CARTOGRAM DATA PROJECTION FOR SELF-ORGANIZING MAPS

BY

DAVID H. BROWN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

UNIVERSITY OF RHODE ISLAND

2012

MASTER OF SCIENCE THESIS

OF

DAVID H. BROWN

APPROVED:

Thesis Committee:

Major Professor Lutz Hamel

Jean-Yves Hervé

Daniel W. Udwary

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2012

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.

ACKNOWLEDGMENTS

This thesis would not exist but for the patient encouragement and guidance of Dr. Lutz Hamel, my major professor, who realized I had found a topic for a thesis long before I did. Thanks also to the additional members of my core committee, Dr. Jean-Yves Hervé and Dr. Daniel Udwary; and to Dr. James Baglama (representing the Graduate School on the defense committee).

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

Figure **Page**

Figure **Page**

Figure **Page**

CHAPTER 1

Introduction

In this project, we present a solution to problems of information hiding and point occlusion found in popular visualizations of the Self-Organizing Map (SOM). The techniques we developed can be used to enhance many existing visualizations, providing additional insight into the data within a familiar frame of reference.

Kohonen's self-organizing maps [1] are a widely used method of visualizing structure in high-dimensional ($\mathbb{R}^n$) data sets. They employ an artificial neural network typically constructed and visualized as a regular two-dimensional mesh of nodes, each representing a vector trained to some point in the $\mathbb{R}^n$ data space. Thus, each node has both a feature-space $\mathbb{R}^n$ value and a 2-D (x, y) position.

In the original SOM algorithm, the $\mathbb{R}^n$ vectors in the map nodes are randomly initialized.[1, ch. 3.2] Each training datum is mapped to the nearest node and that node's $\mathbb{R}^n$ vector is adjusted to more closely match the $\mathbb{R}^n$ value of the datum. A lesser adjustment is also applied to nodes located within a certain radius of the nearest node's 2-D grid position. This neighborhood smoothing results in a global ordering of the map during the earlier iterations: "...the collection of models [nodes] is ordered by definition, if each model is equal to the average of input data mapped to its neighborhood." [1, p. 109] As the training progresses, the neighborhood radius is decreased and so the $\mathbb{R}^n$ vectors nodes will eventually converge on a stable arrangement.

1.1 Visualization of the SOM

The 2-D grid of nodes is appropriately the basis for most visualizations of the SOM map. Nodes are marked by outlines of the same size and shape. The space within may be used variously to display information about the $\mathbb{R}^n$ vector.

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. First we show a standard U-Matrix visualization of the familiar iris data set [4, 5] with mapped data randomly jittered within the cell. Darker shading indicates cluster boundaries (greater U-matrix distance between cells).

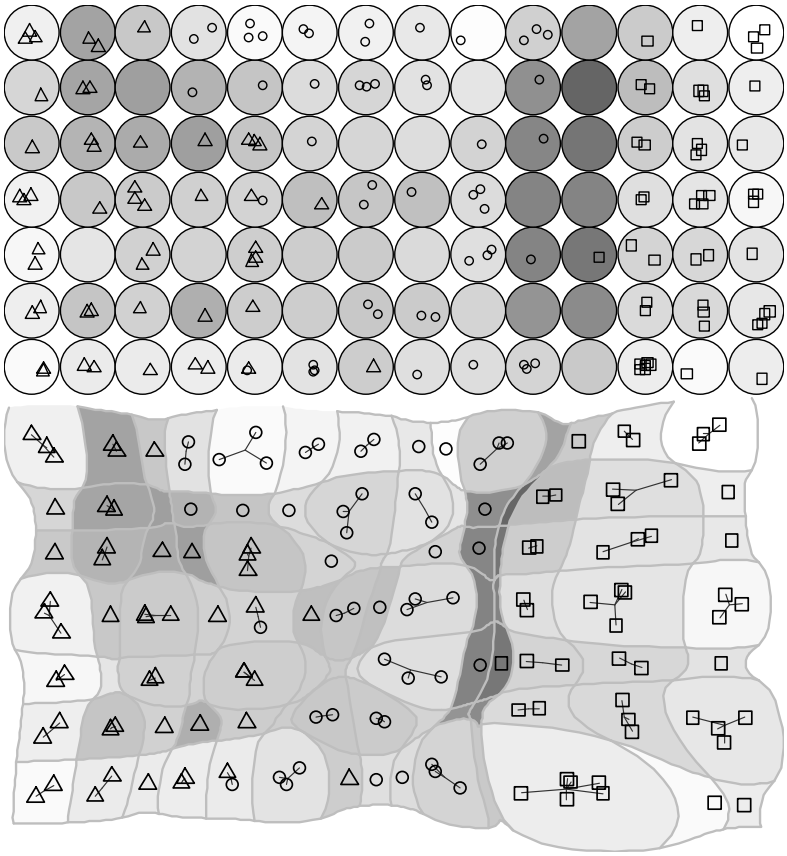

Figure 1. A comparison of the standard and enhanced SOM visualizations. Top, the typical SOM visualization ; bottom, our enhanced version of the same map. The SOM has been constructed from the Fisher/Anderson iris data set. Plot symbols correspond to iris species (class). The linearly separable *Setosa* species is on the right (square symbols).

While their exact location within the cell is meaningless, the plotted points do give a sense of the data density. Our enhanced visualization maintains the same U-matrix shading to indicate clustering and individual points do still imply data density. More explicitly, the size of each cell is scaled according to its data density; denser areas appear larger. The location of the mapped data within the cell is a projection of their similarity to neighboring cells, so a larger spread of the data points indicates how well a node fits its data as well as suggesting the number of data. The grid organization of the SOM mesh is still clearly perceived despite the distortion of the cartogram expansion.

List of References

[1] T. Kohonen, *Self-Organizing Maps*, 3rd ed., Springer Series in Information Sciences. Berlin, Heidelberg, New York: Springer, 2001, no. 30.

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

[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] E. Anderson, "The irises of the gaspe peninsula," *Bulletin of the American Iris society*, no. 59, pp. 2–5, 1935.

[5] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen*, vol. 7, no. Part II, pp. 179–88, 1936.

CHAPTER 2

Literature Review

2.1 Self-Organizing Map

A complete review of literature regarding the Self-Organizing Map is far beyond the scope of this project. As early as 1997, a bibliography [1] of some 3300 papers related to SOM was compiled; this had expanded to 5300 by 2001 [2], 7700 in the most recent compilation (through 2005) [3], and is likely to be over 10,000 today. [4]

These SOM bibliographies helpfully include keyword indices that identify papers of interest in areas such as visualization. However, these keywords are derived automatically from the titles and abstracts and do not always indicate the primary emphasis of the paper. Many papers listed under the "visualization" keyword are applications of the standard visualizations of the SOM in particular problem areas. Another subset of these papers evaluate use of the standard visualizations to evaluate a novel variant algorithm they use to generate the SOM. Similar results are found in electronic searches of various databases

Only a very few of these thousands of papers describe general-purpose methods of visualizing data on the SOM.

Among the most important of these is Ultsch's development of the U-matrix [5] that has become the *de facto* standard visualization. The "Unified distance matrix" (U-matrix) assigns a value to each node in the 2D SOM projection according to that average distance of that node's $\mathbb{R}^n$ codebook vector to the $\mathbb{R}^n$ vectors of the nodes surrounding it on the 2D map as demonstrated in Figure 2. Nodes with greater average distance (in $\mathbb{R}^n$) to their neighbors typically form boundaries between clusters. This average distance value has been represented in several ways, including the height columns in a 3D column plot superimposed on the map

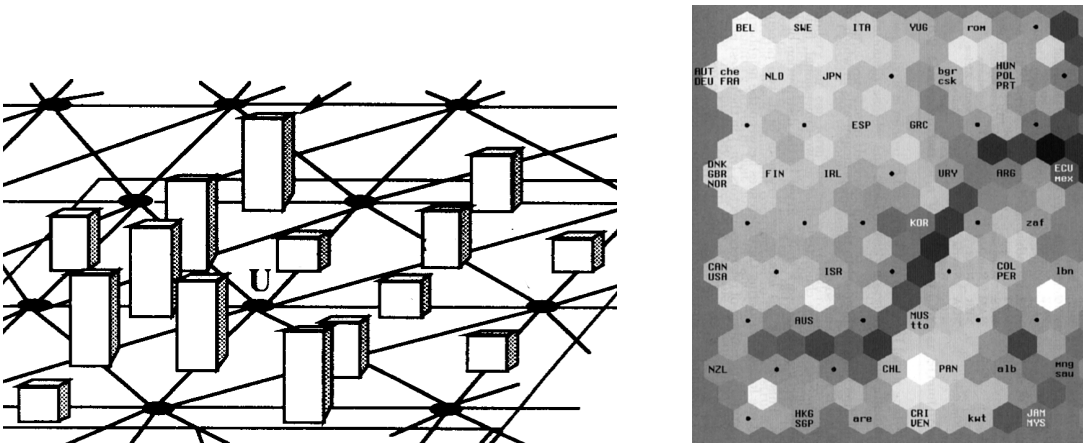

Figure 2. U-Matrix cluster visualization. On the left, 3D columns represent the $\mathbb{R}^n$ distances between nodes used to calculate the U-Matrix (Detail from [5], Figure 5). On the right, a SOM trained on various economic parameters of nation-states. Nodes are labeled with countries for which they are the best match and shaded according to the U-Matrix. (Detail from [7], Figure 3-31.) See also Figure 5.

[5], a third dimension for a rendered surface [6], and (probably most often), as a grayscale or colored heat map. [7]

More recently, Ultsch has developed the U*-Matrix [8] calculated by multiplying the U-Matrix node distance (in $\mathbb{R}^n$) value with an inverted data density (P-matrix) measure so as to emphasize low-density regions typically found between clusters. Utilizing toroidal maps with thousands or tens of thousands of nodes, Ultsch renders the U*-matrix as a 3D surface that can show clustering better than U-matrix alone (Figure 3). Because of the small size of each node in the plot, it would be difficult to apply our enhancements to this visualization. However, the use of data density is similar to our adoption of a density cartogram. Where the U*-Matrix smooths areas of high density, the cartogram expands them. Both approaches call attention to clusters.

The connected components [9] approach to cluster identification adds an intuitive overlay on top of a standard U-matrix plot. Each node in the map is connected to the neighboring node with the least unified distance value; that is,

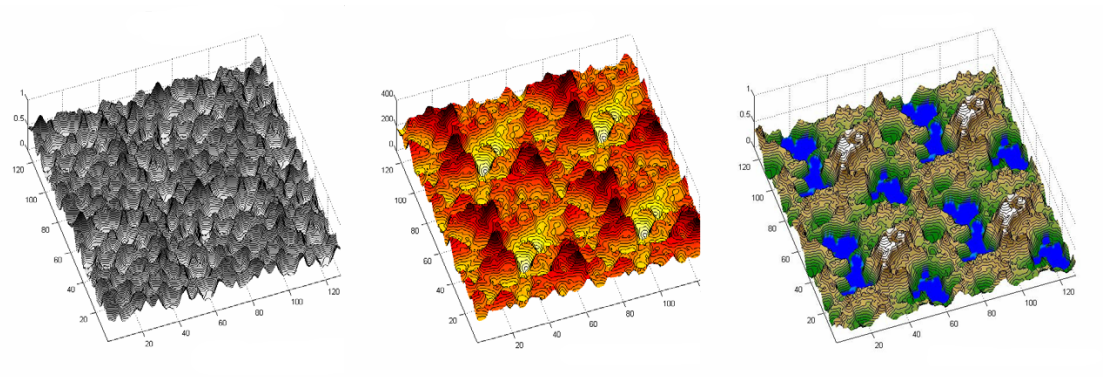

Figure 3. U*-Matrix SOM visualization. On the left, the standard U-Matrix of average $\mathbb{R}^n$ distances between nodes trained on ; center, the P-Matrix showing the average data density in a Pareto sphere around each node; right, the U*-Matrix de-emphasizes areas of high data density. The toroidal map is shown tiled in a 2×2 grid. The smoothed areas more clearly show clusters than U-Matrix alone. (Adapted from [8], Figures 4-6.)

following the maximal descent gradient. For any node, following a chain of these connections will point the way to some node which has the least unified distance value of any of its neighbors. If a node itself has the lowest unified distance of any of its neighbors, the gradient to itself is obviously zero and it may be considered to be the center of a cluster. A line is drawn from each node in the map to its central (zero-gradient) node forming a starburst pattern that clearly identifies the location and extent of clusters (Figure 4). We have implemented this visualization in the current project.

Pölzlbauer's vector field technique [10] (Figure 5) improves somewhat on the U-matrix by not only showing where boundaries are located but also labeling nodes with an arrow pointing toward the most similar adjacent node(s)—that is, most similar in $\mathbb{R}^n$ codebook values and adjacent on the 2D map. The length of the arrow indicates "the degree of how much the area it is pointing to is more similar to it than the opposite direction." Thus nodes at the boundaries of clusters point strongly toward their centers while nodes actually at the center of a cluster might show nothing more than a central dot of an arrow. Using arrows instead of gray levels

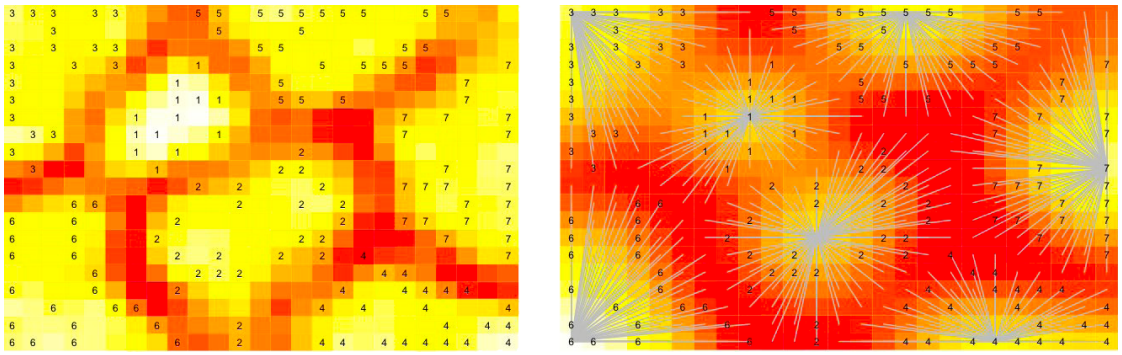

Figure 4. The connected components method of cluster visualization. On the left, a standard U-Matrix visualization of a SOM. On the right, the starburst display of the connected components method (from [9], Figures 4 and 5). The U-Matrix distance values have been smoothed on the right to identify only the most significant clusters.

to indicate boundaries would allow node coloring or shading to represent some other data layer. The technique seems largely compatible with our enhancements and could be a candidate for future inclusion in our visualization toolkit.

Another informative visualization by Pölzlbauer creates a graph from the SOM nodes, adding edges between nodes if some data mapped to one node is within a certain $\mathbb{R}^n$ radius of data mapped to another node. The graph edges are then drawn as lines connecting nodes on the 2D SOM map. In addition to offering insight into data density and clustering, areas in which the topology projection from $\mathbb{R}^n$ to 2D has been distorted can appear as edges that connect non-adjacent nodes. [11] This information could certainly be included with the cartogram-expanded map we propose.

An approach to finding clusters that is visually similar to the U-matrix was proposed by Pampalk *et al.* [12]. In this method, the frequency distribution of training data is mapped, assigning each datum not only to a single node, but proportionately among several, thereby calculating a smoothed data density histogram on top of the SOM. (This is somewhat similar to the P-Matrix that contributes to Ultsch's U*-Matrix.) Contours drawn around the areas of highest data density

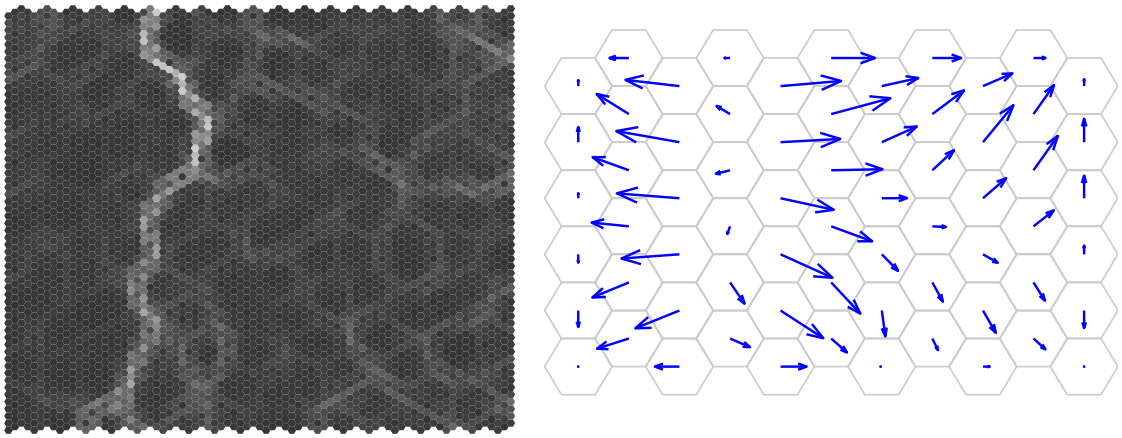

Figure 5. Vector field visualization. On the left, the typical U-Matrix visualization; on the right, the vector field shows not only cluster borders but also indicates the center of a cluster. (Adapted from [10], Figure 1). The SOMs have been constructed from the Fisher/Anderson iris data set.

reveal clustering (Figure 6).

