) can be calculated as: (i, j+1), (i+1,j+1), (i+1, j), (i-1,j-1), (i-1,j), (i, j-1). As mentioned before, there are duplicated points in the map's boundaries. A vertex's duplicate point should not be put into the neighbour list or the weight vector they represent will be updated several times. For example, in Figure 7, point E and E' are the same, so only one of them will be included. In the case when the winning neuron has duplicated points, such as point A in Figure 7, all the neighbours of its duplicated points (A' to A'''') need to be searched.

3.3 Increasing the frequency of the Geodesic Dome

The number of neurons needed in the SOM depends on the dataset size. In order to get the desire amount of neurons, we can tessellate the Icosahedron into different frequencies. The number of vertices on the icosahedron-based geodesic dome can be calculated by: $|V| = 10 \times f^2 + 2$, where f is the frequency number (Hoffmann 2002). Following is an illustration of how to obtain a 3-frequency geodesic dome from an icosahedron using the above data structure. Domes of other frequencies can be calculated under the same scheme.

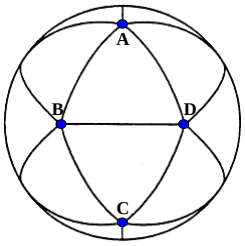

Figure 8 Two spherical triangles on the icosahedron

Suppose the icosahedron's circumscribing sphere is a unit sphere with its centre at the origin. ABD and BCD are two adjacent spherical triangles on the sphere (Figure 8). The left-most picture in Figure 9 shows how they are stored in the data structure. In order to tessellate the two triangles into 3 frequency, firstly break each edge of these two triangles into 3 equal length segments. This is equivalent to inserting 8 new vertices E, F, I, J, G, H, K, L between the old spherical arcs. Given the coordinates of A, B, C and D, it is not difficult to calculate the coordinates of the newly generated vertices geometrically. We can then calculate and insert vertices O and P which are the mid points of the arc FJ and KG respectively. All the other spherical triangles in the geodesic dome can be tessellated using this method.

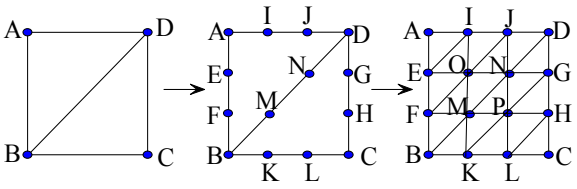

Figure 9 3-Frequency tessellation of the two spherical triangles.

Using our data structure, it is easy to obtain the m*n frequency geodesic dome from the m frequency dome. We only need to calculate and insert (n-1) vertices between the old arcs of the original dome and some new vertices between the new generated arcs. No adjacency matrix or neighbour pointers need to be updated. Hence, it is possible to apply the principal of the Growing SOM (Fritzke 1994, Fritzke 1995) to our GeoSOM.

After generating all the new vertices, some new duplicated points will be created along the boundaries of the map. Therefore we need to scan along the map boundaries to find all the duplicated points.

4 Measuring the Quality of the GeoSOM

The quality of a SOM is commonly evaluated by the quantization error (Kohonen 2001) which is the average distance between an input samples (d_i) to its best matching unit (BMU):

$$e = E / N \,, \tag{1}$$

$$E = \sum_{i=1}^{N} \text{distance}(d_i, w_{bmu}) \tag{2}$$

N is the number of input data and E is the total distance (error) between the data and their BMUs.

According to Kohonen (2001), SOMs "tend to approximate a smooth hyper surface" of the input data. This means in an ideal SOM, the input data should equally represented by the neurons and the errors should be evenly distributed on the map. In a 2D SOM, the neurons at the border have fewer neighbors than the neurons inside. Therefore neurons on the border receive biased training signals towards representing the data, which is modeled by the neurons inside, resulting in more errors.

In order to quantitatively analyze the spatial distribution of error in the GeoSOM, we introduce a new measure—Error Entropy. In Information Theory, entropy is used to measure the uncertainty of a process's outcome (Shannon and Weaver, 1964). It is defined as:

$$H = -(\sum_{i=1}^{c} p_i \log p_i) \,, \tag{3}$$

where c is the number of choices; p_i is the probability of each choice and $p_1+\ldots+p_c=1$. H will increase as the probabilities become more uniform and is maximized when all p_i are equal. Here we use this measure to quantify the degree of uniformity of the error distribution.

