Abstract— The Self-Organizing Map (SOM) is very often visualized by applying Ultsch’s Unified Distance Matrix (U-Matrix) shading and labeling the cells of the 2-D grid with training data observations nearest to that node in feature space. Although powerful and the de facto standard visualization for SOMs, this does not provide for two key pieces of information when considering real world data mining applications: (a) While the U-Matrix indicates the location of possible clusters on the map, it typically does not accurately convey the size of the underlying data population within these clusters. (b) When mapping training data observations onto the 2-D grid of the SOM it often occurs that multiple observations are mapped onto a single cell of the grid. Simply labeling the observations on a single cell does not provide any insights of the feature-space distribution of observations within that cell and in practical data mining applications it is often desirable to understand the distribution or “goodness of fit” of the observations as they are mapped to the individual SOM cells.

We address these problems with two complementary innovations. First, we increase or decrease the 2-D size of each cell according to the number of data elements it contains; an approach derived from the cartogram techniques in geography. Second, we determine the within-cell location of each datum according to its similarity in n-dimensional feature space to each of the neighboring nodes that surround it on the 2-D SOM grid. When multiple observations are mapped to a single cell then the plot locations will convey a sense of the data distribution within that cell. One way to view plotting of the data distribution within a cell is as a visualization of the quantization error of the map. Finally, we found that these techniques lend themselves to additional applications and uses within the context of SOMs and we will explore them briefly.

Keywords- Self organizing feature maps; Data visualization; Data mining; cartogram

I. INTRODUCTION

Kohonen’s self-organizing maps (SOM) [1] employ an artificial neural network to reduce the dimensionality of an \( \mathbb{R}^n \) dataset while preserving the topology of its data relationships. The SOM network is typically constructed and visualized as a regular two-dimensional grid of cells, each representing a single node. Thus, each node has both a feature-space \( \mathbb{R}^n \) value and a 2-D \((x,y)\) position visualized as a cell in the SOM grid. (When we speak of the neighbors of a node, we mean those nodes whose 2-D grid positions are adjacent to that of the indicated node.)

During training, adjustments to each node’s \( n \)-dimensional values are also partially applied to nodes found within a decreasing radius of its 2-D grid position. Thus, changes in feature-space values are smoothed, forming clusters of similar values within the local neighborhood on the 2-D grid.

Clustering is often indicated by shading each cell to indicate the average distance in feature-space of the node to its 2-D grid neighbors; this is the Unified Distance Matrix (U-Matrix) [2]. To map data to this grid, the node nearest to each datum is identified and the data is plotted in the grid cell of that node [3]; often, multiple data map to the same cell in the grid.

This standard visualization of the SOM is a powerful tool for gaining understanding of the overall structure of a dataset, but it can obscure important information about individual data. It does not reliably show the size of the underlying data population within the clusters. The labeling of cells with their data does not provide any insight into the feature-space distribution of observations within that cell.

To remedy the cluster size representation problem, we expand and contract the 2-D SOM grid cells in proportion to the number of data points plotted in each. This shows clusters in proportion to their population and it also opens up space within the more populous cells for plotting the data more informatively. The resulting plot is called a cartogram and is a technique borrowed from geography. To visualize the data distribution within each cell, we need to show the feature-space separation between each datum and the node. Data that are most similar to the node (in feature space) appear at or near the center of the 2-D grid cell. Data that are less similar to the node are moved toward the grid neighbors with which they have the most similarity in feature space. The spread of the data around the center of the cell also indicates the quantization error.

A comparison of the standard visualization and our enhancements is shown in Figure 1. On the left we show a standard U-Matrix visualization of the familiar iris data set [4], [5] and on the right we show our enhanced visualization of the same SOM using the cartogram and the visualization of the quantization error.
II. CARTOGRAM

A cartogram is a geographical map that has been distorted so that the area occupied by each region of the map corresponds to the value of some parameter related to that region. For example, countries of the world might be shown scaled in proportion to their population, per-capita income, or any other metric of interest. In this visualization, we have adopted the diffusion method of Gastner and Newman [6].