Several visualizations based on the SOM have been developed by Vesanto and Himberg [13, 14]. One interesting approach uses the SOM as an intermediate step to summarize and organize a high-dimensional data set preparatory to another visualization. The SOM is smoothly colored so that the neighborhood association between the $\mathbb{R}^n$ codebook values will still be apparent in the final visualization. Particularly interesting are 2D and 3D Sammon's mappings where nodes adjacent in the SOM are connected to form a mesh that reveals something of the $\mathbb{R}^n$ shape of the SOM (Figure 7). [15]

2.2 Data occlusion

Occlusion, or hiding, occurs when two or more data points map to the same location in a data visualization and so are indistinguishable. This overplotting occurs when the data do in fact share the same values, and can also be caused by insufficient resolution in the visualization or by a dimensional reduction such as a Sammon's mapping [16] or the SOM.

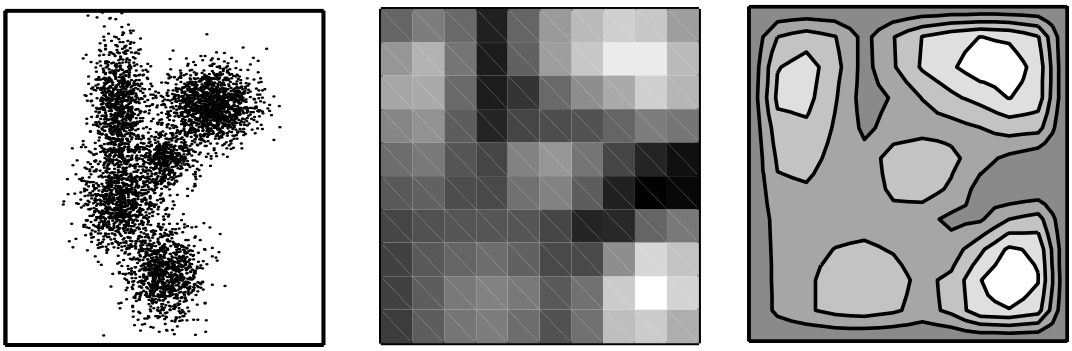

Figure 6. Smoothed Data Histograms for Cluster Visualization. On the left, a synthetic 2D dataset; center, the SOM trained on that data with data density used to shade map nodes (lighter=more dense). On the right, contours traced on the smoothed data density values of the trained SOM indicate clustering. (Adapted from [12], Figures 1-2.)

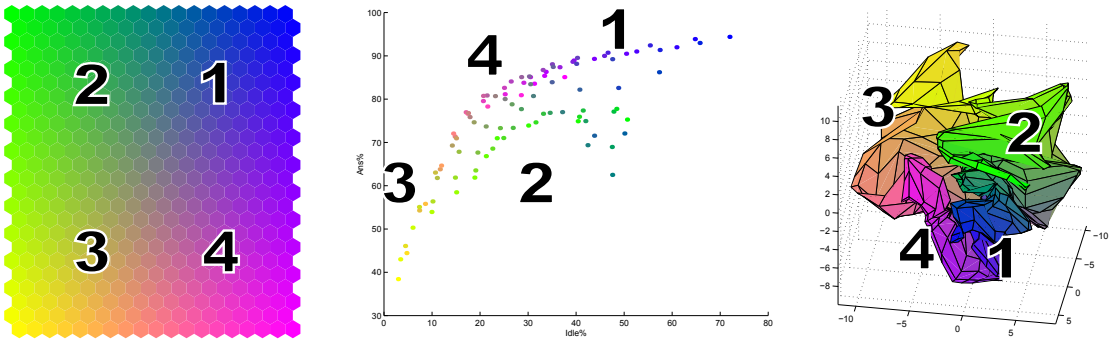

Figure 7. SOM coloring applied to other visualizations. Left, the color-opponent shading applied to the SOM by Vesanto and Himberg. Quadrant labels are added here to in case of monochrome reproduction. Blue and yellow are in opposition from upper right to bottom left (quadrants 1 and 3) with green versus magenta from upper left to bottom right (quadrants 1 and 4). Center, such a coloring applied to a scatter plot [13] clearly reveals that the lower-trending series is not just noise but part of a different cluster of data. Right, one view of an interactive model of the SOM showing a Sammon's mapping of the $\mathbb{R}^n$ codebook values to a 3D surface. [14]

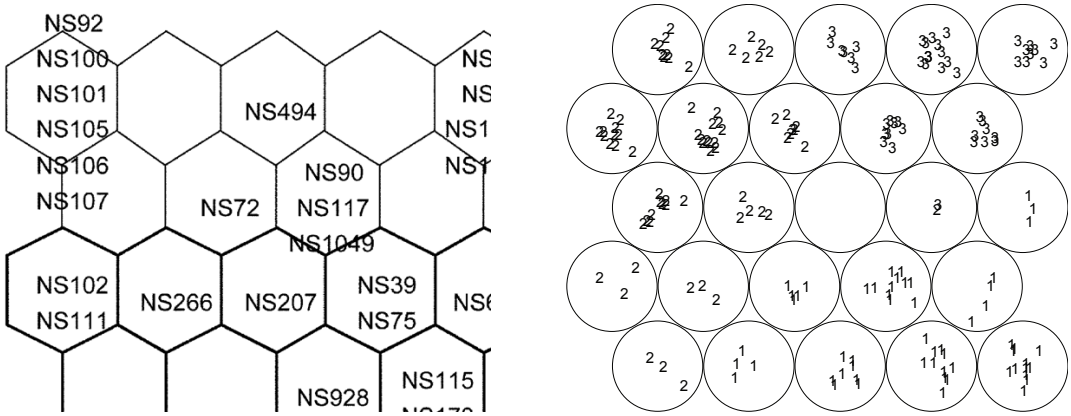

Figure 8. Typical solutions to data occlusion in SOM visualization. On the left, data labels are stacked around the center of the node. (Detail from [17], Figure 2). On the right, a demonstration from the kohonen R package[18] randomly positions (jitters) the labels within the node's cell.

This problem is exacerbated in the SOM because data are mapped to a fairly small number of discrete nodes. If the data are relatively few in number, discrete labels can simply be stacked—ideally, but not always, within the confines of the node's map cell. A larger number of data can be randomly jittered within the cell (Figure 8). Both of these approaches succeed in showing the data density within the SOM along with limited information about the identity of the individual datum (datum label or category mark). They do not convey any information about the distribution of the data subset or how near a datum is to a node's $\mathbb{R}^n$ vector.

Trutschl's Smart Jitter algorithm [19] uses a grid of SOMs to expand areas of high density. In this, he modifies visualizations such as a scatter plot by first grouping nearby points together in a regular grid by binning or rounding one or more of the $\mathbb{R}^n$ elements of the data to be plotted. For each of these primary grid divisions, the data found therein are used to train a small SOM. The data are mapped into the secondary grid of these small SOMs (Figure 9). So long as the number of nodes in the small SOMs is greater than the number of data to be plotted, one can be reasonably sure of seeing distinct marks for the data.

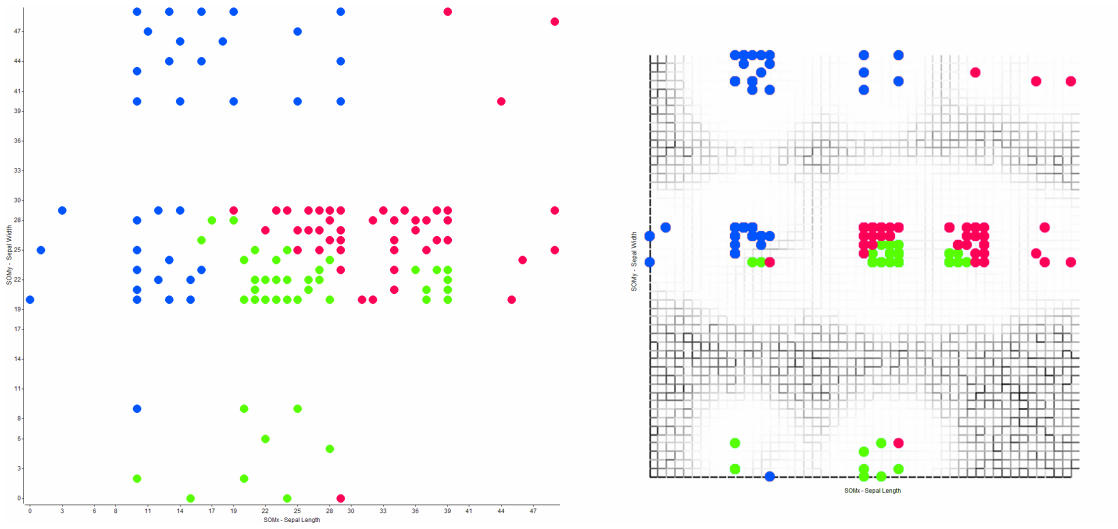

Figure 9. Trustschl's Smart Jitter of the iris data [20, 21] using a grid of SOMs embedded in a scatter plot of sepal length vs. sepal width (From [19]). On the left, a 5×5 primary grid with a 10×10 secondary (SOM) grid. On the right, a 10×10 primary grid with a 5×5 secondary grid.

Data with exactly the same values would still likely appear in the same place. Also, while the primary grid is arranged according to selected variables from the $\mathbb{R}^n$ data space, the SOM grids will likely arrange along the primary components of the data subset; it is not meaningful to identify a datum within a SOM as greater or less than another with respect to the primary grid axes. The resolution of those axes has been reduced to the primary grid. This approach is seen by the author as being most useful for visualizing local relationships among crowded data (e.g., identifying clusters) while retaining the use of more familiar plotting techniques such as the x/y scatter plot. (It would be interesting to apply this technique with the primary grid itself being a SOM.)

Keahy uses a nonlinear magnification [22] to resolve additional detail within 3D mappings of high-dimensional data. The regions to be magnified are selected according to the user's interest, and the author expects applicability of this visualization is limited to interactive graphical systems where the model can be rotated. The appearance of the nonlinearly magnified grid (Figure 10) is similar to the SOM

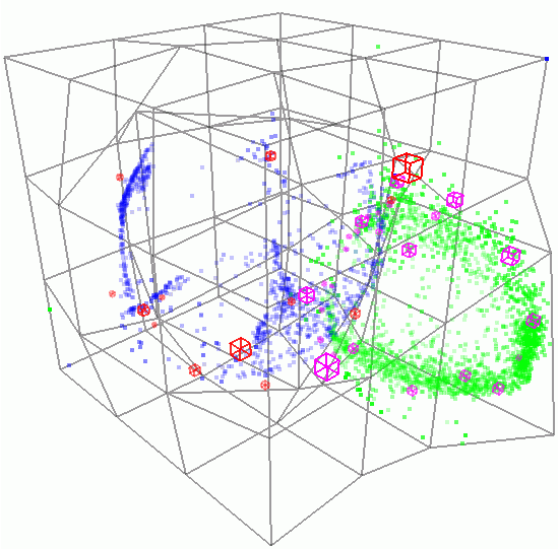

Figure 10. Nonlinear magnification with multiple centers of increased magnification. (From [22] , Figure 7.) Intended for use with an interactive data visualization environment where the model could be rotated in real time. The two rings are actually the same data set. Each ring shows a different subset of three normalized elements from the data vector. Enclosing polygons indicate the same datum brushed to reveal its location in each ring.

grid after the cartogram expansion we propose, if somewhat coarse and angular. (That may simply be a limitation of the computing and graphic resources at the time.)

Another tool for interactive visualization is provided by Ellis's Sampling Lens. [23] Within a specified area of the visualization, some random proportion of data are eliminated. This allows differences in cluster density to be seen where they would otherwise appear to be fully saturated. The effect is somewhat like changing the exposure on a camera to suit the lighting of a scene (Figure 11). It suffers the obvious drawback of not showing all the data in areas under the lens, but it does maintain the underlying grid.

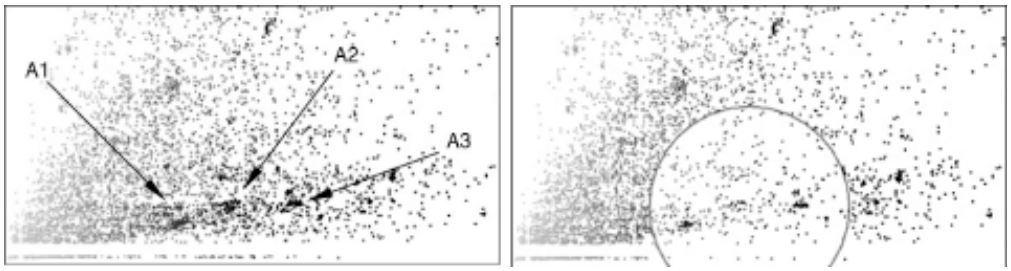

Figure 11. Sampling lens from [23], Figure 1. On the left, the over-saturated data plot. On the right, the sampling lens has reduced the overplotting to reveal that the middle cluster is in fact of lower density than the others.

2.3 Cartogram

A discussion of the history of the cartogram and algorithms for their generation follows in Chapter 3.

List of References

[1] S. Kaski, J. Kangasz, and T. Kohonen, "Bibliography of Self-Organizing map (SOM) papers: 1981-1997," *Neural Computing Surveys*, vol. 1, pp. 102–350, 1998.

[2] M. Oja, S. Kaski, and T. Kohonen, "Bibliography of self-organizing map (SOM) papers: 1998-2001 addendum," *Neural Computing Surveys*, vol. 3, no. 1, pp. 1–156, 2003.

[3] M. Pll, T. Honkela, and T. Kohonen, "Bibliography of self-organizing map (SOM) papers: 2002-2005 addendum," Helsinki University of Technology, Helsinki, Tech. Rep. TKK-ICS-R23, 2007.

[4] "Bibliography of SOM papers," Accessed: 2012-01-22. [Online]. Available: http://www.cis.hut.fi/research/refs/

[5] A. Ultsch, "Self-Organizing neural networks for visualisation and classification," in *Information and classification: concepts, methods, and applications*. University of Dortmund: Springer Verlag, 1993, pp. 307–313.

[6] E. L. Koua, "Using self-organizing maps for information visualization and knowledge discovery in complex geospatial datasets," in *Proceedings of 21st International Cartographic Renaissance (ICC)*, 2003, pp. 1694–1702.

[7] T. Kohonen, *Self-Organizing Maps*, 3rd ed., Springer Series in Information Sciences. Berlin, Heidelberg, New York: Springer, 2001, no. 30.

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

[9] 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: CSREA Press, July 2011, pp. 338–343.

[10] G. Plzlbauer, A. Rauber, and M. Dittenbach, "A vector field visualization technique for self-organizing maps," in *Advances in Knowledge Discovery and Data Mining*, 2005, pp. 399–409.

[11] G. Plzlbauer, A. Rauber, and M. Dittenbach, "Advanced visualization techniques for self-organizing maps with graph-based methods," *Advances in Neural NetworksISNN 2005*, pp. 813–813, 2005.

[12] E. Pampalk, A. Rauber, and D. Merkl, "Using smoothed data histograms for cluster visualization in Self-Organizing maps," in *Artificial Neural Networks ICANN 2002*, J. R. Dorronsoro, Ed., vol. 2415. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 871–876.

[13] J. Himberg, "Enhancing the SOM based data visualization by linking different data projections," in *Proceedings of the International Symposium on Intelligent Data Engineering and Learning (IDEAL'98)*, Hong Kong, Oct. 1998, p. 427434.

[14] J. Vesanto, J. Himberg, M. Siponen, and O. Simula, "Enhancing SOM based data visualization," in *Proceedings of the International Conference on Soft Computing and Information/Intelligent Systems (IIZUKA'98)*, Iizuka, Japan, Oct. 1998, p. 6467.

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

[16] J. Sammon, John W., "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. C-18, no. 5, pp. 401–409, May 1969.