Suppose there are k input samples for which the BMU is the same neuron n. We define the neuron's error as the sum of the Euclidean distances between the neuron's weight vector and the k input samples.

$$E_n = \sum_{j=1}^{k} \text{distance}(d_j, w_n) \tag{4}$$

where w_n is the weight vector of neuron n; d_j is the j^{th} input data the neuron won.

Then the error entropy (EH) is given as:

$$EH = -(\sum_{i=1}^{m} e_i \log e_i)) \tag{5}$$

$$e_i = E_n / E \tag{6}$$

Here m is the total number of neurons; e_i is the proportion of a neuron's error to the total error E which is calculated by (2). According this definition, the more uniform the error distribution is, the higher the error entropy will be.

5 Experiments and Error Analysis

5.1 The Experiment Environment

We carry out several experiments to test the GeoSOM with different datasets. The training time, quantization errors and error entropy are calculated for each experiment. The results are compared against the corresponding 2D hexagonal SOMs generated by SOM_PAK (Kohonen 2001).

In order to reduce the computational overhead due to the programming language used, the GeoSOM is implemented as similar as possible to that of SOM_PAK. The parameters in training the two SOMs are also the same:

- In GeoSOM, the input data and weight vectors are stored in linked lists instead of arrays which are the same as SOM_PAK.
- The number of neurons is chosen to be approximately half the size of the input data. The size of GeoSOM is also selected to be as close to the size of the 2D SOM as possible.
- Before training, the two SOMs are randomly initialized. The input data points are normalized such that each dimension varies from 0 to 1. We use Euclidean distance to measure the difference between the input data and the weight vectors.
- During the training, both SOMs use the same learning rate, update radius and neighborhood function. In each of the following experiments, the initial learning rate is 0.8. The learning rate α(t) and update radius r(t) are decreased linearly according to the training epoch:

$$\alpha(t) = \alpha_0 (1 - t/T)$$

$$r(t) = r_t (1 - t/T)$$

  where t is the training epoch and T is the total number of training epochs. The neighborhood function is Gaussian.

All the experiments are conducted on a Pentium 4 3.0GHz workstation with 1 GB memory. The operating system is Windows XP Professional Edition. Note that our GeoSOM is implemented in Java and SOM_PAK is written in C. There are several benchmark tests discussing whether Java runs slower or faster than C. (Spielman 2004, Lewis 2003). Here we assume they run at almost the same speed.

We also implement a program to visualize the GeoSOM and the 2D hexagonal SOM. In this visualization, the average distances between each neuron and its direct neighbours are calculated and coded with colours. The colours are linearly changing from blue, cyan, yellow to orange. Blue denotes the smallest variance between the weight vectors, and orange the largest.

5.2 The Synthetic Dataset

We generate a dataset containing 2000 three dimensional points. The data is normally distributed with means of (0, 0, 0) and deviation of 1 in each dimension. We use a ten frequency geodesic dome as the grid lattice yielding the GeoSOM with 1002 neurons. The size of the 2D SOM is 37×27 (999 neurons). The initial update radius is 10.

