Cartogram Data Projection for Self-Organizing Maps

David H. Brown and Lutz Hamel
Dept. of Computer Science and Statistics
University of Rhode Island
USA
Email: david_h_brown@my.uri.edu or hamel@cs.uri.edu

Abstract— **Self-Organizing Maps (SOMs) are often visualized by applying Ultsch's Unified Distance Matrix (U-Matrix) and labeling the cells of the 2-D grid with training data observations. 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. However, 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 shortcomings with two complementary visualizations. 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 cartogram techniques in geography. Second, we determine the within-cell location of each mapped training observation according to its similarity in *n*-dimensional feature space to each of the immediate neighbor 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 $\Re^n$ 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 $\Re^n$ 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.)

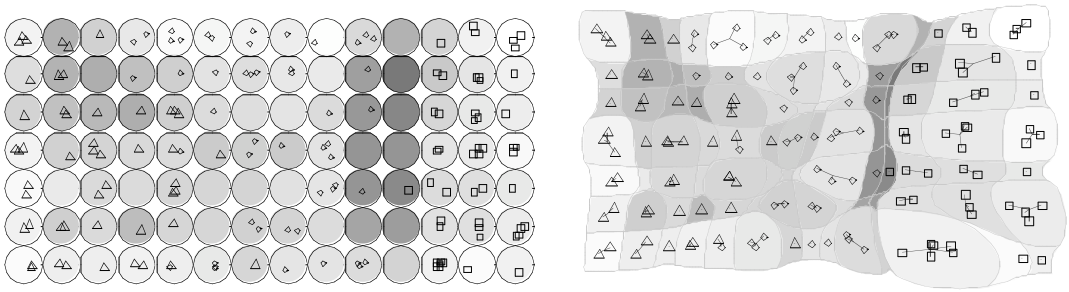
During training, adjustments to each node's n-dimensional values are also partially applied to nodes found within a time step sensitive radius of its 2-D grid position. Thus, changes in feature-space values are smoothed, forming clusters of similar values within the local neighborhoods on the 2-D grid.

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 training data to this grid, the node nearest in feature-space to a training observation is identified. This is the "best-match" node for that observation and the observation is plotted in the grid cell of that node [3]. This is done for all training instances. Often, multiple observations 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 straightforward labeling of cells with their data does not provide any insight into the feature-space distribution of the data 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 [4]. To visualize the data distribution within each cell, we show the feature-space separation between each observation and its corresponding node on the grid. Data points that are most similar to the node (in feature space) appear at or near the center of the 2-D grid cell. Data points that are less similar to the node are moved toward the grid neighbors, which they are most similar to in feature space. The spread of the data around the center of the cell also indicates the quantization error. The quantization error is a measure of "goodness of fit" and is defined as the feature space distance between a training data point and its best matching node on the map [1].

A comparison of the standard visualization and our enhancements is shown in Figure 1. In Figure 1(a) we show a standard U-Matrix visualization of the familiar iris data set [5][6] and in Figure 1(b) we show our enhanced visualization of the same SOM using the cartogram and the visualization of the quantization error. It is easy to see that the maps have three main clusters. In the map in Figure 1(b) the size of the



(a) (b)

Figure 1. (a) The typical SOM visualization combining U-Matrix shading with randomly assigned data positions within the cell. (b) Our enhancements. The SOM has been constructed from the Fisher/Anderson "iris" data set; darker shading indicates cluster boundaries (greater feature-space distance between cells). Plot symbols correspond to iris species (class). The linearly separable *setosa* species is on the right (square symbols).

cells have been resized in the proportion of data points mapped to the cells. The data positions within the cells of the map in Figure 1(a) are random jitter and therefore contain no information. In our visualization, Figure 1(b) the within cell positions are meaningful and are computed from the data.

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 our cartogram visualization, we have adopted the diffusion method of Gastner and Newman [7].

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 [8].)

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 [9][10], 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 [11] – an interface to Newman's implementation of the diffusion cartogram algorithm in C [12]. 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 training observation within the cartogram-expanded cell, we begin (as do other visualizations) by selecting its best-match node in feature-space; the observation will appear within that node's cell. Then:

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

The resulting location meaningfully and consistently places the observation on the map.