The goal of a cartogram is to reshape the features of the map so that the average density of the metric of interest – e.g., population – is uniform throughout the map. The diffusion cartogram achieves this by calculating the density gradient at each vertex in the map and moving the vertices along the gradient toward the less-dense area. Areas where the metric is greater than the average are expanded (and so made less dense) and areas where the metric is less than average are reduced in size (increasing their density). This process repeats until the reshaped map stabilizes: all areas are at the average density and so the gradients are zero. (A more complete, two-page description of the algorithm in pseudo-code is available online as a “Supporting Text” to the original paper [7].)

To construct a cartogram for our visualization, we start with the hexagons or rectangles that outline each of the 2-D cells of the SOM map. The initial density within these polygons is calculated as a function of the number of data points mapped to that cell.

Using the ‘sp’ (Spatial Polygons) package [8], we transfer our original map of polygons – hexagons or rectangles – and their population density values to a fine rectangular mesh that can be processed efficiently by the ‘Rcartogram’ package [9] – an interface to Newman’s implementation of the diffusion cartogram algorithm in C [10]. The “cart” object returned is used to translate grid polygon points to their new positions on the cartogram visualization.

III. DATA MAPPING WITHIN THE CELL

To position the each datum within the cartogram-expanded data cell, we begin (as do other visualizations) by selecting the nearest node in feature-space; the datum will appear within that node’s cell. Then:

- a feature-space vector from that nearest node to each of its neighbors is calculated,
- the relative length of the orthogonal projection of the datum along each neighbor vector is calculated in feature-space, and
- a 2-D offset vector is calculated and added to the nearest node’s position in the 2-D grid.

The resulting location meaningfully and consistently places the datum on the map visualization.

A. Selecting the Nearest Node

The datum to be plotted \((x)\) is compared to each node’s feature-space value \((m_i)\) using some metric such as the least Euclidian distance (Equation 1) [11]:

\[
\sum (m_i - x)^2 = \text{min} \sum (m_i - x)^2 \quad \text{or} \quad c = \text{argmin} \sum (m_i - x)^2.
\]

Node \(c\) is the node nearest to the datum in feature space.

B. Finding Vectors to Neighbors

The feature-space vectors to each of the \(j\) neighbors \((m_j)\) are calculated simply by subtracting the \(k\)th feature-space value of the nearest node, \(m_c\), from that of each neighbor \(m_j\) (a linear translation, Equation 2):

\[
m_j' = m_j - m_c
\]

Likewise, the datum’s translated vector is (Equation 3):

\[
x' = x - m_c
\]

Applying the same translation to the nearest node itself confirms its role as the origin for the calculations that follow (its value is 0 in every coordinate axis):

\[
0 = m_c - m_c
\]

C. Orthogonal Projection

To determine how far a datum should shift from its nearest node toward each neighbor node, we consider its
orthogonal projection onto each neighbor vector in the \( n \)-dimensional feature space of the SOM. The translated datum vector \( x' \) can be thought of as the sum of two component vectors: one \( (x'_1) \) directly along the vector to the neighbor node \( (m'_j) \) and the other at right angles to the first. The vector \( x'' \) is the orthogonal projection of the datum vector onto the neighbor node vector. As shown in Figure 1, the orthogonal projection is equal to the product of some scalar value \( \alpha_j \) and the translated neighbor vector.

This \( \alpha_j \) value (proportional projection toward the neighbor) is found using the dot product (inner product) of vectors [12]:

\[
\alpha_j = \frac{x'_j \cdot m'_j}{x'_j \cdot x'_j}
\]

If the datum is on the “other side” of the origin (headed away from a neighbor), the value of \( \alpha_j \) will be negative. Neighbors with positive \( \alpha \) values will “pull” the datum in their direction on the grid while neighbors with negative \( \alpha \) “push” the datum away.

D. Calculate and scale the 2-D Offset

Again taking the center of nearest node \( c \) as the origin (this time in 2-D grid space: \( g_c \)), the translated grid coordinates of each neighbor \( g_i \) are multiplied by the proportional length \( \alpha_j \) and added together to form a raw 2-D offset vector \( r \):

\[
\vec{r} = \sum_{i=1}^{n} \alpha_j \vec{r}_j
\]

(6)

Typically, several neighbors contribute to this raw offset, exaggerating the datum’s distance from its nearest node. For example, if the neighbor to the left has a positive \( \alpha \), pulling the datum to the left, the neighbor to the right might very well have a negative \( \alpha \) and push the datum even further to the left. Diagonal neighbors can push or pull along both axes.