[17] Y. Park, R. Crghino, A. Compin, and S. Lek, "Applications of artificial neural networks for patterning and predicting aquatic insect species richness in running waters," *Ecological Modelling*, vol. 160, no. 3, pp. 265–280, Feb. 2003.

[18] R. Wehrens and L. Buydens, "Self- and super-organising maps in R: the kohonen package," *J. Stat. Softw.*, vol. 21, no. 5, 2007.

[19] M. Trutschl, G. Grinstein, and U. Cvek, "Intelligently resolving point occlusion," in *IEEE Symposium on Information Visualization, 2003. INFOVIS 2003.* IEEE, Oct. 2003, pp. 131–136.

[20] E. Anderson, "The irises of the gaspe peninsula," *Bulletin of the American Iris society*, no. 59, pp. 2–5, 1935.

[21] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen*, vol. 7, no. Part II, pp. 179–88, 1936.

[22] T. A. Keahey, "Visualization of high-dimensional clusters using nonlinear magnification," *Visual Data Exploration and Analysis VI*, vol. 3643 of SPIE, 1999.

[23] G. Ellis, E. Bertini, and A. Dix, "The sampling lens: making sense of saturated visualisations," in *CHI '05 extended abstracts on Human factors in computing systems*, CHI EA '05. New York, NY, USA: ACM, 2005, pp. 1351–1354.

CHAPTER 3

Cartograms

3.1 Introduction to cartograms

"Cartogram" is a "rather vague term" [2] encompassing any of a variety of maps which set the area enclosed by regions of the map in proportion to some measured attribute of the region (other than its actual area). A common example of a cartogram presents a geopolitical map with regions sized according to their population, such as that of the United States shown in Figure 12. These are also referred to as "density-equalizing map projections." Distorting the shapes of a well-known map in this manner provides an additional channel for data and can work well along with techniques such as coloring or shading, superimposed symbols, or direct text labeling.

In addition to providing another means of communicating data, construction of a cartogram can help to even out the distribution of points of interest, as Raisz noted in 1934 [3]:

> The idea of the statistical cartogram occurred to the author when he had occasion to prepare maps of the United States showing the distribution of various economic units, such as steel factories, textile mills,

Figure 12. A continuous, area-by-value cartogram of the world according to the population of each nation. [1]

> power plants, banks, etc. These maps were far too crowded in the northeast to be useful, while elsewhere, for the most part, they were relatively empty. If a way could be found to increase the scale of the northeastern region and reduce that of the west, distribution could be shown more clearly.

We encounter a similar challenge with the Self-Organizing Map when plotting observations back onto the map. Within the area of a cluster, data points will be crowded, making them difficult to label and interpret; simultaneously, nodes corresponding to boundaries between clusters will have few points. In this visualization, we can expand and contract the plotted area of nodes in proportion to how many data are to be displayed in each.

3.1.1 Other variable-scale maps

This enlargement and reduction can also be applied specifically for the purpose of enhancing and calling attention to certain areas of the map. Harrie *et al.* [4] propose dynamically enlarging the current location of a user of a map on a mobile device so both local detail and the larger region can be simultaneously displayed. Here, the metric of the cartogram is distance from the user. Areas near the user are (presumably) of greater interest or importance and so are emphasized. (Figure 13).

By constructing a matrix of weights for each cell in the SOM according to our interest in its contents, we are able to achieve similar effects. In the iris dataset, there is an "inseparable" area of overlap between two of the three species. Giving each cell a score according to the number of species it contains allows us to call attention to that area of the population (Figure 14).

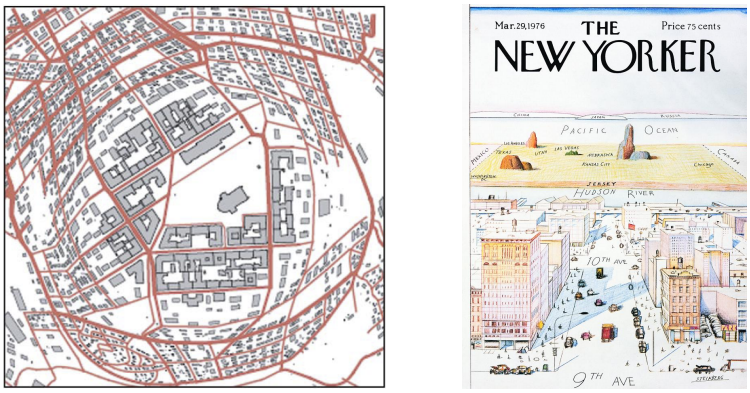

Figure 13. Two examples of variable-scale maps. On the left, a proposed map display [4] where the center area is enlarged to show detail around a mobile user and the periphery is reduced in scale to show as large a region as possible. On the right, the relative importance of local information is dramatized in this cover illustration [5]

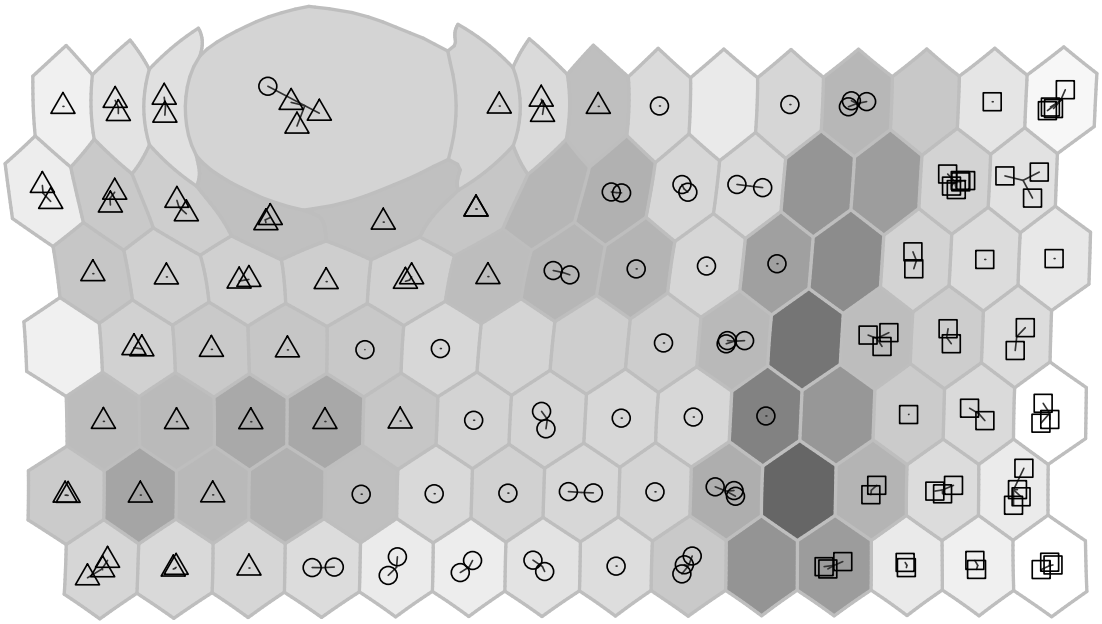

Figure 14. A SOM cartogram emphasizing nodes of particular interest. Cell sizes in this figure are enlarged to emphasize cells to which more than one species of iris map—i.e., the inseparable region in the dataset.

3.2 Selection of a cartogram algorithm

3.2.1 Prioritization of constraints

Designing a cartogram—or a cartogram-generating algorithm—requires compromise. An ideal cartogram would preserve the shape of every region; all regions would be oriented and connected to each other in the same way as in the original map (preserving the topology of the map), and each region will have exactly the intended area. Unsurprisingly, this is not generally achievable. Using simple checkerboard maps, Keim [6] demonstrated that it is not possible to preserve both the original shape of each outline and its topology while achieving the desired area for each shape. Some compromise of the shapes, of the map topology, or of the areas will be required; the rules must be relaxed. Different cartogram designs and algorithms prioritize some aspects and relax other constraints to different degrees. Extremes of these conflicting approaches include Raisz's rectangular statistical cartogram which discards shape entirely (and much of the original topology) to achieve the exact area desired. In contrast, the pseudo-cartogram of Tobler [7] preserves local shape, orientation, and relationships but areas are still far from what is desired. (Figure 15).

An important capability of the SOM is that it preserves topology [8, ch. 3.4] as it maps from the $\mathbb{R}^n$ input space to the 2D grid of the neural network. Thus, preservation of topology should also be a key consideration in our selection of a cartogram-generating algorithm. Given that the original shapes in our visualization are all the same (similar hexagons or rectangles), preservation of shape is of minimal importance.

Relaxing the shape should allow sufficient flexibility to achieve the desired areas with sufficient accuracy. We are adjusting areas primarily as a visual convenience—to allow more distinct observations to be plotted within the 2D borders of a cell—not necessarily to represent specific values.

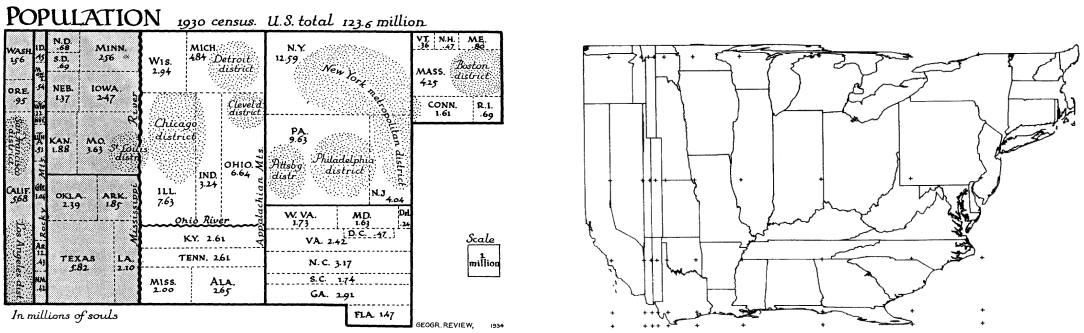

Figure 15. The relative balance between shape, topography, and areas will produce markedly different cartograms. In the rectangular statistical cartogram (left), Raisz choses to "discard altogether the outlines of the country"—both the shapes of the states and most of the map topology—in order to achieve the exact area required [3]. The pseudo-cartogram (right) preserves topology while only slightly relaxing shape constraints, but areas are far from their intended values. [7]

3.2.2 Shapes or sheet?

Algorithms which focus on the shapes (outlines) of individual areas devote significant attention and code to issues such as preventing self-intersection, simplifying outlines, reconnecting adjacent objects that moved apart, and preserving important features of the shapes such as angles between segments. [9, 6] These considerations are important for presenting a recognizable map based on familiar geography, but are largely irrelevant to the simple, tiled grids of the SOM visualization.

We need be able to address arbitrary points anywhere within our grid (not just the corners of the cells). Algorithms that can transform any point within the 2D coordinate plane rather than only existing vertices of the shapes are going to be much easier for us to use.

Tobler's rubber sheet method [10] was one of the first cartogram algorithms to change an underlying grid rather than the shapes of the map. Guesin-Zade and Tikunov [11] divide the plane of the map into cells, calculate a density value for each cell, and construct a vector field that seeks to make that density uniform

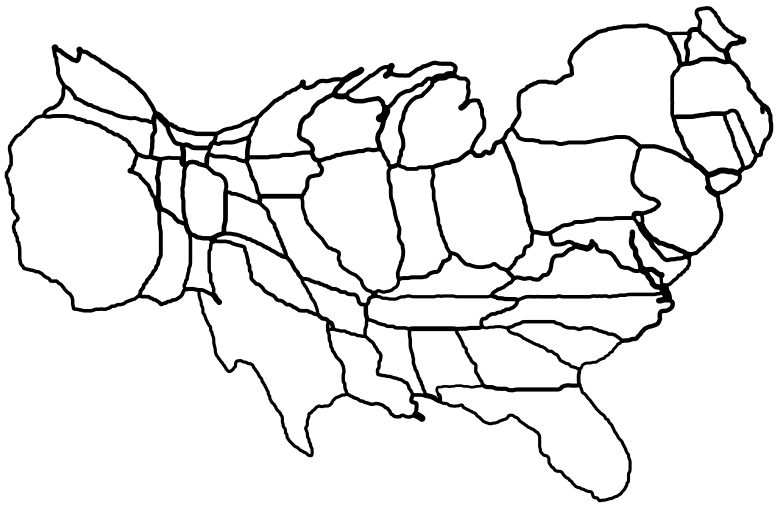

Figure 16. A continuous, area-by-value cartogram of the United States of America based on population data.[11]

with (an example is shown in Figure 16). Gastner and Newmann take a similar approach [12] based on density gradients, allowing more flexibility in the selection of a density function. The results of this approach are often quite aesthetic; Figure 12 opened this chapter with an example of their work.

3.2.3 Selecting available code

Development of a novel cartogram algorithm is beyond the scope of this project. Some ([9, 6]) offer pseudo-code and key formulae. Most usefully, Mark Newmann provided portable C code for the density-equalizing maps [12] which has been built into a R package by Duncan Temple Lang, available at the Omegahat repository [13]. This package is used to perform the cartogram transformation in this project.

List of References

[1] M. Newman, "Images of the social and economic world," Accessed: 2012-02-26. [Online]. Available: http://www-personal.umich.edu/~mejn/cartograms/

[2] W. R. Tobler, "Geographic area and map projections," *Geographical Review*, vol. 53, no. 1, pp. 59–78, Jan. 1963.

[3] E. Raisz, "The rectangular statistical cartogram," *Geographical Review*, vol. 24, no. 2, pp. 292–296, Apr. 1934.

[4] L. Harrie, L. T. Sarjakoski, and L. Lehto, "A variable-scale map for small-display cartography," *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, vol. 34, no. 4, pp. 237–242, 2002.

[5] S. Steinberg, "Front cover," *The New Yorker*, p. OFC, Mar. 1976.

[6] D. A. Keim, S. C. North, and C. Panse, "CartoDraw: a fast algorithm for generating contiguous cartograms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 1, pp. 95–110, 2004.

[7] W. R. Tobler, "Pseudo-cartograms," *Cartography and Geographic Information Science*, vol. 13, no. 1, pp. 43–50, 1986.

[8] T. Kohonen, *Self-Organizing Maps*, 3rd ed., Springer Series in Information Sciences. Berlin, Heidelberg, New York: Springer, 2001, no. 30.

[9] D. H. House and C. J. Kocmoud, "Continuous cartogram construction," in *Proceedings of the conference on Visualization '98*. Research Triangle Park, North Carolina, United States: IEEE Computer Society Press, 1998, pp. 197–204.

[10] W. R. Tobler, "A continuous transformation useful for districting," *Annals, New York Academy of Sciences*, no. 219, pp. 215–220, 1973.

[11] S. M. Gusein-Zade and V. S. Tikunov, "A new technique for constructing continuous cartograms," *Cartography and Geographic Information Science*, vol. 20, pp. 167–173, July 1993.

[12] 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.

[13] D. Temple Lang, "Rcartogram: Interface to mark newman's cartogram software," Accessed: 2011-06-16, Nov. 2008, R package version 0.2-2. [Online]. Available: http://www.omegahat.org/Rcartogram/

CHAPTER 4

Important Structures of the Self-Organizing Map

In this section, we identify and discuss aspects of the self-organizing map that persist after training and so can be used in creating the visualizations of this project. Data structures used by the existing R packages that implement the SOM algorithm—kohonen and som [1, 2]—are of particular interest.

As might be expected in a developing field with hundreds of contributing authors, terminology is not precisely and invariably defined. Kohonen's book [3] is of course an important guide, and in [4], Ultsch conveniently outlines his preferred terminology. We have tried to adopt the terminology of these authors where appropriate and unambiguous.

4.1 Properties of the Map

The first step in creating a Self-Organizing Map in R is to define the geometry of the grid of nodes. The size and shape of the grid the number of nodes along each of its x and y dimensions is of primary importance. Depending on the SOM-generating algorithm selected [1, 2], additional parameters may include whether the nodes are to be arranged in a rectangular or a hexagonal grid and whether the map is toroidal. A toroidal map considers the top row to be adjacent to the bottom and the left column adjacent to the right, thereby avoiding edge effects.

4.2 Properties of the Data

4.2.1 Training Data

Training data are those data values used to build the SOM. A sampling of the training data may be used to initialize the codebook values of the nodes. During training, values of the training data are used to adjust the codebook values. The SOM is a mapping of the training data.

4.2.2 Dimensionality of the Data Space

The data are points in some multidimensional space $\mathbb{R}^n$, where n is the number of real-valued attributes of each datum. A datum is sometimes referred to as a data vector; n is the length of the vector. Ultsch uses "data space" to refer to that "subspace of $\mathbb{R}^n$ where data points of an application can be observed." (Some work has been done to adapt SOM to categorical data [5] but we do not attempt to support such data.)