A. Selecting the Best-Match node

The data point to be plotted (x) is compared to each node's feature-space value (m_i) using some metric such as the least Euclidian distance [13],

$$\|x - m_b\| = \min_i \{\|x - m_i\|\}, \text{ or}$$
$$b = \arg\min_i \{\|x - m_i\|\} \quad (1)$$

Node b is the node nearest to the data point in feature space: the best-match node.

B. Finding Vectors to Neighbors

The feature-space vectors to each of the j neighbors (m_j') are calculated simply by subtracting the $\Re^n$ feature-space value of the best-match node, m_b, from that of each neighbor m_j (a linear translation,),

$$m_j' = m_j - m_b \quad (2)$$

Likewise, the data point's translated vector is,

$$x' = x - m_b \quad (3)$$

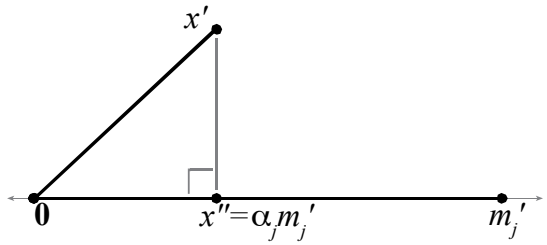


Figure 2. Orthogonal projection (x"') of a training instance vector (x') onto a neighboring node vector (m_j'). These calculation is generalizable to the $\Re^n$ feature space of the SOM [8].

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

$$\mathbf{0} = m_b - m_b \tag{4}$$

C. Orthogonal Projection

In order to determine how far a data point should shift from its best-match 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 data point vector (x') can be thought of as the sum of two component vectors: one (x'') 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 data point vector onto the neighbor node vector. As shown in Figure 2, the orthogonal projection is equal to the product of some scalar value α_j and the translated neighbor vector.

This α_j value (proportional projection toward the neighbor) is found using the dot product (inner product) of vectors [14],

$$\alpha_j = \frac{x' \cdot m_j'}{m_j' \cdot m_j'} \tag{5}$$

If the data point is on the "other side" of the origin (headed away from a neighbor), the value of α_j will be negative. Neighbors with positive α values will "pull" the data point in their direction on the grid while neighbors with negative α "push" the data point away.

D. Calculate and scale the 2-D Offset

Again taking the center of best-match node b as the origin (this time in 2-D grid space: g_b), the translated grid coordinates of each neighbor g_j are multiplied by the proportional length α_j and added together to form a raw 2-D offset vector r,

$$r = \sum_{j \in \{\text{neighbors}\}} \alpha_j \left(g_j - g_b \right) \tag{6}$$

Typically, several neighbors contribute to this raw offset, exaggerating the data point's distance from its best-match node. For example, if the neighbor to the left has a positive α, pulling the data point to the left, the neighbor to the right might very well have a negative α and push the data point even further to the left. Diagonal neighbors can push or pull along both axes.

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

$$s = r \div \|\{\text{neighbors}\}\| \tag{7}$$

There is an aesthetic and practical tension between ensuring that data points are displayed within the area of their best-match nodes while not limiting offsets to a range too small to be perceptible. Alternate approaches to scaling are possible and continue to be explored but this simple scaling scheme described here seems to work appropriately.

Finally, the scaled offset s is added to the 2-D coordinates of the best-match node (g_b), giving the plotted grid position of the datum, g_d,

$$g_d = g_b + s \tag{8}$$

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_b. This visually reinforces the interpretation of the plotted position as a vector with respect to the best-match node.

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 our cartogram map in Figure 1(b). Despite the distortion of the shape of the cell, one can perceive that the data point plotted there is at the center because the connecting line has length zero and so disappears.

Intuitively, all data points should appear somewhere within the grid cell representing their best-match node. If a map has very high quantization errors, data points might be pushed into adjacent grid cells. The connecting line makes this immediately evident; without it, the data points might be seen as belonging to the wrong cells.

IV. Examples

For our first experiment we selected the very-well-known Fisher/Anderson "Iris" data [5][6] to demonstrate this visualization. This dataset is included in R's built-in datasets package. Plot symbols are assigned to three iris species: square=*setosa*; circle=*versicolor*; and triangle=*virginica*.