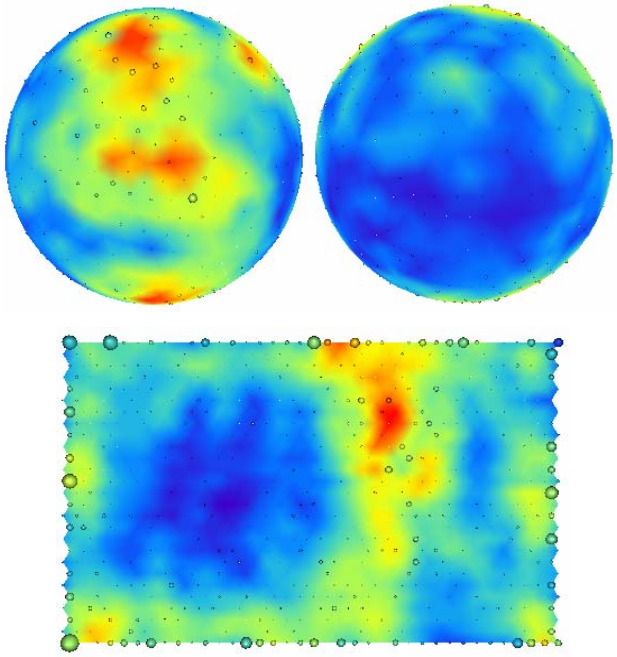

Figure10 The GeoSOM (above) and the 2D SOM (below) trained for 50 epoch.

In order to test our Error Entropy measure, we visualize the error of each neuron with spheres. Each sphere's size is determined by the amount of the error at the corresponding neuron. Figure 10 shows the GeoSOM and the 2D hexagonal SOM trained for 50 epochs. As we can see, the 2D SOM has larger errors along the boundaries while the errors on GeoSOM are distributed more evenly. This can also be observed from Figure 11. GeoSOM always has higher entropy than the 2D SOM during the training. The result shows that the Error Entropy can accurately measure the uniform degree of the error distribution.

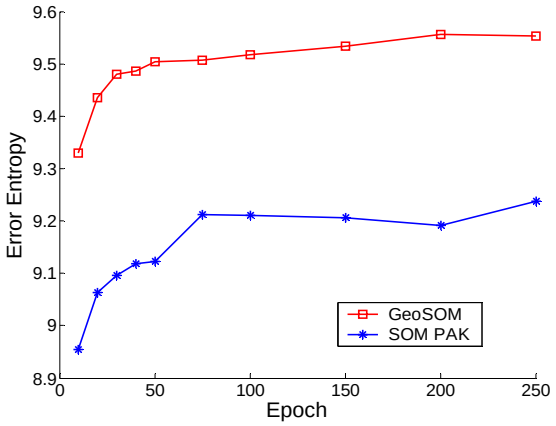

Figure 11 The error entropies of the GeoSOM and the 2D SOM at different training steps.

Figure 12 shows the quantization error and the training time at different training epochs. The GeoSOM achieve a quantization error of 0.041369915 at only 10 epochs while

the 2D SOM only approximately approached this error rate after training for 250 epochs. In terms of training time, GeoSOM is only slightly longer than SOM_PAK indicating our data structure supports neighborhood searching on the Geodesic dome as fast as on the 2D grid. In addition, the GeoSOM can achieve more accurate representation of the input data more quickly.

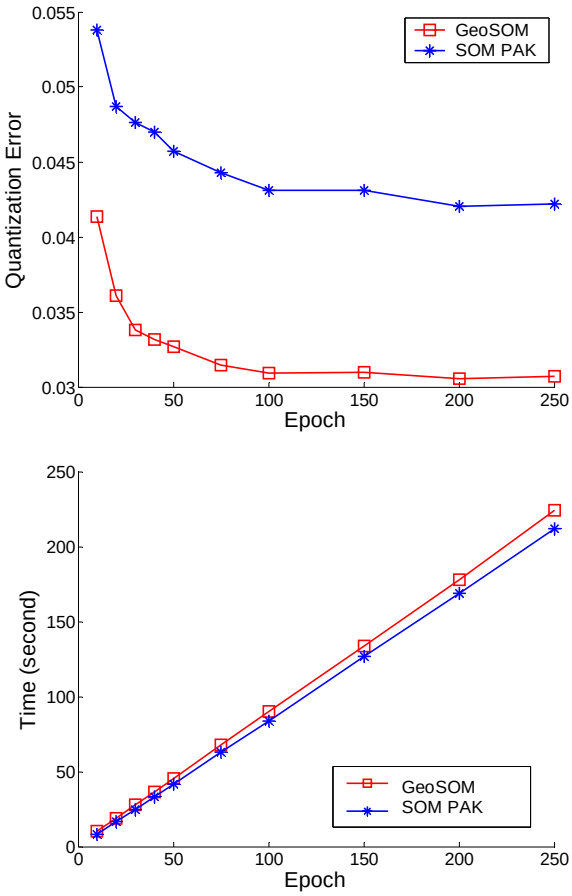

Figure 12 The quantization error (above) and the training time (below).