4.2.3 Normalization

The data used to train the SOM may be normalized, i.e., shifted and scaled to have a mean of 0 and a standard deviation (variance) of 1. Where this is done, the centers and scales will need to be retained if new data is to be plotted to the map or if the codebook values of the trained nodes are to be expressed in the original $\mathbb{R}^n$ space. (R's scale function is ideal for this; it operates on an entire matrix and returns the scale and centers along with the transformed matrix).

4.2.4 Best-Match Node

During and after training, any datum in $\mathbb{R}^n$ (not just training data) may be compared to each of the codebook values of the SOM nodes and their separation calculated (e.g., Euclidean distance). Whichever node has the codebook value nearest to the datum is the best-match node for that datum. During training, these associations are changing, but after training they are fixed. Because it requires so many comparisons, some SOM algorithms will save the final best-match nodes of the training data for later use.

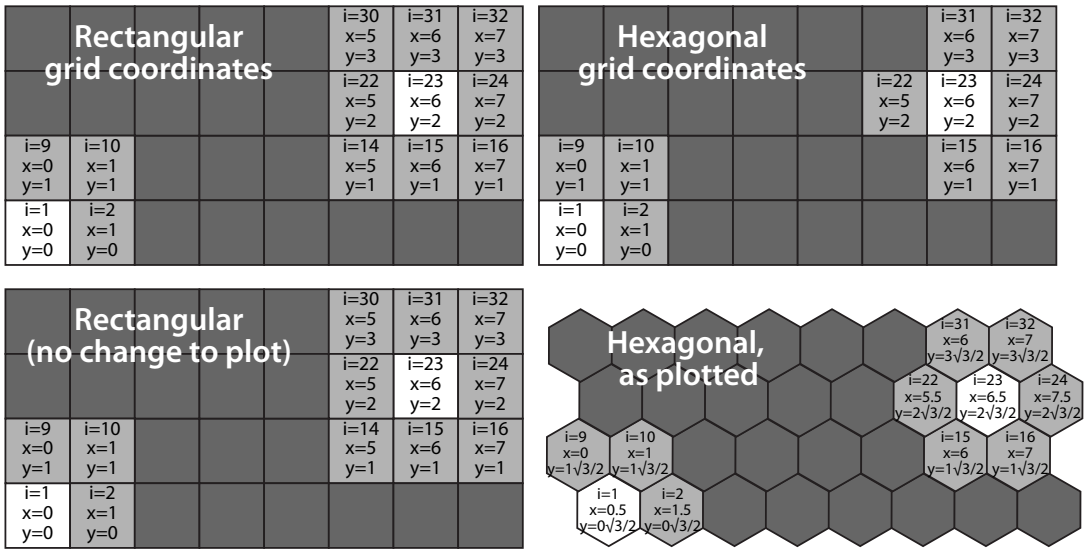

Figure 17. Transformation of node index to (x, y) (row, column) grid coordinate pairs. Selected nodes are labeled with their node index i and (x, y) grid coordinates. Neighbors of the white-background nodes are shown with a light background. Non-neighbor nodes are shown with a dark background.

4.3 Inherent Properties of Nodes

4.3.1 Row-Column Grid Coordinates

Both rectangular and hexagonal SOM grids are defined by the number of rows and number of columns in the grid. The only real difference is how the neighborhood is determined (see Figure 17). Each node may be identified by its row and column in this grid, an (x, y) position, where x and y are integers.

For convenience in certain calculations, these values may be zero-indexed, so if a $w \times h$ (width by height) map is created, the (x, y) grid positions will range from $(0, 0)$ to $(w - 1, h - 1)$.

4.3.2 2D Plotting Center

For a rectangular (square) SOM, the (x, y) row-column grid coordinates of a node can also serve as the plotting location of a node in a visualization. We chose to consider this value to be the center of the cell rather than some corner for two reasons: first, hexagons do not have any vertices coincident with those of a square

but they do have a center; second, our data mapping plot each datum relative to the center of its best-match node cell.

To transform the grid coordinates of the node centers for plotting a hexagonal grid, one need only multiply the y coordinate by $(\sqrt{3} \div 2)$ and in alternate rows, the x coordinate is shifted by 0.5. This is shown in the bottom-right grid in Figure 17.

The plotting centers may be pre-calculated and stored with the map (the kohonen package does so).

4.3.3 Neighboring Nodes

The neighbors of a node are those nodes that are adjacent in the 2D representation of the SOM grid. Each node has several neighbors in the map as shown in Figure 17. The number of neighbors is dependent on the geometry of the map and will range from a minimum of 3 in a corner to a maximum of 6 or 8 in the center of a hexagonal or rectangular grid, respectively.

The set of neighbors may be determined by examination of the row-column grid coordinates—e.g., the nodes at the relative positions $(+1, 0)$, $(-1, 0)$, etc. are neighbors unless the resulting coordinates are outside the grid. A more complicated Euclidean distance measurement between the plotted centers might alternatively be used.

Calculation in the neighbor set can vary according to the needs of a particular algorithm. For example, when the kohonen package calculates U-matrix distances in a rectangular map, it considers only orthogonal neighbors, not diagonals.

Some SOM constructions consider the map to be toroidal, wrapping from left-to-right and top-to-bottom (and the reverse) to remove any edge effects. In such a map, each node has the maximum number of neighbors.

4.3.4 Node Index

It would be cumbersome to refer to the map nodes by their (x, y) row-column positions, using two values where one might suffice. Even worse to use the plotted centers where for hexagonal maps, the coordinates are not integers. Both the kohonen and som packages instead store properties of their nodes as matrices where each row contains the value for a single node. The $\mathbb{R}^n$ data-space values of each node are stored, for example, as a $m \times n$ matrix where m is the total number of nodes. This approach is both efficient and convenient when programming; the node index—the row number of this $m \times n$ matrix—identifies the node in most contexts, allowing us to identify sets of nodes with a simple numeric vector.

The node index, i, is easily calculated from the column and row positions of the node (1a) and the number of columns in the grid and a node index can also be transformed back to (x, y) coordinates (1b, 1c). Node 1 is at the bottom left of the grid; Node 2 is to its right; when the end of the row is reached, counting resumes at the left edge of the second row from the bottom and the node with the greatest index is at the upper right. The R expressions for these conversions follow where i is the node index, x and y are the node's 0-indexed row-column position, and w is the grid width (number of columns in each row):

$$i = 1 + x + (y * n) \tag{1a}$$

$$x = (i - 1) \,\%\%\, n \tag{1b}$$

$$y = (i - 1) \,\%/\%\, n. \tag{1c}$$

As with many programming languages, * indicates multiplication; %% is R's modulus operator (remainder after integer division), and %/% is R's integer division operator (throws away any remainder).

4.3.5 Node cell and enclosing polygon

For the purpose of our visualizations, we consider the 2D space of the SOM projection to be continuous, not restricted to the integer coordinates of the nodes themselves. To achieve this, we must continuously tile the 2D grid, enclosing each node at the center a polygon: nominally a square or a hexagon depending on the map configuration. (Some packages plot nodes as circles which leave gaps in the plane; see Figure 1 in Chapter 1.) It is desirable that any graphical information pertaining to a node should be plotted within the area bounded by this polygon; this area is the cell.

4.4 Trained Properties of Nodes

Once training of the SOM is complete, each node can be thought of as acquiring additional properties.

4.4.1 $\mathbb{R}^n$ value

After training, each node retains the final n-dimensional data-space value learned from the training data; these values are the essential information produced by the SOM algorithm[3]. Various descriptive names have been used in the literature, including "model vector," "reference vector," "n-dimensional weight value," "codebook vector," "weight vector," "prototype," "weight," "map unit vector," and yet more. Many of these connote additional meaning in their specific context. We will adopt "$\mathbb{R}^n$ value" as the most universal term.

4.4.2 Unified distance, or U-Matrix

For each node, the average distance in $\mathbb{R}^n$ space to its neighboring nodes in the 2D mesh is calculated; this the unified distance, or unit distance measure. These values are typically used to shade the background of a cell, The Unified Distance Matrix (U-matrix) [6] is a popular method of visualizing the SOM.

4.4.3 Mapped Training Data

A node's mapped training data are that subset of the training data that (after training is complete) are nearer to this node than to any other node—i.e., this node is best-match node for those data. Counting the number of data mapped to a node gives us the data density and can be used to control the cartogram expansion/contraction of the map.

4.4.4 Quantization Error

The quantization error is a measure of how well a node represents its mapped training data. A typical calculation would be to average the Euclidean $\mathbb{R}^n$ distances between a node's codebook value and each of its mapped training data, though SOM variants may use some other distance measure. Let D_i be the set of training data mapping to a node at index i whose $\mathbb{R}^n$ value is x_i:

$$q_i = \begin{cases} \text{NA}, & \text{if } D \text{ is an empty set} \\ \frac{\sum_{d \in D} \|x_i - d\|}{|D|}, & \text{otherwise} \end{cases} \tag{2}$$

List of References

[1] R. Wehrens and L. Buydens, "Self- and super-organising maps in R: the kohonen package," *J. Stat. Softw.*, vol. 21, no. 5, 2007.

[2] J. Yan, "som: Self-Organizing map," Accessed: 2011-06-16, 2010, R package version 0.3-5. [Online]. Available: http://CRAN.R-project.org/package=som

[3] T. Kohonen, *Self-Organizing Maps*, 3rd ed., Springer Series in Information Sciences. Berlin, Heidelberg, New York: Springer, 2001, no. 30.

[4] A. Ultsch, "Maps for the visualization of high-dimensional data spaces," in *Proc. Workshop on Self organizing Maps*, 2003, p. 225230.

[5] C. C. Hsu, "Generalizing self-organizing map for categorical data," *IEEE Transactions on Neural Networks*, vol. 17, no. 2, pp. 294–304, 2006.

[6] A. Ultsch, "Self-Organizing neural networks for visualisation and classification," in *Information and classification: concepts, methods, and applications*. University of Dortmund: Springer Verlag, 1993, pp. 307–313.

CHAPTER 5

Data Projection Within the Cell

To position each datum within the cartogram-expanded data cell, we begin (as do other visualizations) by selecting the best-match node (the node nearest to the datum in the $\mathbb{R}^n$ data space). We will plot the datum at some location within that node's cell (enclosing polygon). 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.

5.1 Selecting the Best-matching 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 Euclidean distance (3) [1]:

$$\|x - m_c\| = \min_i\{\|x - m_i\|\}. \tag{3a}$$

$$c = \arg\min_i\{\|x - m_i\|\}. \tag{3b}$$

Node c is the node nearest to the datum in the data space—i.e., its best-match node.

5.1.1 Computational Complexity

To calculate the Euclidean distance between two arbitrary points in $\mathbb{R}^n$ space requires n subtractions to calculate the difference vector $x - m_i$ n multiplications to calculate the square of each component of this vector, $n - 1$ additions to sum the squares, and finally one square root calculation. The calculation cannot be done in fewer steps nor will it ever require more, so the asymptotic complexity of Euclidean distance is $\Theta(n)$. We assume that the best-match node must be found using an exhaustive search of all m nodes in the SOM map, requiring m calculations of Euclidean distance in $\mathbb{R}^n$ space and m comparisons. Thus the asymptotic complexity of selecting the nearest node is $\Theta(m \times n)$ for each datum to be plotted. This is the most complex operation our approach requires. However, finding the bestmatch node for the training data is also required for building the SOM; some packages will save this information in which case we don't need to recalculate it.

5.2 Finding Vectors to Neighbors

The feature-space vectors to each of the j neighbors (m'_j) are calculated simply by subtracting the $\mathbb{R}^n$ feature-space value of the nearest node, m_c, from that of each neighbor m_j, a linear translation, (4):

$$m'_j = m_j - m_c. \tag{4}$$

Likewise, the datum's translated vector is (5):

$$x' = x - m_c. \tag{5}$$

Applying the same translation to the nearest node itself (6) confirms its role as the origin for the calculations that follow (its value is 0 in every $\mathbb{R}^n$ coordinate axis):

$$\mathbf{0} = m_c - m_c. \tag{6}$$

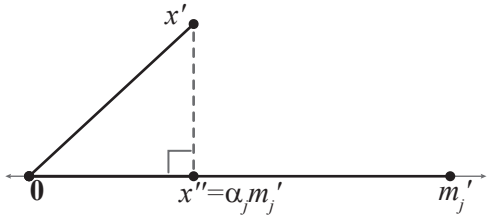

Figure 18. Orthogonal projection (x'') of a datum vector (x') onto a neighboring node vector (m'_j). This calculation is generalizable to the $\mathbb{R}^n$ feature space of the SOM. [2]

5.2.1 Computational Complexity

Translating the datum and each neighbor in $\mathbb{R}^n$ space requires $n(t_i + 1)$ subtractions where t is the number of neighboring nodes and i is the index of the nearest node. The value t_i will vary according to the topology of the map, ranging from $t_i = 3$ for a corner of a (non-toroidal) map (rectangular or hexagonal) to $t_i = 8$ for a node in the interior of a rectangular map. Because t_i is so bounded to single-digit values, the asymptotic complexity is $O(n)$.

5.3 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'') 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 18, 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 (7) [2]:

$$\alpha_j = \frac{x' \cdot m'_j}{m'_j \cdot m'_j}. \tag{7}$$

If the datum 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 datum in their direction on the grid while neighbors with negative α push the datum away.

5.3.1 Computational Complexity

In $\mathbb{R}^n$ space, the calculation of the dot product requires n multiplications and $n-1$ additions. Calculating a projection require two dot products and a division. To find all the projections to neighbors requires $2t_i(2(n+n-1)+1)$ operations which is $\Theta(n)$.

5.4 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_j are multiplied by the proportional length α_j and added together (8) to form a raw 2-D offset vector, r:

$$r = \sum_{j \in \{\text{neighbors}\}} \alpha_j (g_j - g_c). \tag{8}$$

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 α, pulling the datum to the left, the neighbor to the right might very well have a negative α 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 that the simplest satisfactory scaling

function (9) is to divide the raw offset by the number of neighbors (t_i) surrounding the nearest node:

$$s = \frac{r}{t_i}. \tag{9}$$

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 and may be addressed in future work. Finally, the scaled offset s is added to the 2-D coordinates of the nearest node, g_c (10), giving the plotted grid position of the datum, g_d :

$$g_d = g_c + s. \tag{10}$$

5.4.1 Computational Complexity

To translate the neighbor nodes' 2D grid coordinates requires $2(t_i)$ subtractions and scaling each by its α is another $2(t_i)$ multiplications. Combining these 2D offsets is $2(t_i - 1)$ additions. Scaling this 2D offset requires a further 2 multiplications. The total $6(t_i)$ operations are $O(1)$ because t_i can never be greater than 8.

5.5 Overall Computational Complexity of Mapping

The computation required to position a point on the SOM in this manner is determined by the size and geometry of the underlying SOM. the sum of the following components:

- finding the nearest node, $O(m) \times O(n) = O(m \times n)$,
- calculating distances to each neighbor, $O(n)$,
- determining the orthogonal projection toward each neighbor, $O(n)$, and
- scaling the 2D offset, $O(1)$ for this method.

Adding these together, we have $O(m \times n) + 2O(n) + O(1)$. Either m (the number of nodes in the map) or n (the $\mathbb{R}^n$ dimension of the data) could be large and dominate. Ultsch's U*-Matrix examples [3] contain $(2^6)^2 = 4096$ and $(2^7)^2 = 16536$ nodes. Some public data sets—not only those involving a time series—have tens of thousands of numerical attributes [4]. The most appropriate description of complexity is $O(m \times n)$. Having mapped one datum provides little-to-no information to use in the next, so if there are a series of p data to be mapped, the complexity is $O(m \times n \times p)$.

If the datum to be mapped is a member of the map's training data, then its nearest node may already be known. If so, the complexity of mapping that one datum would simply be $O(n)$; of a set, $O(n \times p)$.

5.6 Visual representation

5.6.1 Point symbol

An appropriate symbol or label is drawn at the position g_d calculated by Equation (10).

5.6.2 Center trace

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 cell to the left of the bottom-right corner of Figure 22. Despite the dramatic 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.

List of References

[1] 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.

[2] D. C. Lay, *Linear algebra and its applications*, 3rd ed. Addison Wesley, Sept. 2005.