A. Cluster Population Size

Each of the three iris species is observed 50 times. In Figure 1, we see that the *versicolor* species appears in 28 of

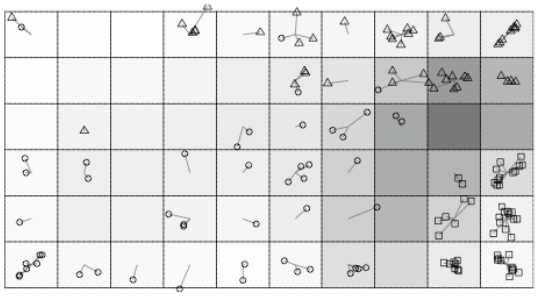


Figure 3. Visualization of the quantization error of a poorly trained SOM.

the 98 grid cells, *virginica* in 30, and *setosa* in only 23. (19 cells are empty.) In the cartogram representation of the map the cells have been rescaled according to the number of data points mapped to each cell of the SOM.

B. Quality and Quantization Error

The average distance in the $\Re^n$ feature space between each training observation and its best-match node measures the overall fit of the map to the data. Maps which minimize this average quantization error are to be preferred [13]. 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 U-Matrix visualization is almost indistinguishable from the map in Figure 1(a). 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 and over the edges of their respective cells. In order to emphasize the quantization error we did not apply the cartogram cell expansions in this particular visualization.

V. CARTOGRAM VARIATIONS

In Figure 1(b), the cartogram scaling parameter is the number of data points mapped to a cell – the data density of the map. It is possible to use other metrics such as the U-Matrix distance or the number of classes found within a cell as scaling parameters for cartogram visualizations.

In Figure 4, we used the as the U-Matrix distances as a scaling parameter: areas where the U-Matrix distances are low – i.e., areas within clusters [2] – are expanded, area where the U-Matrix distances are high are contracted. This

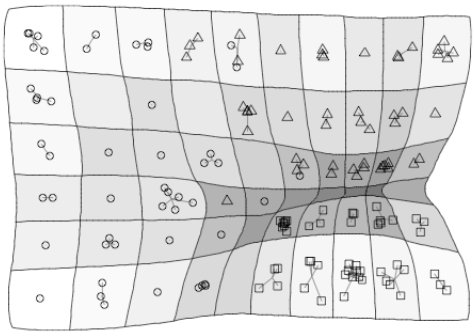


Figure 4. A visualization based on the unified node distance as the cartogram parameter.

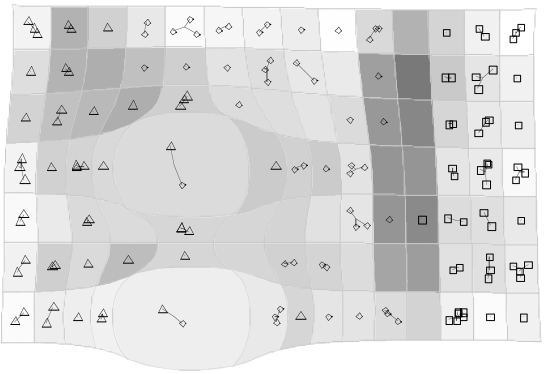


Figure 5. This SOM cartogram emphasizes cells in which more than one species of iris are mapped.

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 [15] 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 5, the SOM cells showing this overlap are expanded and so drawn to our attention.

VI. ADDITIONAL EXPERIMENTS

The iris data set is useful for initial exploration of new techniques, but with only 150 observations and 4 attributes, it is not a good representative of the very large, very-high-dimensional data sets often encountered in real-world settings. The UCI Machine Learning Repository [16] provides a variety of more extensive data sets.

We selected the Cardiotocography [17] data set for further experimentation. A cardiotocogram is a recording of both uterine contractions and fetal heartbeat [18] used in obstetrics. This data set contains results of 2126 examinations each with 21 measured real- or integer-valued attributes plus two additional classification attributes assigned by the consensus of three expert obstetricians. One classification attribute indicates one of ten classes of events being observed such as "calm sleep" or "decelerative pattern." The second classification attribute is a risk

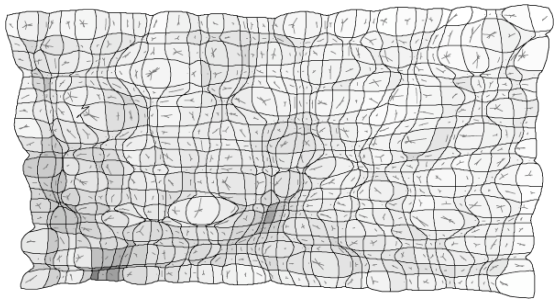


Figure 6. Neither the U-Matrix distances (gray levels) nor data density (cell size) provide insight into the map of cardiotocography data set.