5.3 The Waveform Dataset

This dataset was downloaded from the UCI machine learning repository. It is also an artificial dataset which contains 5000 instances and each instance has 21 attributes. The values of the attributes are continuous between 0 and 6. The dataset is classified into 3 types of waves. Each class occupies 1/3 of the 5000 instances. This learning task is somewhat difficult as the optimal Bayes classification rate is only 86%.

We chose a 15-frequency (2252 vertices) geodesic dome for the GeoSOM. The corresponding 2D SOM is 49×46 (2254 vertices). The initial update radius was 15. In this experiment, we label each neuron according to the type of the data it won — 0, 1 or 2. If a neuron won more than one type of data, we labeled it by the majority data type. Figure 13 shows the visualization result. Due to limited space, only one of the hemispheres is shown for the GeoSOM. The approximate boundaries between the clusters are manually drawn.

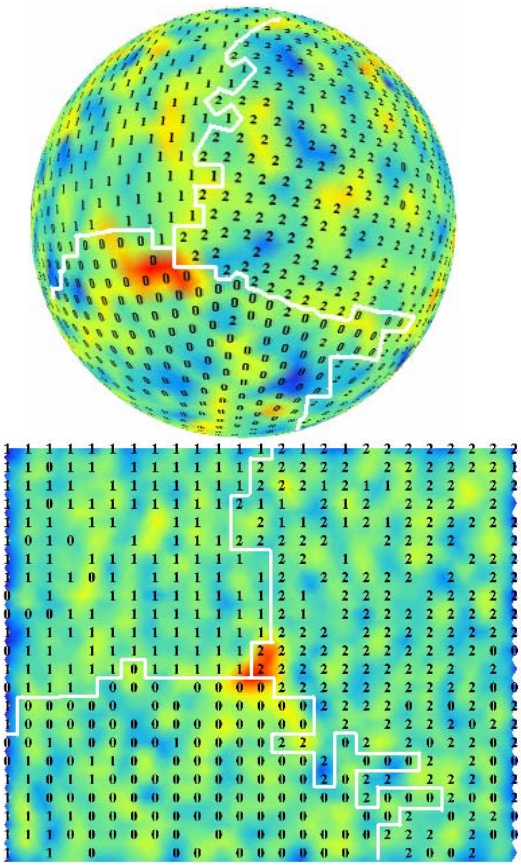

Figure 13 The GeoSOM (above) and 2D SOM (below) trained for 100 epoch with the waveform dataset.

Figure 14 shows the diagram of the error entropies during training. At the 10th epoch, both of the SOMs have the same entropy value. However, the GeoSOM dramatically raised its error entropy at the 20th epoch while the 2D SOM remain almost the same. After training for 250 epochs, the 2D SOM only had an error value of 10.68259, which is still less than the entropy of the GeoSOM at the 20th epoch.

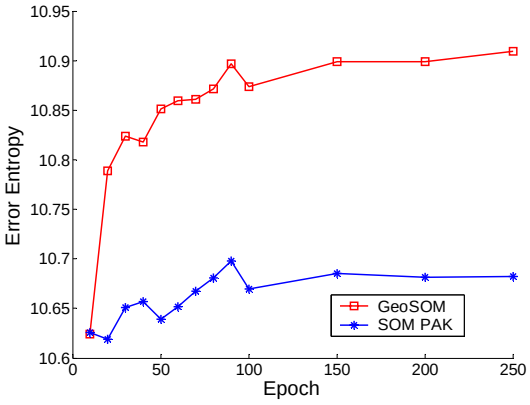

Figure 14 The Error Entropies of the GeoSOM and 2D SOM with the waveform dataset.

The changing pattern of the quantization errors is similar to that of the error entropy. The two SOMs started at almost the same error rate and then from the 20th to the 30th epoch, the GeoSOM rapidly reduced its error. After training for 250 epochs, the 2D SOM only achieved the quantization error closed to the GeoSOM at the 20th epoch.