If the raw offset is used, the datum will frequently appear outside the area of its nearest node’s cell; this incorrectly suggests that some other node is nearest. We have found the simplest satisfactory scaling function is to divide the raw offset by the number of neighbors surrounding the nearest node:

\[
s = r \div \| \{ \text{neighbors} \} \|
\]

(7)

There is an aesthetic and practical tension between ensuring that data are displayed within the area of their nearest nodes while not limiting offsets to a range too small to be perceptible. Alternate approaches to scaling are possible; one is described in the “Examples and Experiments” section below.

Finally, the scaled offset \( s \) is added to the 2-D coordinates of the nearest node \( (g_c) \), giving the plotted grid position of the datum, \( g_d \):

\[
g_d = g_c + s
\]

(8)

E. Visual representation

An appropriate symbol or label is drawn at the position \( g_d \) calculated in Equation 8. We also add a thin line connecting each symbol back to the center of its cell, \( g_c \). This visually reinforces the interpretation of the plotted position as a vector from the nearest node toward neighbors.

On occasion, the cartogram reshaping of the map can produce cell outlines where the true center is not immediately obvious. An example may be found in the second-to-last cell in the bottom-right corner of Figure 1. Despite the distortion of the shape of the cell, one can perceive that the datum plotted there is at the center because the connecting line has length zero and so disappears.

Intuitively, all data should appear somewhere within the grid cell representing their nearest node. If a map has very high quantization errors or too extreme a scaling function is used (see experiments below), data might be pushed into adjacent grid cells. The connecting line makes this immediately evident; without it, the data might be seen as belonging to the wrong cell.

IV. Examples

We selected the very-well-known Fisher/Anderson “Iris” data [4], [5] to demonstrate this visualization. This dataset is included in R’s built-in datasets package. Plot symbols are assigned to the three iris species: square=\textit{setosa}; circle=\textit{versicolor}; and triangle=\textit{virginica}.

A. Cluster Population Size

Each of the three iris species is observed 50 times. In Figure 1, we see that the \textit{versicolor} species appears in 28 of the 98 grid cells, \textit{virginica} in 30, and \textit{setosa} in only 23. (19 cells are empty.) This under-representation of \textit{setosa} (square) is corrected for by the cartogram expansion of cells in which more data appear.

B. Quality and Quantization Error

The average distance in the \( \mathbb{R}^n \) feature space between each training datum and its nearest node measures the overall fit of the map to the data. Maps which minimize this average quantization error are to be preferred [11]. In Figure 3, we
have deliberately created a poor quality SOM with high quantization error by limiting the number of training iterations. Despite this, the standard visualization is almost indistinguishable from that of Figure 1. One subtle indication is that there are more empty cells, but that is not very informative. Our visualization in Figure 3 clearly shows the high quantization error with numerous data pushed to the very edges of their cells. The cartogram expansion also emphasizes the contrast between over-full and empty cells.

V. EXPERIMENTAL SCALING MODIFICATIONS

The distance at which a datum is plotted from the center of its nearest node’s grid is a critical aspect of this visualization. Data should be displayed in a range around their nearest nodes that is neither too large nor too small. Where the 2-D offsets are larger than the cell radius (±0.5), the data will appear outside the borders of the grid cell for their nearest node; this incorrectly implies that a different node is nearest. However, if the spread is too small, all data will converge about the center of their grid cells and the advantages of this visualization are lost.

We have experimented with two complementary approaches to scaling: the first scales the individual contribution of each neighbor node; the second applies an exponential scaling to all data offsets.