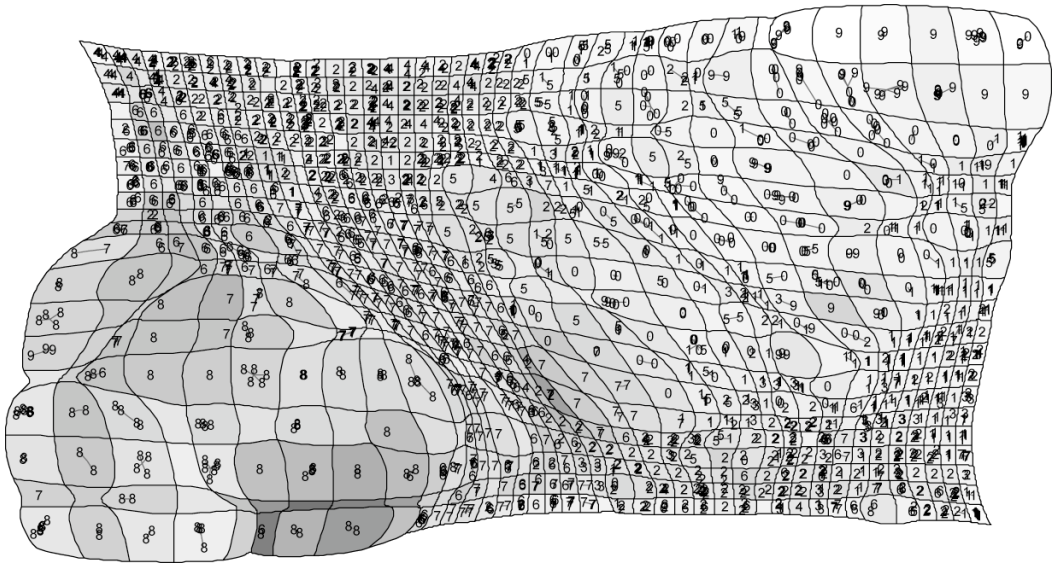


Figure 7. SOM of the cardiotocography data set using the expert assessment of risk as cell size.

measure: "normal," "suspect," or "pathologic."

After normalizing the 26 measured attributes with R's scale function, we constructed a 40×20 rectangular SOM. The expert interpretations (risk and category) were not used to train the SOM. Neither the U-Matrix visualization nor the data density based cartogram of this SOM showed any clear pattern of clustering (Figure 6). There were two pockets of higher U-Matrix distances but no clear divisions between regions. A data density cartogram expansion was not very helpful because the training data are evenly distributed through the map, with cell densities ranging from 0 to 10 with a mean around 2.6 and standard deviation about 1.8.

We obtained a much more interesting plot when we used the cartogram expansion to show the average risk classification (1=normal; 2=suspect; 3=pathologic) of the examination observations mapping to a given cell (Figure 7). This risk assessment was not used to train the SOM, but it seems to correspond to a particular region in the data-space which maps to the lower-left corner of the SOM. The cartogram expansion of that high-risk area gives us more room to see data markers (omitted in Figure 6); the high-risk region is dominated by the codes "largely decelerative" (code 8) and "decelerative" (code 7) of the fetal state classes. As with the risk assessment, the state codes were not used to train the SOM.

Simple color coding would, of course, be able to show the high-risk region, but it would hide the U-Matrix and would not facilitate the closer examination of the data allowed by the cartogram expansion.

VII. Related Work

Vesanto [19] 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 'kohnen' package [20]), but this prevents using shading to show the U-Matrix feature-space distances between cells.

Vesanto [19] 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 visualization shown in Figure 1(a)), 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 [21][22], 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 4. 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 [23].

VIII. Implementation in R

A. Existing packages

Various packages for training self-organizing maps are available through the Comprehensive R Archive Network

(CRAN) [24]. These include 'class' [25], 'RSNNS' [26], 'som' [27] and 'kohonen' [20]. 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(a).