The most significant result in this experiment is that the GeoSOM runs even faster than the 2D SOM as shown in the lower diagram of Figure 15. According to Lewis (2003), Java runs faster than C in some benchmark tests but slower in the others. So here we can only conclude that GeoSOM may have better performance when dealing with large datasets.

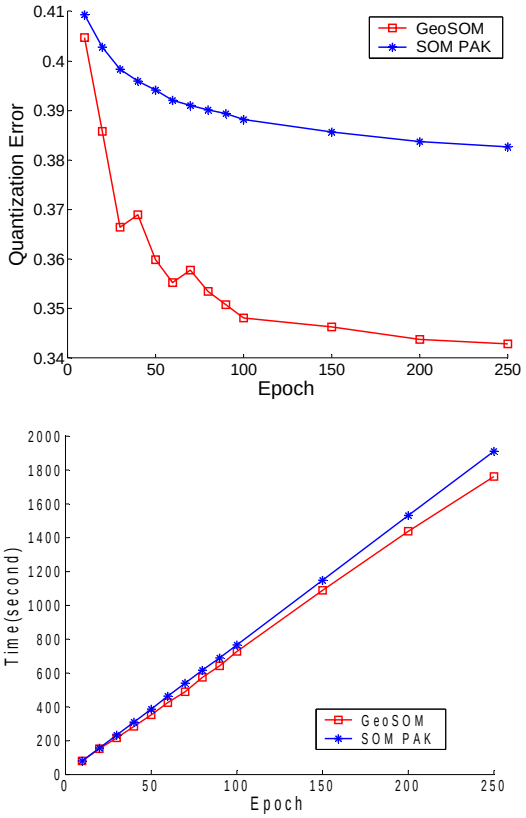

Figure 15 The quantization error and training speed of the GeoSOM and 2D SOM with the waveform dataset.

5.4 The Breast Cancer Dataset

We used a real world dataset in this experiment. The breast cancer dataset was also downloaded from the UCI machine learning repository. It contains 699 breast lump samples and each sample is described by 9 attributes (clump thickness, marginal adhesion etc.). The attributes values are integers varies from 1 to 10. The samples are classified as Benign (65.5%) and Malignant (34.5%). In visualization, the Benign samples were labeled with "2" and the other with "4". In the dataset, 16 instances have one attribute missing. We ignore the missing attribute when finding the best matching units and also in updating the weight vectors.

A six-frequency geodesic dome (362 vertices) is used for the GeoSOM and a 20×18 hexagonal grid for the 2D SOM. The initial update radius is 6.

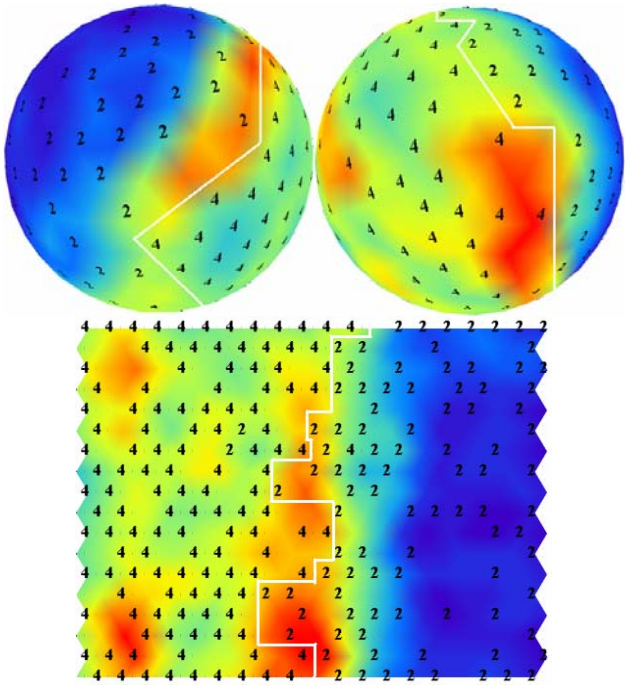

Figure 16 The GeoSOM and 2D hexagonal SOM trained for 80 epochs. The approximated boundaries of the two classes were added by hand.