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

[4] A. Frank and A. Asuncion, "UCI machine learning repository," Accessed: 2011-02-18, 2010, university of California, Irvine, School of Information and Computer Sciences. [Online]. Available: http://archive.ics.uci.edu/ml

CHAPTER 6

Development and Implementation in R

A primary goal of this project was to make this visualization widely available to the research community as usable software. We selected the R Language and Environment for Statistical Computing [1] in which to develop this visualization. As free and open-source (GPL) software, R enjoys strong and active development, research, and user communities (e.g., [2, 3]). R supports extension through user-defined packages written in the same interpreted language (R) by which users interact with the system. R is used as a data mining and visualization environment in courses in the University of Rhode Island Computer Science and Statistics department. The graphics systems available in R are sufficiently capable to realize the visualizations intended by this project. (We are using the grid graphics library [4].) At least two packages for creating and visualizing self-organizing maps are available in R, we use both the som package [5] and the kohonen package [6]. Support for Gastner and Newmann's diffusion cartograms [7] is available in the Rcartogram package [8] which provides an R interface to Newmann's Cart software [9].

Other options considered include WEKA [10], another open-source data exploration and visualization environment. In contrast to R's text-driven operation, WEKA is primarily controlled through its extensive graphical user interface. Recent versions offer increasing extensibility capability through plug-ins written in Java, and a package that implements the SOM algorithm does exist (though without any visualization capability) [11]. However, only a handful of visualization plug-in categories are listed in the most recent WEKA documentation [12], none of which seem to provide the capabilities needed by this project. MATLAB [13]

would also be a viable option, but its cost made it less attractive and a GPL SOM Toolbox for MATLAB has not yet been updated to work with recent versions of MATLAB [14].

6.1 R package architecture

R is extensible through the installation of packages distributed online through various sources such as the Comprehensive R Archive Network (CRAN) [3] and R-forge [2]. The procedure of making packages is thoroughly documented in the "Writing R Extensions" section of the online R manual [1].

The name of our package, "somTools" was selected to be able to accommodate the variety of functions, techniques, and methods being developed by members of the University of Rhode Island Department of Computer Science and Statistics Machine Learning Group. In this initial form, the somTools package includes the somTools class, numerous functions supporting the somTools to realize the visualizations, and a few more general functions that are not specific to somTools.

6.2 The somTools class

While there was no need to write yet another SOM-generating routine to implement this visualization, we do need access to the information contained within a SOM. The data structures created by the kohonen and som packages are similar but are not identical. Because we want to support both packages—and because variant algorithms for constructing the SOM are an area of continuing research and development [15, Ch. 5] which seem likely to implement yet more variant data structures—it was appropriate to create our own universal SOM structure, implemented as a S4 class [16] in R (see Figure 19 and Listing A.1). In the somTools, we retain the following essential properties without which the object is not valid:

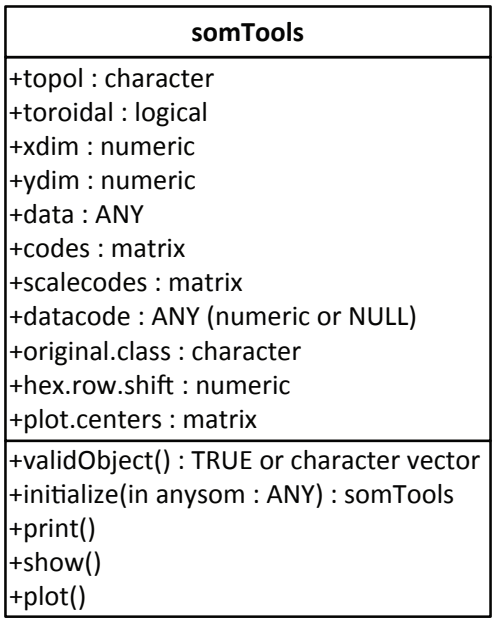

Figure 19. UML class structure diagram of important data and methods of the somTools class.

- codes—matrix of the codebook $\mathbb{R}^n$ values of each node in the map; each row represents one node and the row number is the node index
- topol—the topology of the map: a character string beginning either "hex" for hexagonal maps or "rect" for rectangular (square) maps
- original.class—a character string such as "kohonen" or "som"
- xdim, ydim—the number of columns and rows of nodes in the map

Additionally, the somTools can store the following, either copied from the original object, calculated during initialization or set by the user:

- toroidal—whether the map was calculated with joined edges; not all of our methods support such maps
- data—a matrix the original training data used to build the map

- scalecodes—a copy of the codebook values (stored in codes) scaled according to R defaults such that values ±1 are 1 standard deviation away from the mean (0)

- datacode—a vector of node indexes indicating the best-matching code node for each of the training data

6.2.1 Methods and functions of somTools

The function is the most important programming unit in R [17]; this is in contrast to languages such as Java where the object (as defined by its class) is primary. Common actions such as print() or plot() might require very different behavior for different data types (classes). In R, these common actions are termed generic functions and a call to setMethod() creates a binding between a signature—the list of class names of each argument to the function—and the appropriate function definition. With this information, the correct implementation of a generic function call can be dispatched.

The somTools package provides methods for a few generic functions. Listing A.2 in the appendix shows the intialization method of the somTools object, with translation functions for each of the two supported classes of SOM, kohonen and som. Other generic functions of the somTools class include a print/show method that summarizes its contents as text output.

Numerous additional functions are defined as part of the package. Many of these are primarily intended for internal use in plotting the visualization and are prefixed somTools.*functionName*().

Where practical, functions have been designed to use R's efficient vector operations rather than iteration.[16, Ch. 6.4]

6.3 Locating and Interpolating Between SOM Nodes

One innovation in our visualization is that we plot each mapped data within its best-match node cell according to its similarity neighboring nodes' $\mathbb{R}^n$ values. This algorithm is described in Chapter 5 and is calculated using the row-column grid position of the nodes as integer reference points in a continuous 2D plane. This simplifies some of the calculations and allows us to use the same equations for rectangular and hexagonal grids.

6.3.1 Adaptation for Hexagonal Maps

As described in Section 4.3, this row-column grid of a rectangular map also corresponds to the plotting coordinates; for these maps, we can use our calculated values directly. However, if the map topology is hexagonal, we must transform both the x and y from row-column grid coordinates to plotting coordinates. The function (11) to transform the y coordinate is trivial:

$$y_{\text{hex}} = y_{\text{rect}} \times \frac{\sqrt{3}}{2}. \tag{11}$$

Because every other row is shifted $\frac{1}{2}$ unit in the x axis, our x transformation adds an x offset calculated as a continuous function of y. Such a"triangle wave" function can be given as $\arcsin(\sin(\theta))$, but the formulation we developed avoids trigonometry by carefully combining two discontinuous processes: taking the absolute value and round(), a function built into R that returns the nearest integer:

$$x_{\text{hex}} = x_{\text{rect}} + \left| \frac{y_{\text{rect}}}{2} - \text{round}\left(\frac{y_{\text{rect}}}{2}\right) \right|. \tag{12}$$

With these equations, we can transform points calculated our continuous grid-space surface into a coordinates suitable for plotting a hexagonal SOM as shown in Figure 20. In the somTools package, they are combined into the somTools.hexify.xy() function that also takes into account that the kohonen package shifts odd-numbered

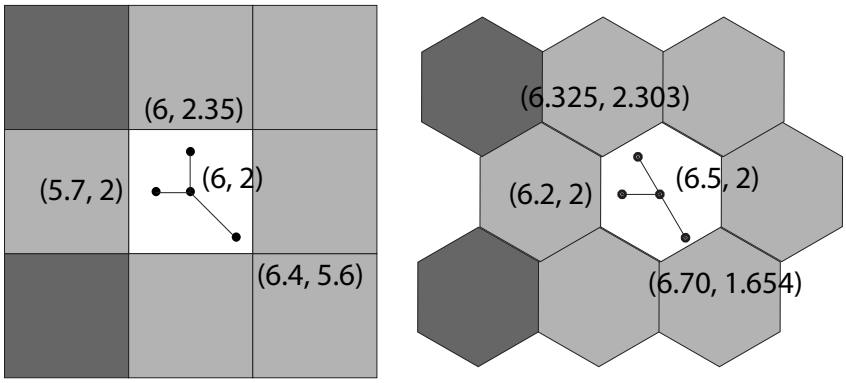

Figure 20. Transformation of rectangular grid space to hexadecimal plotting coordinates by equations (11) and (12). Example coordinate values determined in the rectangular grid space are shown on the left. On the right, these values transformed for plotting a hexagonal map. Note that the center of the node remains centered and points that had been radiating out toward adjacent node centers are still on a line from the center to those adjacent nodes.

rows to the right while the som package shifts even rows (listing in Appendix A.3).

6.4 Cartogram Construction

To construct the cartogram, we need to first compute the desired area of each node cell. As we will see in Chapter 7, this could be the data density or some other metric.

The cartogram algorithm [9, 8] requires a regular rectangular matrix of density values. For maximum efficiency, this will be a $2^a \times 2^b$ matrix and should be scaled to provide a buffer of neutral-density space around the actual map as shown in Figure 21. (The buffer helps minimize edge effects.)

For a rectangular SOM, one might be able to construct a cartogram overlay grid that exactly coincided with the areas of each SOM node. This could not be done for a hexagonal map. To support both topologies, we have adapted a method being developed in a pre-release package on R-Forge [18]. This approach uses the overlay() method of the spatial data package, sp [19], to calculate in which SOM

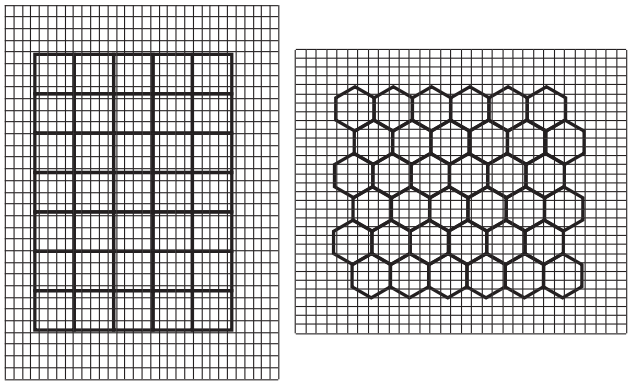

Figure 21. A relatively high-resolution rectangular grid is overlain and extended beyond the plotted area of example SOM maps to form the matrix on which the cartogram will be calculated. (SOM nodes are drawn with heavier outlines; rectangular SOM on the left; hexagonal SOM on the right.)

node each intersection on the grid resides. This creates a density matrix according to the desired area values.

Our functions somTools.node.polygon() and somTools.grid.SpatialPolygons() construct the outlines around each node and wrap those into the structure expected by the sp package. (Code listings in Appendix A.4 and A.5.) The vector of node data densities is provided by somTools.map.density() (Appendix A.6). The cartogram is constructed by somTools.grid.cartogram() (Appendix A.7).

The returned cartogram structure includes a transform function which must be applied to all plotting coordinates.

6.4.1 Computational complexity

The complexity of the inner loop of the cartogram is $O(n \log n)$—the run-time of the fast Fourier transform, where n is the number of cells in the density grid. According to Gastner and Newmann [7], the speed of this transform is dominant in the overall calculation of the cartogram; they do not offer an analysis of the outer loop which terminates when the array of points has converged.

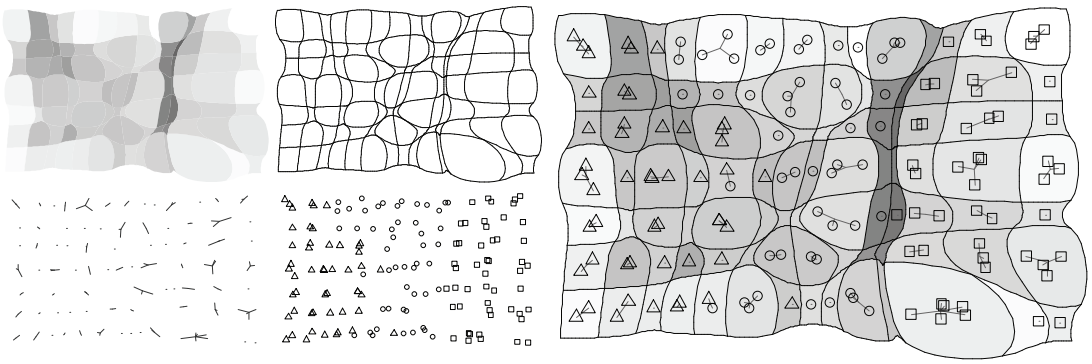

Figure 22. Grob Layers. The four smaller images on the left show the individual gTree components of the basic visualization. From upper left to bottom right, they are the gridPolys, gridEdges, tails, and points. The larger image on the right combines the layers.

6.5 Plotting the map

The grid graphics model of R [4] provides a flexible, object-oriented approach to building a display. A grob—graphics object—is one of several simple types such as a polygon or a set of points, with associated visual parameters such as color or line thickness. Grobs can be drawn individually or collected into gTree structures which can provide default parameters while still allowing overrides by individual grobs. Our basic map consists of four main gTrees, drawn as successive layers, shown separately and together in Figure 22:

- gridPolys—the individual areas representing each cell; often these are shaded or colored to show the U-matrix value, cluster membership, or some other attribute of the node. An optional subdivision parameter interpolates additional points along each edge, allowing smoother reshaping by the cartogram.

- gridEdges—the outlines of the areas representing each cell; these are usually drawn as a thin black line to subtly reinforce the regular appearance of the mesh of nodes (especially useful when the cartogram has been applied). Like the polys, the edges also support interpolation and must be interpolated in the same way as the polys if they are to align in a cartogram.

- tails—when data are mapped to interpolated locations, the optional tail connects the datum back to its node center

- points—the individual data plotted at their interpolated locations; these may use different point codes to show categorical information about the datum

List of References

[1] R. D. C. Team, *R: A Language and Environment for Statistical Computing*, Vienna, Austria, 2011, ISBN 3-900051-07-0.

[2] "R-Forge: welcome," Accessed: 2011-05-25. [Online]. Available: https://r-forge.r-project.org/

[3] R Foundation for Statistical Computing,, "The comprehensive R archive network," Accessed: 2011-05-18. [Online]. Available: http://cran.r-project.org/

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

[5] J. Yan, "som: Self-Organizing map," Accessed: 2011-06-16, 2010, R package version 0.3-5. [Online]. Available: http://CRAN.R-project.org/package=som

[6] R. Wehrens and L. Buydens, "Self- and super-organising maps in R: the kohonen package," *J. Stat. Softw.*, vol. 21, no. 5, 2007.

[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] D. Temple Lang, "Rcartogram: Interface to mark newman's cartogram software," Accessed: 2011-06-16, Nov. 2008, R package version 0.2-2. [Online]. Available: http://www.omegahat.org/Rcartogram/

[9] M. E. J. Newman, "Cart: Computer software for making cartograms," Accessed: 2011-06-17, Nov. 2006. [Online]. Available: http://www-personal.umich.edu/~mejn/cart/

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, p. 1018, Nov. 2009.

[11] J. Brownlee, "WEKA classification algorithms," Accessed: 2012-02-09, May 2011. [Online]. Available: http://sourceforge.net/projects/wekaclassalgos/

[12] R. R. Bouckaert, E. Frank, M. Hall, and R. Kirkby, "WEKA manual for version 3-7-5," Accessed: 2012-02-08, Oct. 2011, the University of Waikato. [Online]. Available: http://prdownloads.sourceforge.net/weka/WekaManual-3-7-5.pdf?download

[13] "MATLAB," The Mathworks, Inc.; Natick, Massachusetts, USA, Mar. 2012.

[14] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, "Self-organizing map in MATLAB: the SOM toolbox," CiteSeerX, Tech. Rep., 1999.

[15] T. Kohonen, *Self-Organizing Maps*, 3rd ed., Springer Series in Information Sciences. Berlin, Heidelberg, New York: Springer, 2001, no. 30.

[16] J. Chambers, *Software for Data Analysis: Programming with R*, 1st ed. Springer, July 2008.

[17] J. Chambers, "How S4 methods work," Accessed: 2010-05-13, Aug. 2006. [Online]. Available: http://developer.r-project.org/howMethodsWork.pdf

[18] T. Zumbrunn, "R-Forge: diffusion-based cartograms," Accessed: 2010-12-01, Sept. 2010. [Online]. Available: https://r-forge.r-project.org/projects/cart/

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

CHAPTER 7

Demonstrations and Experiments

We begin our demonstration by applying our visualization to the Iris data set. These data [1, 2, 3] have been models for countless studies of high-dimensional visualization, including several specific to the SOM [4, 5, 6, 7, 8]. More information about this data set is available in Appendix B.

7.1 Solutions to Data Hiding

We first create a SOM of 10×6 nodes and train it to the iris data using the kohonen package [9] (som [10] does not offer a mapping plot). A uniform distribution of 150 data through the map would result in two or three data mapped to each of the 60 nodes. The actual data density ranges from 0 to 10 per node, thus there is opportunity for data occlusion, or hiding.