B. *Our implementation*

Our implementation of the cartogram visualization is part of a larger package of SOM visualization and manipulation methods for the R system [28] 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'[27] and 'kohonen'[20]
- visualizations use the 'grid' graphics system [29] 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 [30] (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 [31] in mid-2012.

IX. Computational Complexity

In practice, we find that the calculation of the SOM itself requires far more effort than calculating the data projection within the cell or the cartogram. There are many variations on the basic SOM algorithm; an examination of the implementation in the kohonen package of R shows it to be $O(i \times d \times m \times n)$ where i is the number of iterations over which to train the map; d is the number of observations in the training set; m is the number of nodes in the map; and n is the number of dimensions in the feature space.

At the conclusion of SOM training, the best-match node for each training observation is known and may be saved. Otherwise, finding best-match nodes is a $O(m \times d \times n)$ operation as each node in the map must be examined for each observation. With this information, to locate a single observation within its cell requires a small handful of operations in $\Re^n$ space; it need never consider more than 8 neighbors, so we may say the time required is $O(n)$ for a single observation or $O(d \times n)$ for a set of data.

Time required process the cartogram is dominated by a Fast Fourier transform: $O(c \log c)$ where c is the number of intersections in the rectangular mesh upon which the cartogram is calculated [7]. We have found that meshes as coarse as 128×128 give acceptable results, though we usually use 256×256. The runtime of the 'sp' package's "overlay" method is a product of the number of points (intersections in the cartogram mesh, c) and the number of polygon edges (a small multiple of the number of nodes in our map). Thus in our use of it, the complexity of "overlay" is $O(c \times m)$.

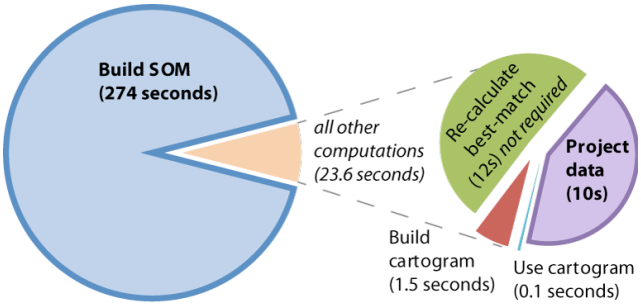


Figure 8. User-time to perform calculations for Figure 7.

To quickly confirm our expectations and experience, we used R's system.time() function to display the user-time taken for several of the main computational steps required to produce Figure 7. Figure 8 shows how the SOM calculation itself (274 seconds) far exceeded the time required for other calculations (23.6 seconds). Time to render the illustration itself is not included.

The primary system available for testing and development is a commercially-available notebook computer running 64-bit R 2.13.0 under Windows 7. This machine's CPU (Intel Core i7-2820QM @ 2.30GHz) has four cores, but no computation was ever observed to use more than a single core. R's memory use peaked under 200MB, a very modest footprint. Time tests were repeated at least twice and did not vary by more than one percent of the reported value despite leaving numerous other applications running on the system.

X. Conclusion and Future Work

With this visualization, we have overcome two limitations of the standard SOM visualization: (a) data population visualization for clusters and (b) visualization of the quantization error of a map. Our 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 points mapped to individual cells is revealed by positioning each point according to its similarity to neighbors.

Some further work is warranted to refine the method of scaling the 2-D data offsets. Furthermore, support for toroidal SOMs (where the edges of the maps are joined) in the data position calculations would extend our cartogram technique to a few more applications of the SOM.

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.

REFERENCES

[1] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Berlin, Heidelberg, New York: Springer, 2001.

[2] A. Ultsch, "Self-Organizing Neural Networks for Visualisation and Classification," in *Information and classification: concepts, methods, and applications*, University of Dortmund, 1993, pp. 307-313.

[3] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen, "SOM PAK: The self-organizing map program package," *Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science*, 1996.

[4] M. Newman, "Images of the social and economic world." [Online]. Available: http://www-personal.umich.edu/~mejn/cartograms/. [Accessed: 26-Feb-2012].

[5] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Human Genetics*, vol. 7, no. 2, pp. 179-188, 1936.

[6] E. Anderson, "The irises of the Gaspe Peninsula," *Bulletin of the American Iris society*, no. 59, pp. 2-5, 1935.