A. Projection Component Scaling

The offsets calculated in Equation 5 need to be scaled to produce an appropriate and intuitive representation of the data. The simplest approach of Equation 7 is to simply divide the raw offset by the number of neighboring cells. It is also possible to scale each component projection to neighboring nodes individually. We can intuit that the farther the datum is from the neighboring node, the less important its calculated projection toward that node. A simple measure of this importance is the ratio of the distance between the datum \(x\) and the nearest node \(m_j\) to the distance between the datum and the neighbor node \(m_i\). So, we extend Equation 5 to include the ratio of these distances (Equation 9):

\[
\frac{\square}{\square} = \frac{\square \cdot \square - \square \cdot \square}{\square \cdot \square - \square \cdot \square}
\]

This distance ratio multiplier will be \(\leq 1\) because we know that the datum cannot be closer to the neighbor node \(j\) than to its nearest node \(c\). As a datum approaches its nearest node in feature space, this ratio approaches zero, locking data to the center of the cell.

In practice, this scaling seems to be particularly effective at rejecting large “pushes” from strongly dissimilar neighbors and there seems to be less scaling discontinuity between the center and the edges of the map. With no further scaling applied (replacing Equation 7 simply with \(s=r\)), this produces a spread fairly similar to that resulting to the simpler neighbor-count method. These are compared in Figure 4, using a smaller SOM to force more iris observations into each cell.

B. Exponential grid offset scaling

As data move away from the center of a grid cell, it is easier to perceive multiple data points and their relationships simply because there is more room. Expanding the calculated grid offsets can thereby aid in visual data exploration. However, we still want to keep large offsets within the boundaries of the grid cell.

As seen in Figure 5, an exponential function “squeezes” all values closer to 1.0. Grid cells are ±1.0 units apart, so to stay within boundaries, we want to squeeze offset values should be \(\leq 0.5\). Equation 12 combines an exponential scaling parameter \(a\), a linear scaling parameter \(b\), and centers the “squeeze” at \(c\):

\[
\begin{align*}
\text{Figure 4.} & \quad \text{Exponential scaling conditions values toward 1.0 to a varying degree according to the exponential parameter (a in Equation 12).} \\
\end{align*}
\]
VI. CARTOGRAM VARIATIONS

Figure 5. Comparison of scaling methods. Above (a), the simplest scaling divides the composite offset by the number of neighbors of that cell. In the middle (b), each offset component is individually scaled according to the proportional distance of the datum to the nodes. Below (c), the exponential offset scaling applied to Figure 4b spreads out data plotted near the center of each cell. The scaling parameters (Equation 12) are set to $a=0.5$, $b=0.4$, $c=0.3$. (Shown without U-Matrix shading or cartogram expansion for clarity.)

This scaling function operates independently on the $x$ and $y$ components of the raw 2-D grid offset vector ($r$) to produce the scaled offset vector ($s$). The exponential function is undefined for negative, non-integer values of $r$ (where $a$ is not a natural number), so we take its absolute value and preserve its sign. Equation 12 can replace the earlier scaling method of dividing by the number of neighbors (Equation 7).

An example of this magnification is shown in Figure 4c. This scaling method provides additional (but optional) control of the spread of data within the cell and can be adjusted experimentally to facilitate data exploration.

Figure 6. This SOM cartogram scaling inverts the U-Matrix distance to expand cluster areas. Compare to Figure 1.

Figure 7. This SOM cartogram emphasizes cells in which more than one species of iris are mapped. Compare to Figure 1.

This SOM cartogram scaling inverts the U-Matrix distance to expand cluster areas. Compare to Figure 1.

VI. CARTOGRAM VARIATIONS

In Figures 1 and 3, the cartogram scaling parameter is the number of data in the cell – the data density of the map. It is possible to use other aspects of the data map such as the U-Matrix distance or the number of classes found within a cell.

In Figure 6, areas where the U-Matrix distances are low – i.e., areas within clusters [2] – are expanded. This produces an effect similar a contour map or wireframe model where the clusters appear to form “hills” separated by “valleys.” While the standard U-Matrix shading is still shown for reference, it is redundant and could be replaced with some other shading such as a color code for class.

The cartogram technique can also be utilized to draw attention to areas of interest, treating the plot as a sort of 3-dimensional pliable surface [13] and pulling the areas of interest “toward” the viewer. The iris data includes three classes (species): one is easily separated, but the other two overlap. In Figure 7, the SOM cells showing this overlap are expanded and so drawn to our attention.