In a SOM, data are usually mapped to a best-match (nearest) node. Thus in this SOM, there are up to 10 data that need to be plotted at a specific point. Jitter is a data visualization technique that distorts the true position of a datum by a small random amount.[11] When the positions are discrete instead of continuous—such as when mapping to the nearest SOM node—no information is lost. The datum is displayed within the area of the node—just not at the exact center of it where it would be difficult to distinguish from any other data mapping to the same cell. Jitter is easy and quick to calculate and it allows the density of data in a particular area to be seen.

While jitter might not lose data, it does not convey as much information as it might. Taken all together the plotted data in a cell convey only a few bits worth of additional information—the number of data in the cell. Individually, the plotted location of each datum embodies two real values—$(\Delta x, \Delta y)$ or (ρ, θ)—but jitter

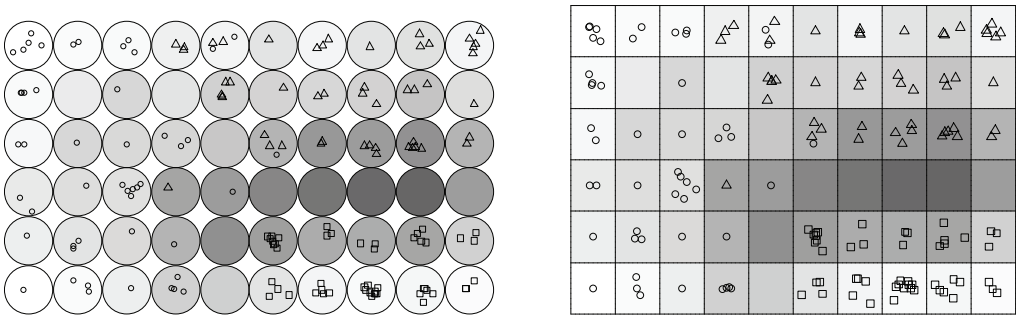

Figure 23. Jitter compared with projection. On the left, a mapping plot from the kohonen package uses jitter to accommodate multiple data in a cell. On the right, our data mapping considers similarity to neighboring cells to locate each datum. U-matrix distances are shown as gray backgrounds (darker = greater distance).

does not use these to convey any information.

In Figure 23, we compare the mapping plot from kohonen with ours, in which we adjust the location of each plotted datum to reflect its similarity to neighboring nodes. The differences in information content are subtle. The visual representation of data density is very similar to that provided by jitter.

The spread of the points around the cell conveys the node's local quantization error. The data mapping projection is calculated as a similarity to adjacent nodes only, so when the U-Matrix distances are high, it will require a greater quantization error to show the same apparent spread.

It may occur during training of a SOM that only a single datum maps to a node and the training radius has shrunk to a point where no other data will influence this node. If this occurs while sufficient iterations remain and the learning rate is still high enough, the node's codebook value will match its datum. Because the quantization error is essentially zero, such data will appear precisely centered in their node's cell. We can see this at several locations within our version of Figure 23; this effect is entirely obscured by the random jitter. It would not be correct to presume that any cell with one datum in any map has no quantization error; our visualization helps reveal this situation at a glance.

Furthermore, we anticipate that the distribution of data as plotted throughout the cell may indicate something about local $\mathbb{R}^n$ structure within the data subset that maps to a cell or to a smaller region of cells. There seem to be some cells whose data are evenly arranged; others seem to show distinct subgroups.

7.1.1 Density Cartogram

The local-neighborhood projection of data into the cell shows where we have multiple data in the cell, but these data may still be very crowded, making interpretation of any possible smaller-scale structures difficult. To relieve this crowding we apply the density-equalizing cartogram technique, enlarging node cells to which many data map and reducing the area devoted to less populous nodes. In this manner, the available space is used more efficiently, effectively zooming in on the most densely populated cells to better reveal relationships among their data.

Not always, but not uncommonly, data-dense areas of a SOM correspond to clusters and empty cells are found on the borders between clusters. Thus, clusters will tend to appear inflated and boundaries take on a pinched appearance.

7.1.2 Detection of Poorly Converged Map

When a map is not yet converged, the quantization error may be unusually high, especially where the map has not yet adapted to encompass atypical training data. Our data projection technique helps to reveal this, as shown in Figure 25 where the SOM was allowed only 20 training iterations. Several data are shown beyond the cell enclosing their best-match node; a few even project outside the map borders. This defect would not be so apparent with the jitter plotting.

7.1.3 Other Cartogram Applications

The cartogram expansion of the SOM map effectively adds an additional channel of information. The appearance of a regular SOM grid distorted by the car-

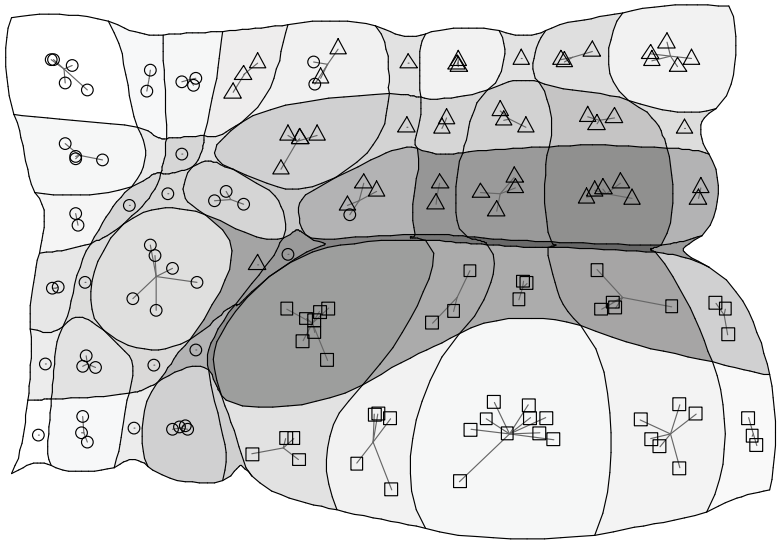

Figure 24. U-matrix shading, projected data, and cartogram expansion. The 10×6 iris SOM shown with U-matrix background shading, data mapped to projected locations, and cartogram expansion based on data density.

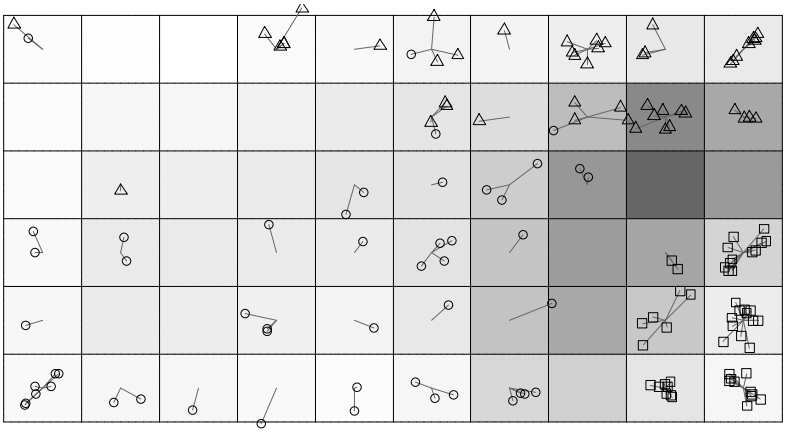

Figure 25. A poorly converged SOM. The 10×6 iris SOM shown after only 20 training iterations. The deliberately poor convergence of this SOM is indicated by data projecting far from their best-match node centers.

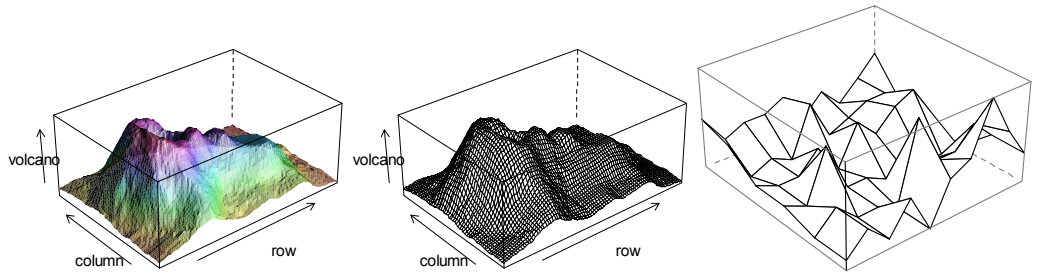

Figure 26. Wireframe used to show third dimension. On the left, a demonstration shaded rendering from the lattice package of the volcano data set (Topographic Information on Auckland's Maunga Whau Volcano) that is bundled with R.[13] Center, the shading has been removed, showing the underlying wireframe. On the right, the same package is used to render the data densities of the SOM in Figure 24. The SOM wireframe is of quite low resolution, a matrix of only 60 elements compared to the 5307 elements in the volcano dataset.

togram is similar to a 3D wireframe rendering of a surface if viewed from above. Examples of wireframes in R are provided by the lattice package[12].

In [6], Ultsch renders his U*-Matrix as a 3D contour map with hidden surface removal, showing boundaries between clusters as "mountains"—going so far as to use geographic map-maker's color scheme and white "snow" at the peaks of the boundaries (see Figure 3). If we use the cartogram expansion to show the umat distances (Figure 27), the boundaries between clusters visually seem to push the clusters away from each other, enhancing the perception of different regions of the map, though with a consequent loss of plot area in which to show data. Alternatively, we can reverse this emphasis, pinching the cluster boundaries (making them look a bit like fences or gullies) and expanding the more stable areas where clusters are found.

The cartogram effect can be based on information channels other than data density. One characteristic of the iris dataset is that two of the three species are linearly inseparable. In Figure 28, we expand the cells to which more than one species map, immediately calling the viewers attention to those nodes.

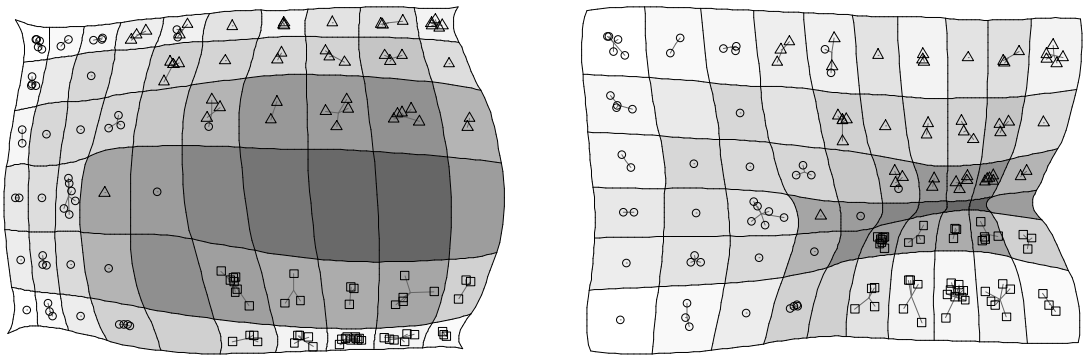

Figure 27. Cartogram based on U-Matrix (umat) distances. The (10×6) iris SOM is shown on the left with cell areas increased where the umat distances are highest. On the right, high-umat cells are shrunk. The grayscale shading of cells also shows the umat distances, as in previous figures.

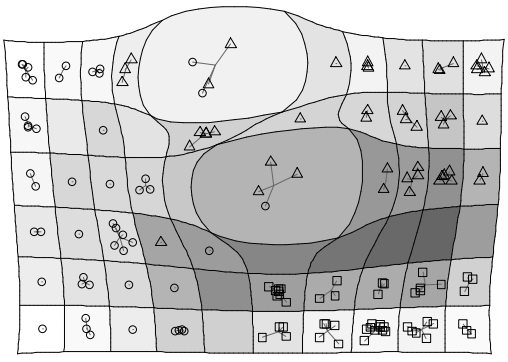

Figure 28. Cartogram based on the number of species found in a node.

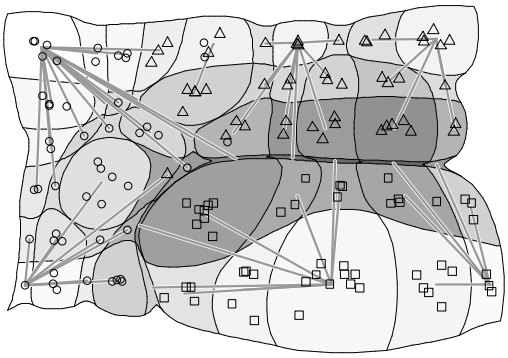

Figure 29. The connected Components clustering [14] can easily be added to this visualization. These "starburst" lines identify the node centers, making the tails of the data plots mostly redundant; they have been omitted.

7.1.4 Other information layers

The cartogram expansion offers a new visual device for encoding information in a plotted SOM. As we've seen, it does not interfere with (and may enhance) other information layers such as U-Matrix shading and data mapping.

This visualization complements the Connected Components cluster visualization [14] nicely, too. In Figure 29, we add thick lines connecting each cell to its proposed cluster center according to Hamel's method.

7.2 Further Experiments

The iris data are tremendously useful for initial exploration of new techniques, but with only 150 members and 4 attributes, they are not a good representative of the very large, very-high-dimensional data sets often encountered in contemporary research. The UCI Machine Learning Repository [15] provides a variety of more extensive data.

We selected the Cardiotocography [16] data set for further experimentation. A cardiotocogram is a recording of both uterine contractions and fetal heartbeat [17] used in obstetrics. This data set contains results of 2126 examinations each with 21 measured real- or integer-valued attributes plus two additional classification at-

tributes assigned by the consensus of three expert obstetricians. One classification attribute indicates one of ten classes of event being observed such as "calm sleep" or "decelerative pattern." The second classification attribute is a risk measure: "normal," "suspect," or "pathologic."

After normalizing the 21 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. A U-Matrix plot of this SOM (Figure 30, left) does not show any clear pattern of clustering; there are two pockets of higher umat distance but no clear divisions between regions. A data density cartogram expansion (Figure 30, right) 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 of only about 1.8.

We obtained a much more interesting plot (Figure 31) 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. 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 30); it is dominated by "largely decelerative" (8) and"decelerative' (7) fetal state classes. As with the risk assessment, the state codes were not used to train the SOM.

7.2.1 Time and Resources Required

To assess whether these techniques can be adopted in practice, we employed the system.time() function in R to find the user time taken by the various computations required to produce Figure 31. The primary system available for testing and development is a commercially-available notebook computer running 64-bit R

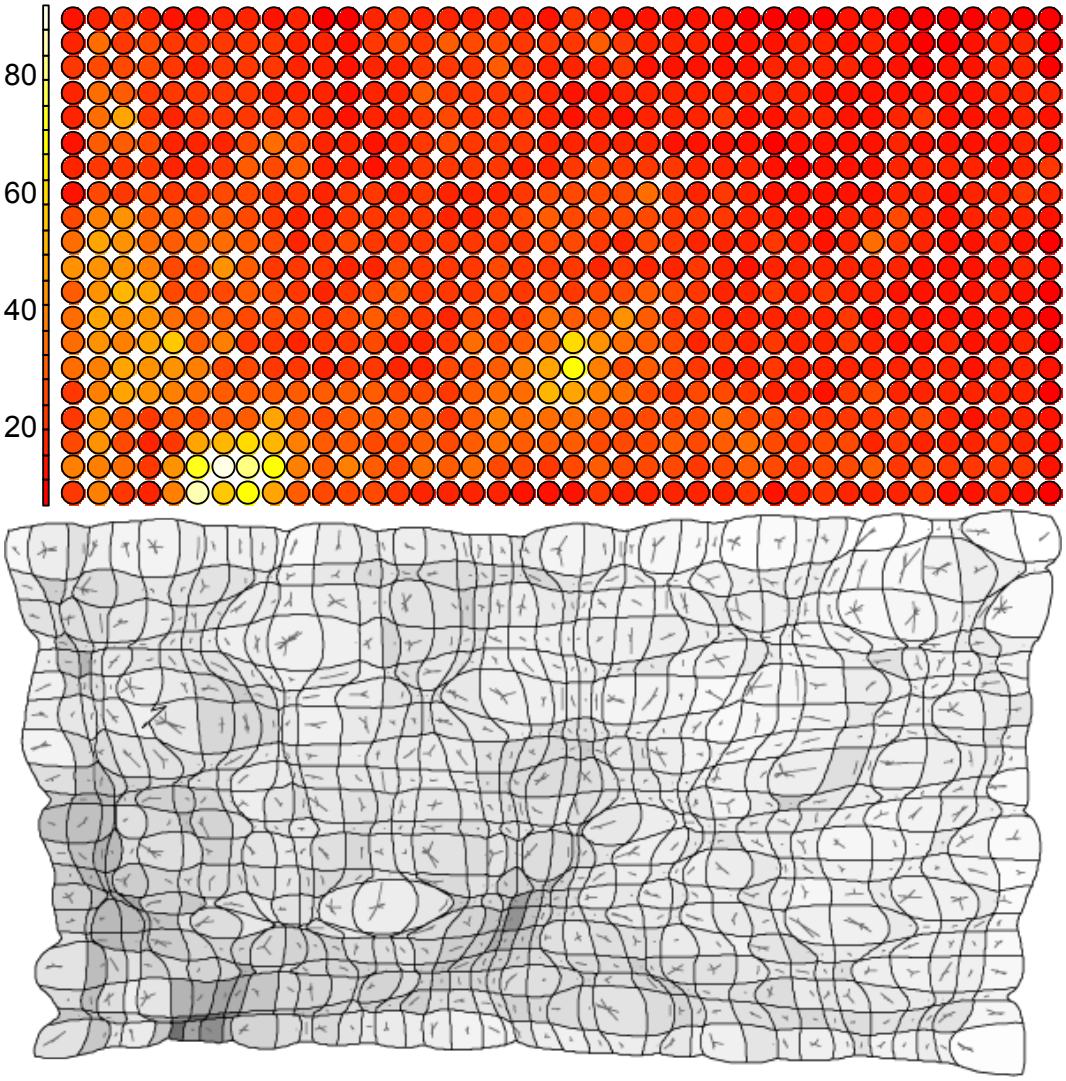

Figure 30. A SOM of the Cardiotocography data set. The standard dist.neighbours (U-matrix) plot from the kohonen package is above; below is our visualization. We include data tails to show quantization error and adjust cell size to show data density. The data set appears fairly homogeneous, with no strongly separated clusters nor any extreme variations in data density.