[7] M. T. Gastner and M. E. J. Newman, "Diffusion-based method for producing density-equalizing maps," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 20, pp. 7499-7504, May 2004.

[8] M. T. Gastner and M. E. J. Newman, "Supporting Text for: Diffusion-based method for producing density-equalizing maps." [Online]. Available: http://www.pnas.org/content/101/20/7499/suppl/DC1. [Accessed: 17-Jun-2011].

[9] E. J. Pebesma and R.S. Bivand, "Classes and methods for spatial data in R," *R News*, vol. 5, no. 2, 2005.

[10] T. Zumbrunn, "R-Forge: Diffusion-based cartograms," 26-Sep-2010. [Online]. Available: https://r-forge.r-project.org/projects/cart/. [Accessed: 01-Dec-2010].

[11] D. Temple Lang, "Rcartogram: Interface to Mark Newman's cartogram software," 15-Nov-2008. [Online]. Available: http://www.omegahat.org/Rcartogram/. [Accessed: 16-Jun-2011].

[12] Newman, M. E. J., "Cart: Computer software for making cartograms," 09-Nov-2006. [Online]. Available: http://www-personal.umich.edu/~mejn/cart/. [Accessed: 17-Jun-2011].

[13] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen, "SOM PAK: The Self-Organizing Map Program Package," Helsinki University of Technology, Laboratory of Computer and Information Science, FIN-02150 Espoo, Finland, Technical Report A31, 1996.

[14] D. C. Lay, *Linear Algebra and Its Applications, 3rd Updated Edition*, 3rd ed. Addison Wesley, 2005.

[15] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia, "3-dimensional pliable surfaces: for the effective presentation of visual information," *Proceedings of the 8th annual ACM symposium on User interface and software technology*, pp. 217–226, 1995.

[16] A. Frank and A. Asuncion, "UCI Machine Learning Repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml. [Accessed: 18-Feb-2011].

[17] D. Ayres-de Campos, J. Bernardes, A. Garrido, J. Marques-de-Sa, and L. Pereira-Leite, "SisPorto 2.0: a program for automated analysis of cardiotocograms.," *J Matern Fetal Med*, vol. 9, no. 5, pp. 311-8, 2000.

[18] "Cardiotocography - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Cardiotocography. [Accessed: 27-Feb-2012].

[19] J. Vesanto, "SOM-based data visualization methods," *Intelligent Data Analysis*, vol. 3, no. 2, pp. 111-126, Aug. 1999.

[20] R. Wehrens and L. M. C. Buydens, "Self- and Super-organising Maps in R: the kohonen package," *J. Stat. Softw.*, vol. 21, no. 5, 2007.

[21] A. Ultsch and F. Mörchen, "ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM," Dept. of Mathematics and Computer Science, University of Marburg, Germany, Technical Report No. 46, 2005.

[22] A. Ultsch, *U*-matrix: a tool to visualize clusters in high dimensional data*. Fachbereich Mathematik und Informatik, 2003.

[23] R. Henriques, F. BaCao, and V. Lobo, "Carto-SOM: cartogram creation using self-organizing maps," *Int. J. Geogr. Inf. Sci.*, vol. 23, no. 4, pp. 483-511, 2009.

[24] T. R Foundation for Statistical Computing, "The Comprehensive R Archive Network." [Online]. Available: http://cran.r-project.org/. [Accessed: 18-May-2011].

[25] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, Fourth. New York: Springer, 2002.

[26] C. Bergmeir and J. M. Benítez, *Neural Networks in R using the Stuttgart Neural Network Simulator: RSNNS*. 2010.

[27] J. Yan, "som: Self-Organizing Map," 2010. [Online]. Available: http://CRAN.R-project.org/package=som. [Accessed: 16-Jun-2011].

[28] R Development Core Team, *R: A Language and Environment for Statistical Computing*. Vienna, Austria: , 2011.

[29] P. Murrell, *R Graphics*, 1st ed. Chapman and Hall/CRC, 2005.

[30] L. Hamel and C. Brown, "Improved interpretability of the Unified Distance Matrix with Connected Components," in *Proceeding of the 7th International Conference on Data Mining*, Las Vegas Nevada, USA, 2011, pp. 338-343.

[31] "R-Forge: Welcome." [Online]. Available: https://r-forge.r-project.org/. [Accessed: 25-May-2011].