VII. RELATED WORK

Vesanto [14] demonstrated two methods for showing the quantization error in a SOM. In one, the SOM map is tilted back into a 3-D perspective and bars corresponding to the quantization error project upward. For the second, a circle whose area corresponds to the quantization error surrounds the grid cell. These approaches seem to be appropriate when only a few cells are of interest. They do not lend themselves to seeing the quantization error for the entire map: the bars and circles would obscure each other. Some existing packages will shade the grid cells according to the...
quantization error (for example, the “quality” map of the ‘kohonen’ package [15]), but this prevents using shading to show the U-Matrix feature-space distances between cells.

Vesanto [14] also offers methods to visualize the number of data within a cell, potentially helping to understand the size of a cluster. In one, the cell is progressively filled from the center, such that the area of the shape in the center of the cell corresponds to the number of data. The second projects bars upward, making a sort of 3-D histogram. The third “scatters” the data (much like the reverence visualizations shown in Figure 1 and Figure 3), randomly placing one dot per data sample in the cell. None of these offer any information about the quantization error.

In the Emergent Self-Organizing Map [16], [17], Ultsch takes the U*-Matrix – a combination of distance and density – as a height value for the grid. This is shown in 2-D as a sort of topographic contour map that is also colored as in a geographic map (from blue seas to white peaks); the image is also rendered in 3-D. Boundaries are shown as “mountain ranges” separating the “valleys” of the clusters themselves. In our visualization, the cartogram technique, can also present a sort of 3-D appearance as seen in Figure 6 and Figure 7. The reshaped edges of the polygons are strongly reminiscent of contour lines on a topographic map.

We have not found any work using a cartogram to aid in the visualization of a SOM, though we did find one paper where the SOM helps to construct a cartogram [18].

VIII. IMPLEMENTATION IN R

A. Existing packages

Various packages for training self-organizing maps are available through the Comprehensive R Archive Network (CRAN) [19]. These include ‘class’ [20], ‘RSNNS’ [21], ‘som’ [22] and ‘kohonen’ [15]. A few other packages offer application-specific visualizations of SOM objects.

Of these, only the ‘kohonen’ package includes a data mapping visualization. It positions the plotted locations of data points using a randomized normal distribution about the center of the cell, as seen in the “standard” visualization shown in Figure 1 and Figure 3.

B. Our implementation

Our implementation of this visualization is part of a larger package of SOM visualization and manipulation methods for the R system [23] currently in development. Key features include:

• an S4 adapter class provides a common interface to allow use of SOM objects created by other packages such as ‘som’[22] and ‘kohonen’[15]
• visualizations use the ‘grid’ graphics system [24] to facilitate subsequent manipulation and reuse
• support for both square and hexagonal maps
• GPL license

The visualization functions include the capability to construct and display a variety features, including control of:

• cell background shading (i.e., to show the U-Matrix, quantization error, or some other measure),
• individual cell borders,
• the outer borders of contiguous groups of cells (i.e., for outlining clusters),
• the connected components of the map [25] (enhances the display of clusters seen with the U-Matrix),
• data mapping onto the grid and within grid cells, and
• cartogram expansion of the grid.

We anticipate releasing this software through R-forge [26] early in 2012.

IX. CONCLUSION AND FUTURE WORK

With this visualization, we have overcome two limitations of the standard SOM visualization. Application of the density diffusion cartogram to the U-Matrix scales clusters in proportion to their population. Information about the quantization error and structure of data mapped to individual cells is revealed by positioning each datum according to its similarity to neighbors.

Some further work is warranted to refine the method of scaling the 2-D data offsets. We plan to investigate whether the number of neighbors needs to be considered to help ensure that the appearance at edges is consistent with the interior. Support for toroidal SOMs (where the edges of the maps are joined) in the data position calculation would extend this technique to a few more applications.

The cartogram’s adaptability to represent any of a variety of different aspects of the data is quite intriguing. It nicely complements shading/coloring and labeling, adding another layer of information to the SOM visualization without overwhelming our ability to understand the image. We intend to investigate additional ways to use this capability in data mining and exploration.

ACKNOWLEDGMENT

(Acknowledgements to be added to camera ready version upon acceptance.)

REFERENCES