From the error entropy diagram (Figure 17), we can see that the entropy values are not strictly increasing when the training goes longer. Especially for the 2D SOM, the error entropy at 90th epoch is even lower than that of 10th epoch. The fluctuation also occurs with the GeoSOM, but is more stable. We are still not sure what causes the fluctuation. It is possibly due to the instances with missing value changing their BMUs all the time. Another reason may be that the Malignant ("4") samples are quite different from each other (this can be seen from their colours on the map), so they keep changing their positions on the map. We would like to investigate this phenomenon in the future.

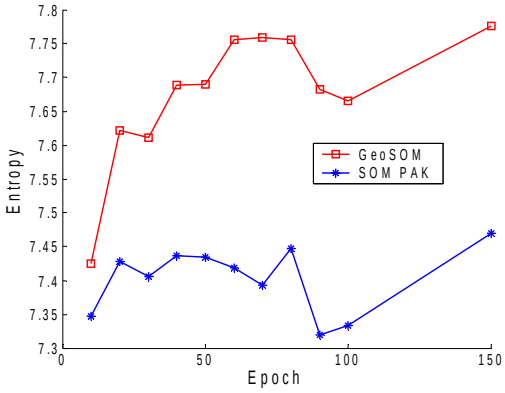

Figure 17 The error entropies of the GeoSOM and 2D SOM for the breast cancer dataset.

In the quantization error diagram, the GeoSOM once again shows it can achieve better accuracy in fewer training epochs. The extent it reduced the error rate during the training is much greater than the 2D SOM. For this relatively small dataset, the GeoSOM runs little bit slower than the SOM_PAK.

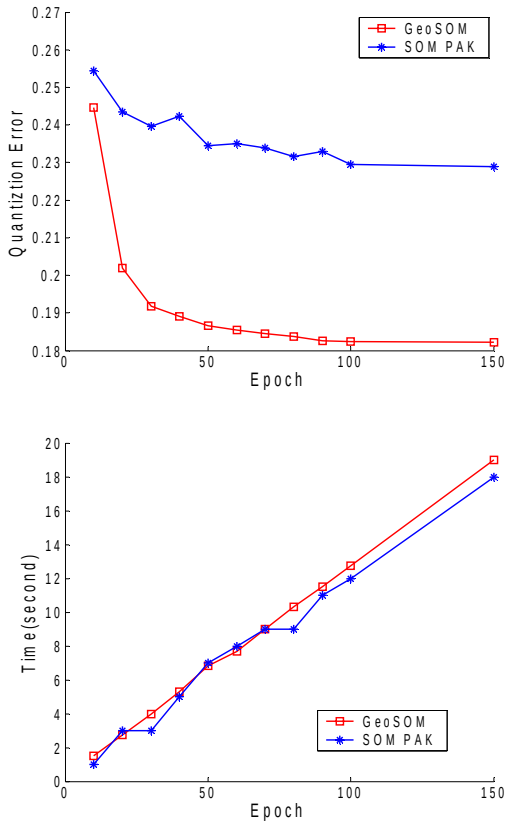

Figure 18 The quantization error and training time for the GeoSOM and 2D SOM with the breast cancer dataset.

6 Conclusion

It has been demonstrated that a spherical SOM can effectively remove the border effect of the normal 2D SOM. The use of the geodesic dome as the grid lattice has been proposed by Ritter (1999), but the lack of efficient methods for finding the neighbourhood on a geodesic dome has possibly limited the wide use of this structure. In this paper, we review our proposed data structure for the GeoSOM and compare it with the conventional 2D hexagonal SOM by experiments.

In order to quantitatively evaluate the advantage of our GeoSOM, we introduced a new measure—the Error Entropy. A high Error Entropy indicates that the error is evenly distributed on the SOM and thus is a smoother representation of the input data. As shown in the results, the GeoSOM always achieved higher Error Entropy values than the conventional 2D SOM. It also has smaller quantization error when training for the same number of epochs. Despite the more complex indexing scheme, the GeoSOM runs in almost the same speed as the 2D SOM.