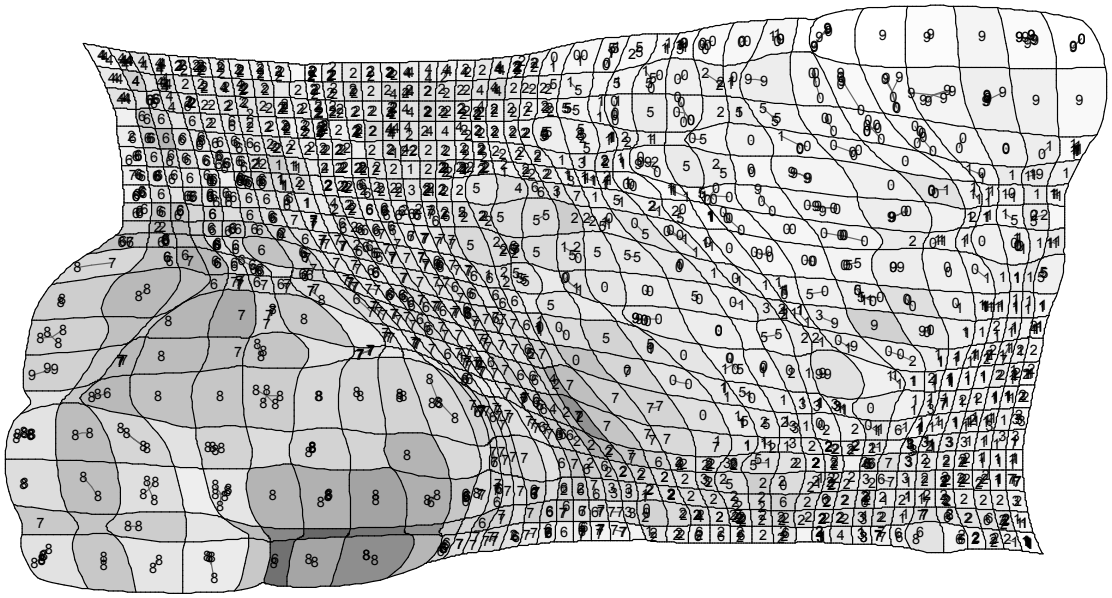

Figure 31. A SOM of the Cardiotocography data set where the area of each cell represents the average risk classification of examinations mapping to that node. Examinations are assessed by consensus of three obstetricians as 1=normal (smallest cells); 2=suspect; or 3=pathologic (largest cells). Data labels are the fetal state class codes; 1 is "A" (calm sleep), 2="B" (REM sleep), 3="C" (calm vigilance), 4="D" (active vigilance), 5="SH" (shift pattern), 6="AD" (accelerative/decelerative; stress), 7="DE" (decelerative), 8="LD" (largely decelerative), 9="FS" (flat-sinusoidal), and 0="susp" (suspect).

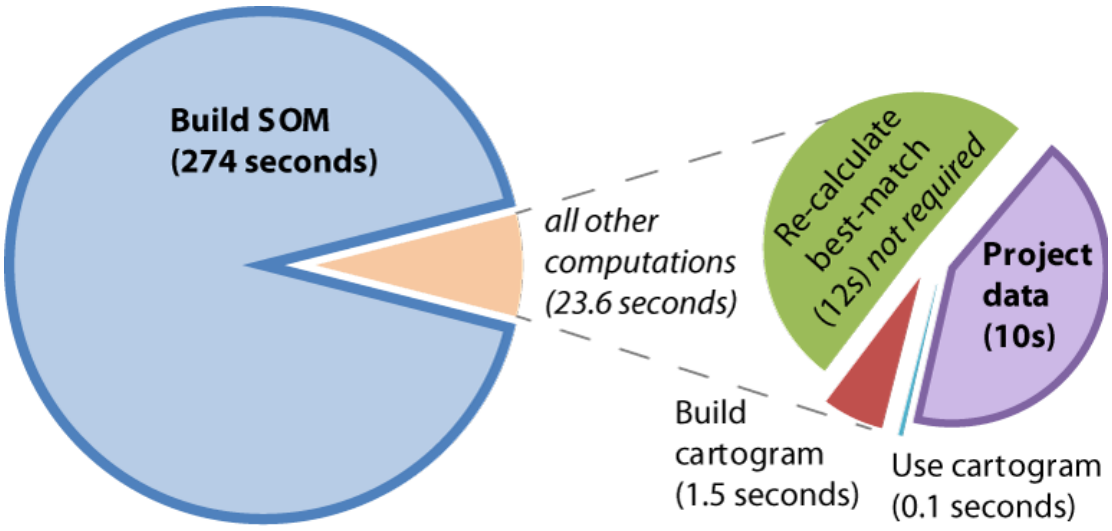

Figure 32. User time required to calculate a large SOM visualization (Figure 31). The SOM calculation itself dwarfs the time required for other computations.

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 at 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.

The kohonen package required 274 seconds to complete 5000 iterations. This is by far the most costly computation task required, as shown in Figure 32. Calculation of the cartogram on a 256×256 grid was unexpectedly rapid, requiring only about 1.5 seconds. Calculating the polygons that fill the 800 node cells with 9 subdivisions within each of the 4 sides required about 0.3 seconds as did calculating the matching edges used to trace their outline. Calculating the projected location within the cell took about 10 seconds; additional time to recalculate the best-match node increased that to 22 seconds. Recalculating the best-match node was not actually required for the training data; this unnecessary step accounted for half of the non-SOM computation time. Transforming all the node centers, data locations, node edges, and node polygons according to the cartogram was barely measurable, requiring only about 0.1 seconds.

It seems clear that if it is practical to calculate the SOM itself, the complexity of this visualization will not be problematic.

List of References

[1] E. Anderson, "The irises of the gaspe peninsula," *Bulletin of the American Iris society*, no. 59, pp. 2–5, 1935.

[2] J. C. Bezdek, J. M. Keller, R. Krishnapuram, L. I. Kuncheva, and N. R. Pal, "Will the real iris data please stand up?" *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 3, pp. 368–369, 1999.

[3] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen*, vol. 7, no. Part II, pp. 179–88, 1936.

[4] M. A. Kraaijveld, J. Mao, and A. K. Jain, "A nonlinear projection method based on Kohonen's topology preserving maps," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 548–559, May 1995.

[5] D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, G. Plzlbauer, A. Rauber, and M. Dittenbach, "A vector field visualization technique for self-organizing maps," in *Advances in Knowledge Discovery and Data Mining*, T. B. Ho, D. Cheung, and H. Liu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, vol. 3518, pp. 399–409.

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

[7] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, "Self-organizing map in MATLAB: the SOM toolbox," CiteSeerX, Tech. Rep., 1999.

[8] H. Yin, "Data visualisation and manifold mapping using the ViSOM," *Neural Networks: The Official Journal of the International Neural Network Society*, vol. 15, no. 8-9, pp. 1005–1016, Nov. 2002, PMID: 12416690.

[9] R. Wehrens and L. Buydens, "Self- and super-organising maps in R: the kohonen package," *J. Stat. Softw.*, vol. 21, no. 5, 2007.

[10] J. Yan, "som: Self-Organizing map," Accessed: 2011-06-16, 2010, R package version 0.3-5. [Online]. Available: http://CRAN.R-project.org/package=som

[11] M. O. Ward, "A taxonomy of glyph placement strategies for multidimensional data visualization," *Information Visualization*, vol. 1, no. 3-4, pp. 194–210, Dec. 2002.

[12] D. Sarkar, *Lattice: Multivariate Data Visualization with R.* New York: Springer, 2008, ISBN 978-0-387-75968-5.

[13] R. D. C. Team, *R: A Language and Environment for Statistical Computing*, Vienna, Austria, 2011, ISBN 3-900051-07-0.

[14] 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: CSREA Press, July 2011, pp. 338–343.

[15] A. Frank and A. Asuncion, "UCI machine learning repository," Accessed: 2011-02-18, 2010, university of California, Irvine, School of Information and Computer Sciences. [Online]. Available: http://archive.ics.uci.edu/ml

[16] 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.

[17] "Cardiotocography - Wikipedia, the free encyclopedia," Accessed: 2012-02-27. [Online]. Available: http://en.wikipedia.org/wiki/Cardiotocography

CHAPTER 8

Conclusions and Future Work

With this project, we have achieved our primary goal of increasing the visual area available for plotting data within a Self-Organizing Map. The two techniques used—the cartogram and locating plotted data within the cell in a meaningful way—have shown their usefulness in unanticipated contexts beyond what we expected.

The cartogram can convey an additional layer of information, encouraging the viewer to focus attention on areas of a map of particular interest while also expanding those areas to better see data detail. Examples include the inseparable area of the Iris species (Figure 28) or the higher-risk pregnancies of the Cardiotocography examinations (Figure 31).

Projecting the data according to its relationship to the neighboring nodes allows us to see the local quantization error which may also indicate how well the SOM has converged (Figure 25).

8.1 Future Work

8.1.1 Support Other SOM algorithms

The R community continues to develop SOM implementations (e.g., [1]) with unique features and capabilities which we would like to be able to use with this visualization. The key task is to extend the "initialize" method of the package to recognize additional SOM classes and copy their data into the appropriate slots.

8.1.2 Handle missing data

As we reviewed existing algorithms in R, we noticed that some of them included additional code to handle problems with the input data such as missing values. This was not a concern with the somewhat idealized data sets we used in

developing these techniques, but it would be a concern for their wider application. Accordingly, it would be appropriate to incorporate better handling of missing data into our routines.

8.1.3 Projection scaling

As described in section 5.4, the orthogonal projections of each datum from its best-match node toward neighboring nodes must be scaled. The equations currently employed seem to adequately handle most reasonably well-converged maps but may be somewhat conservative. We note that the data projections infrequently fill the cell, even in "smooth" areas of the map where unit distances between nodes are low. If it were appropriate to expand the area of the cell used, this would aid in avoiding data occlusion.

8.1.4 Optimization of calculations

At present, our calculations seem reasonably efficient and practical for fairly substantial data sets on a mid-to-high-end computer. It is unclear that effort to make them faster would produce significant improvements for the end-user. Nonetheless, we have noticed some areas that could be enhanced.

Lists of matrices Most of our algorithms have been adapted to use R's vectorization and it is unlikely that interpreted code could run faster. There are still a few instances where we have to iterate through a list of matrices. It might be possible to reconsider how those data structures are implemented and fully vectorize them.

Mesh of subdivided polygons At present, we are passing the 9×-subdivided polygon outlines to the sp package's overlay method for calculating the cartogram density mesh. It should be possible to calculate this map on the undivided polygons

and so speed the calculation of the mesh. Note, though, that the entire cartogram construction is accomplished in a matter of seconds; this task is a relatively small piece of that process.

Best-Match While optimizing the SOM algorithm itself is entirely outside the scope of the present project, it could not escape our notice that the SOM implementation used was unable to benefit from more than a single CPU core, churning away for minutes while the CPU reported only a 12% or 25% workload (depending on whether "hyperthreading" was enabled, allowing each core to function as two virtual cores). Continuing research projects investigate how to parallelize the complete SOM algorithm, especially how to coordinate simultaneous map updates.[2, 3, 4] However, the inner loop of the SOM algorithm—finding the best-matching node for the current training datum—could much more easily be done in parallel. Each thread would search an equal slice of the map and return its best candidate and distance to determine the ultimate winner. Such algorithms exist;[5] it could be useful to bring them to the R community.

8.1.5 Further visualization techniques

Where appropriate, we will continue to add additional capabilities to the somTools package. Preliminary code to outline contiguous regions of nodes (e.g., clusters) has already been written. To provide feature parity with other packages, a jitter data projection may be useful in some situations as could be a routine to stack data labels around a node center.

A mechanism to place a small figure in each cell, representing the data-space values of that node, would be a useful addition.

List of References

[1] F. Rossi, "R-Forge: yasomi: Self-Organising Map in R: Project home," Accessed: 2012-02-27, Feb. 2011. [Online]. Available: https://r-forge.r-project.org/projects/yasomi/

[2] D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, M. Takatsuka, and M. Bui, "Parallel batch training of the Self-Organizing map using OpenCL," in *Neural Information Processing. Models and Applications*, K. W. Wong, B. S. U. Mendis, and A. Bouzerdoum, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6444, pp. 470–476.

[3] M. H. Yang and N. Ahuja, "A data partition method for parallel self-organizing map," in *International Joint Conference on Neural Networks*, vol. 3. IEEE, 1999, pp. 1929–1933 vol. 3.

[4] B. Silva and N. Marques, "A hybrid parallel SOM algorithm for large maps in data-mining," in *Proceedings of the Portuguese Conference on Artificial Intelligence*, 2007.

[5] G. Myklebust and J. G. Solheim, "Parallel self-organizing maps for actual applications," in *International Conference on Neural Networks*, vol. 2. IEEE, 1995, pp. 1054–1059 vol. 2.

APPENDIX A

Code Listings

(Automatic line breaks have been inserted where necessary to fit thesis format requirements.)