The current major disadvantage of the GeoSOM is that a user needs to rotate the sphere to get a whole view of the entire picture, which is not as convenient as the traditional 2D SOM. We are planning to project the well-trained GeoSOM onto a two dimensional map. Various world maps have been developed based on the icosahedron hence those cartographic techniques would be applicable to our case. We are also seeking other statistical methods to analyse how the high dimensional data are distorted when mapping on to a spherical SOM.

7 References

Brown, J. R. et al (1995), *Visualization: Using Computer Graphics to Explore Data and Present Information*, John Wiley & Sons, Inc. NY.

Fritzke, B. (1995), Growing Grid - a self organizing network with constant neighborhood range and adaptation strength, *Neural Processing Letters*, Vol.2, No. 5, page 9-13

Fritzke, B. (1994), Growing cell structures - a self organizing network for unsupervised and supervised learning, *Neural Networks* (1994), Vol. 7, No. 9, page 1441-1460

Fuller, R. B. (1975), *Synergetics*. New York, New York: MacMillan.

Hoffmann, G. (2002),Hoffmann, G. (2002), Sphere Tessellation by Icosahedron Subdivision.

Hsiang, L.K. (2001), Mobile Robots That Learn to Navigate, honours year project report, National University of Singapore.

Kohonen, T. (2001), *Self-Organizing Maps*, Third Edition, Springer.

Lewis, J.P. (2003), Performance of Java versus C++, http://www.idiom.com/~zilla/Computer/javaCbenchmark.html , Accessed 30 Aug 2004

Lin Xia (2001), AuthorLink: Instant Author Co-Citation Mapping for Online Searching, Proceedings of National Online (New York, May 15-17, 2001) pp. 233 - 241.

Oyabu, Fukuoka and Nakatsuka, http://www2.kanazawa-it.ac.jp/oyabu/som/somint.htm, Clustering using a Self-Organizing Map. Accessed 30 August 2004

Pugh, A. (1976), *Polyhedra—a Visual Approach*, University of California Press, Ltd.

Rakhmonv, E.A, Saff, E.B. and Zhou, Y. M. (1994), Minimal discrete energy on the sphere. *Math, Res. Lett.* 1, 647-662.

Ritter, H. et. al. (1992), *Neural Computation and Self Organizing Maps*, Addison-Wesley publishing company.

Ritter, H. (1999), Self-Organizing Maps on Non-Euclidean Spaces, In: Oja E, Kaski S, editors. *Kohonen maps*. Amsterdam. The Netherlands: Elsevier BV; 1999. p.97-109

Saff, E. B. and Kuijlaars, A. B. J. (1997), Distributing Many Points on a Sphere, *The Mathematical Intelligencer*, Springer-Verlag Volume 19, Number 1.

Sahr, K., White, D. and A. Jon Kimerling (2003), Geodesic Discrete Global Grid Systems, *Cartography and Geographic Information Science*, Vol. 30, No. 2, pp. 121-134.

Sangole, A. and Knopf, G.K. (2003), Visualization of Randomly Ordered Numeric Data Sets Using Spherical Self-Organizing Feature Maps, *Computer & Graphics 27*, p963-976

Shannon, C.E. and Weaver, W. (1964), *The Mathematical Theory of Communication*, Eleventh printing.

Spielman, D.A. (2004), Speed in Matlab, C and Java, http://www-math.mit.edu/~spielman/ECC/speedTests.html, Accessed 30 Aug, 2004

Tokutaka, H. et. al. (1999), Application of Self-Organizing Map (SOM) to Chemical Data Analysis, *Journal of Surface Analysis*, vol. 5, no. 1, pp. 102—105

UCI Machine Learning Repository, http://www.ics.uci.edu/~mlearn/MLRepository.html, Accessed 30 Aug 2004

Wand, M.P., and Jones, M.C. (1995), *Kernel Smoothing,* London: Chapman & Hall.

Wu, Y.X and Takatsuka, M., "Geodesic Self-Organizing Map", accepted by the SPIE conference on Visualization and Data Analysis 2005.