A.1 Description of the somTools class

In: somTools/R/AllCasses.R

```
#from somTools
#Copyright 2012 David H. Brown
#GPL license available
#' Class definition for somTools
#'
#'
setClass("somTools",
  representation(
    topol = "character",
    toroidal = "logical",
    xdim = "numeric",
    ydim = "numeric",
    #whatever the interfacing function says
    data = "ANY",
    #value in n-space at each node
    codes = "matrix",
    scalecodes = "matrix", #a saved copy of scale(codes)
    #list of nodeIdexes for each observation
    #ANY becuase this is numeric or NULL
    datacode = "ANY", #like kohonen$unit.classif

    #' original.class records the type of SOM that
       initialized this somTools
    original.class = "character"
    ) #representation
    ,
    #validity parameter creates the validObject() method
    validity=function(object) {
  messages <- NULL #by default

  #simple known problems:
  if (is.null(object@codes)) {
```

```
    messages<-c(messages, "Codes not known (@codes) -- has
        the map been trained?")
    }
  if (is.null(object@topol)) {
    messages<-c(messages, "Topology not known (@topol)")
    }
  if (is.null(object@original.class)) {
    messages<-c(messages, "Original class not known (
        @original.class)")
    }
  if (object@xdim < 1) {
    messages<-c(messages, "X-dimension must be at least 1 (
        @xdim)")
    }
  if (object@ydim < 1) {
    messages<-c(messages, "Y-dimension must be at least 1 (
        @ydim)")
      }
  #did we pass all the tests?
  if (is.null(messages)) { TRUE }
  #or do we have problems?
  else { messages }
} #validity function
  ) #setClass
```

A.2 Initialization of the somTools object

In: somTools/R/AllGenerics.R

```
#' initialize a somTools;
#' if passed a \link{som} or \link{kohonen} object,
#' will assign values to appropriate slots
setMethod("initialize","somTools",
  function(.Object,anysom,...) {
    if(! missing(anysom)) {
if (class(anysom) == "kohonen") {
          .Object@original.class = "kohonen"
          .Object@data <- anysom$data
          .Object@topol <- substr(anysom$grid$topo,1,4)
          .Object@toroidal <- anysom$toroidal
          .Object@xdim <- anysom$grid$xdim
          .Object@ydim <- anysom$grid$ydim
          .Object@datacode <- anysom$unit.classif
          .Object@codes <- anysom$code
          .Object@scalecodes <- scale(.Object@codes)
```

```
      } #if kohonen

if (class(anysom) == "som") {
        .Object@original.class = "som"
        .Object@topol <- anysom$topol
        .Object@toroidal <- FALSE
        .Object@data <- anysom$data #som::som always keeps
            data
        .Object@xdim <- anysom$xdim
        .Object@ydim <- anysom$ydim
        #som uses x+y*xdim; which starts at 0; our
            functions assume nodeIndex starts at 1:
        .Object@datacode <- (anysom$visual[,1]+(anysom$xdim
            * anysom$visual[,2]) + 1)
        .Object@codes <- anysom$code
        .Object@scalecodes <- scale(.Object@codes)
  } #if som
## Add future classes of self-organizing map here

## If we did not find a class, anysom probably wasn't meant
    for us and should
## be passed to the next method?
if (is.null(.Object@original.class)) {
  classNextMethod(.Object,anysom,...)
} else {
       callNextMethod(.Object,...)
    }
      .Object
  } #if anysom not missing
  else {callNextMethod(.Object,...)}
  } #function
  ) #setMethod initialize
```

A.3 somTools.hexify.xy() – transform rectangular grid points to hexagonal

In: somTools/R/somTools.hexify.xy.R

```
#from somTools
#Copyright 2012 David H. Brown
#GPL license available
#scale data coordinates from square xy to align with hexes
   (if appropriate to map)
#currently takes lists of x, y or a matrix for x ; this may
    change?
```

```
#returns a 2-column matrix
somTools.hexify.xy <- function(somTools, x, y) {
#check parameters:
  if(!is(somTools, "somTools")) {
    stop("Must have a somTools as first argument.")
  }
if(missing(y)) {#x better be a matrix, then
  y <- x[,2]
  x <- x[,1]
} #missing y
if(length(x) != length(y)) {
  stop('Length of x must match length of y')
  }

  if (identical(somTools@topol,'rect')) {
    return(matrix(c(x,y), ncol=2, dimnames=list(NULL,c('x',
       'y')))) #unchanged as matrix
    }


  #we'll add 1 to y to account for maps which shift odd
     rows
  if(somTools@original.class %in% c('kohonen')) {
    ys <- 1+y
    } else {
    ys <- y
    }

  #here we calculate a zig-zag offset based on the y value
  xadj <- abs(ys/2-round(ys/2))

  #return value:
  matrix(c(x+xadj, y * sqrt(3)/2), ncol=2, dimnames=list(
     NULL,c('x','y')) )
}
```

A.4 somTools.node.polygon() – outline node plot areas

In: somTools/R/somTools.node.polygon.R

```
#from somTools
#Copyright 2012 David H. Brown
#GPL license available
# returns a list[[node.index]] where each entry is a n*2
    matrix
```

```
# listing x/y points that surround the node
# the x/y points HAVE been hexified
# sp:Polygon requires the polygon to be closed=TRUE
# @TODO: move the test for the original class and which hex
    row shifts
#       to be a class property and set during
    initialization
somTools.node.polygon <- function(somTools, node.index,
    closed=FALSE) {
  if(!is(somTools, "somTools")) {
    stop("Must have a somTools as first argument.")
  }

  if(missing(node.index)) node.index <- (1:(somTools@xdim*
      somTools@ydim))


  xy <- somTools.nodeIndex.to.xy(somTools,node.index)
  x <- xy[, 'x']
  y <- xy[, 'y']
  #we can form a monster list l of x for all node.index, y
      for all node.index
  #and then reassemble it as needed in a matrix

  l <- c( node.index ) #keep track to identify entries in
      list

  if(identical(somTools@topol,'rect')) {
  #let's handle just rectangular grids first...
  l <- c(l,
    x-0.5, y-0.5, #ll
    x-0.5, y+0.5, #ul
    x+0.5, y+0.5, #ur
    x+0.5, y-0.5) #lr
  if(closed) { l <- c(l,x-0.5, y-0.5) }
  }

  if(identical(somTools@topol,'hexa')) {
  #hexagon grids are arranged such that rows are on a line
      and columns stagger
  #For the distance between these parallel sides to be
      equal to 1.0, the
  #length of the side must be sqrt(3)/3.
```

```
  #For kohonen maps, the 1st (bottom) and other "odd" rows
     are offset to the right
  #For som hex maps, the 2nd-from-bottom and "even" rows
     are offset to the right
  sq3 <- sqrt(3) #not really saving time by precomputing
     these, but it's a
  sq33 <- sq3/3  #maybe a little clearer in the code below?
  sq36 <- sq3/6

  if(identical(somTools@original.class,'kohonen')) {
    x <- x+((1-(y %%2)) * .5)
    } else {
    x <- x+((y %% 2) * .5)
  }

  y <- y*(sq3/2) #scaly y height to nest

  l <- c(l,
    x, y-sq33, #bottom
    x-0.5, y-sq36, #lower-left
    x-0.5, y+sq36, #upper-left
    x, y+sq33, #top
    x+0.5, y+sq36, #upper-right
    x+0.5, y-sq36) #lower-right
  if(closed) { l <- c(l,x, y-sq33) }
  }
  m <- matrix(l, nrow=length(node.index)   ) #matrix
  l <- list()
  for (i in 1:length(node.index)) {
    l[[m[i,1]]] <- matrix(m[i,-1],ncol=2,byrow=TRUE,dimname
       =list(NULL,c('x','y')))
    }
  l
} #somTools.node.polygon
```

A.5 somTools.grid.SpatialPolygons() – prepare node oulines for cartogram

In: somTools/R/somTools.grid.SpatialPolygons.R

```
#from somTools
#Copyright 2012 David H. Brown
#GPL license available
##' create a sp:Polygons object containing all the polygons
    in the map.
```

```
somTools.grid.SpatialPolygons <- function(somTools) {
  if(!is(somTools, "somTools")) {
    stop("Must have a somTools as first argument.")
  }
  xys <- somTools.node.polygon(somTools,closed=TRUE)
  polys <- list()
  for (i in 1:length(xys)) {
    polys[[i]] <- Polygons(list(Polygon(xys[[i]])), ID=i)
    }
  SpatialPolygons(polys, pO=1:length(xys))
}
```

A.6 somTools.map.density() – calculate data density of SOM

In: somTools/R/somTools.map.density.R

```
#from somTools
#Copyright 2012 David H. Brown
#GPL license available
##' somTools.map.density
##'
##' returns a vector of data count for each grid cell
##' if newdata is provided, its nearest nodes will be
   calculated
##' otherwise, the datacode or data values stored in the
   somTools
##' will be used.
##'
##' @param somTools somTools
##' @param newdata numeric
##' @param scalefun function

somTools.map.density <- function(somTools, newdata,
   scalefun = function(vec) {vec}) {
    if(!is(somTools, "somTools")) {
    stop("somTools.map.density must have a somTools as
       first argument.")
    }
#if newdata is missing, we can try to use training data
   from somTools@datacode or @data
if(missing(newdata)) {
    if(is.null(somTools@datacode)) {
      if(is.null(somTools@data)) { #neither data nor
         datacode
```

```
        stop("This somTools does not have either data nor
            datacode; cannot infer density; you must specify
            newdata")
        } else { #we do have data but no datacode
          nearest <- somTools.nearest.node(somTools,
            somTools@data)
          } #else we do have data
      } else { #we do have datacode
        nearest <- somTools@datacode
        } #else we do have datacode
  } else { #we do have newdata
  newdata <- somTools.newdata.massage(somTools, newdata)
  if(identical(FALSE,newdata)) {
    stop("somTools@codes and newdata must match; see
      warnings.")
  }
  nearest <- somTools.nearest.node(somTools, newdata)
} #else we do have newdata

#at this point, all we want is the vector tab, with one
   density value per nodeIndex
somTools.nodeIndexVector.to.countVector(somTools, nearest)
}
```

A.7 somTools.grid.cartogram() – construct the cartogram

In: somTools/R/somTools.grid.cartogram.R

```
#from somTools
#Copyright 2012 David H. Brown
#GPL license available
##' obtain a cartogram on somTools according to areas given
    as vector, one-per-cell.

##' some code relating to working with the sp package is
   thanks to
##' (or at least inspired by) Thomas Zumbrumm; see a
   currently unfinished
##' project at: https://r-forge.r-project.org/projects/cart
   /
somTools.grid.cartogram <- function(somTools, areas, sea.
   expansion=0.2, grid.dimensions=c(128,128), blur=0) {
  if(!is(somTools, "somTools")) {
    stop("Must have a somTools as first argument.")
  }
```

```
if(missing(areas)) {
  #include some minimum area to avoid squishing a cell to
      nothing
  areas <- (somTools.map.density(somTools) + 0.75) ^ 1.5
     #squaring it seemed too extreme
  }
if(length(areas) != somTools@xdim*somTools@ydim) {
  warning("Areas should be equal to the number of cells
     in the SOM")
  }
if (length(grid.dimensions) == 1) grid.dimensions <- rep(
    grid.dimensions,2)
if (length(grid.dimensions) != 2) {
  warning("Should specify grid.dimensions as a vector of
     2 numerics (x,y)")
  }
if (length(grid.dimensions) < 2) stop("Unusable grid.
    dimensions.")

if (!isTRUE(all.equal(log(grid.dimensions, 2), floor(log(
    grid.dimensions, 2))))) {
  warning("Set grid.dimensions to powers of 2 for faster
     calculation")
  }

spolys <- somTools.grid.SpatialPolygons(somTools)

#the cartogram process wants a fairly large "sea" of
    space around the
#actual map area so as to avoid edge effects
#note: the cartogram function can do this itself...
range <- diff(t(spolys@bbox))
shift <- range * sea.expansion
bbox.expanded <- spolys@bbox + c(-shift, shift)
range.expanded <- diff(t(bbox.expanded))
spgrid <- SpatialGrid(
          GridTopology(
            cellcentre.offset = as.numeric(bbox.expanded
               [, "min"]),
            cellsize = as.numeric(range.expanded / (grid.
               dimensions - 1)),
            cells.dim = grid.dimensions #
               spgrid@grid@cells.dim
          )#GridTopology #spgrid@grid
```

```
        )#SpatialGrid

## overlay grid and polygons

## This is an extension of the point-in-polygon problem.
   We obtain a vector of
## indices of the polygons in spdf.
ind <- overlay(spgrid, spolys)

## calculate "density"

## For each grid cell, we need to determine the fraction
   of the units of
## "variable" as the number of units per cell. For NAs, i
   .e. for the "sea",
## insert the mean value for the whole "land" mass. For
   the tabulation, the
## levels need to be enforced because there might be
   polygons with count zero.
## For these cells, division by zero is corrected by
   replacing the resulting
## infinite result by zero.
tab <- xtabs(~ factor(ind, levels = seq(along =
   spolys@polygons)))
indVar <- areas[as.numeric(names(tab))] / tab
indVar[is.infinite(indVar)] <- 0
tmean <- sum(tab * indVar[as.numeric(names(tab))]) / sum(
   tab)
ind[is.na(ind)] <- length(areas) + 1
indVar[length(areas) + 1] <- tmean

dens <- matrix(indVar[ind], byrow = TRUE, ncol = grid.
   dimensions[2])

cart <- cartogram(dens, blur=blur)

#now we need to add some extra information to the
   cartogram
#first, the bounding box of the expanded grid this
   cartogram represents
cart$bbox <- bbox.expanded
cart$cellcentre.offset <- spgrid@grid@cellcentre.offset
cart$cellsize <- spgrid@grid@cellsize
cart$cells.dim <- spgrid@grid@cells.dim
```

```
  #we're going to include a function; functions bring along
      their environment,
  #so we'll remove largish objects we no longer need:
  rm('spgrid','spolys','areas','tab','indVar','ind','dens')
  #the transform function scales data from the original
     space to the
  #cartogram grid.dimensions space
  cart$transform <- function(x, y = NULL) {
    if(missing(y) || is.null(y)) {
     y = x[,2]
     x = x[,1]
    }
    x <- (x-cart$cellcentre.offset[1]) / cart$cellsize[1]
    #we need to flip the y axis to match the cartogram
       structure with [1,1] at UL:
    y <- cart$cells.dim[2]- ( (y-cart$cellcentre.offset[2])
        / cart$cellsize[2] )

    tmp <- predict(cart, x, y)
    #and we need to flip the y axis back to match how
       things plot with (0,0) at LL:
    tmp$y <- cart$cells.dim[2]-tmp$y

    matrix(c(tmp$x * cart$cellsize[1] + cart$cellcentre.
       offset[1],
             tmp$y * cart$cellsize[2] + cart$cellcentre.
                offset[2])
           , ncol=2, dimnames=list(NULL,c('x','y')))
  }# cart$transform

  return(cart)
}
```

APPENDIX B

Iris Data

Anderson's iris data, first published by Fisher in 1936, [1] have been the subject of innumerable analyses and experiments. The data are mentioned (but not reported) in various articles by Anderson, especially *Iris of the Gaspé Peninsula*[2] in which he does seem to recognize the enduring interest of his "juicy morsel" of data which provides two of the three species in the iris data set:

> I have found no other opportunity quite like the field from Ile Verte to Trois Pistoles.[1] There for mile after mile one could gather irises at will and assemble for comparison one hundred full-blown flowers of *Iris versicolor* and of *Iris setosa canadensis*, each from a different plant, but all from the same pasture, and picked on the same day and measured at the same time by the same person with the same apparatus. The result is, to ordinary eyes, a few pages of singularly dry statistics, but to the biomathematician a juicy morsel quite worth looking ten years to find.

In *The Species Problem in Iris*[3], Anderson presents the case that *Iris Versicolor* is a hybrid between the *Setosa* and *Virginica* species based on several considerations, including the morphological characteristics of the petal and sepal length and width (Figure B.33).

Interestingly, Anderson notes that because *Virginica* has twice as many chromosomes as *Setosa*, he expects and finds *Versicolor* to be substantially closer to *Virginica*—this similarity is very familiar to us as *Versicolor* and *Virginica* are the difficult-to-separate species. The iris data were published by Fisher to demonstrate his method of linear discriminant functions.[1] [4].

Iris sestosa				*Iris versicolor*				*Iris virginica*			
Sepal		Petal		Sepal		Petal		Sepal		Petal	
Len.	Wid.	Len.	Wid.	Len.	Wid.	Len.	Wid.	Len.	Wid.	Len.	Wid.
5.1	3.5	1.4	0.2	7.0	3.2	4.7	1.4	6.3	3.3	6.0	2.5
4.9	3.0	1.4	0.2	6.4	3.2	4.5	1.5	5.8	2.7	5.1	1.9
4.7	3.2	1.3	0.2	6.9	3.1	4.9	1.5	7.1	3.0	5.9	2.1
4.6	3.1	1.5	0.2	5.5	2.3	4.0	1.3	6.3	2.9	5.6	1.8
5.0	3.6	1.4	0.2	6.5	2.8	4.6	1.5	6.5	3.0	5.8	2.2
5.4	3.9	1.7	0.4	5.7	2.8	4.5	1.3	7.6	3.0	6.6	2.1
4.6	3.4	1.4	0.3	6.3	3.3	4.7	1.6	4.9	2.5	4.5	1.7
5.0	3.4	1.5	0.2	4.9	2.4	3.3	1.0	7.3	2.9	6.3	1.8

[1]The Gaspé peninsula (officially, the *Gaspésie*) forms the southern shore of the Saint Lawrence River in the Canadian province of Quebec. The town of L'Isle-Verte is about 230km northeast of Quebec City. A national wildlife area extends along the Saint Lawrence about half-way toward Trois-Pistoles which is about 20km further northeast.

Iris sestosa				*Iris versicolor*				*Iris virginica*			
Sepal		Petal		Sepal		Petal		Sepal		Petal	
Len.	Wid.	Len.	Wid.	Len.	Wid.	Len.	Wid.	Len.	Wid.	Len.	Wid.
4.4	2.9	1.4	0.2	6.6	2.9	4.6	1.3	6.7	2.5	5.8	1.8
4.9	3.1	1.5	0.1	5.2	2.7	3.9	1.4	7.2	3.6	6.1	2.5
5.4	3.7	1.5	0.2	5.0	2.0	3.5	1.0	6.5	3.2	5.1	2.0
4.8	3.4	1.6	0.2	5.9	3.0	4.2	1.5	6.4	2.7	5.3	1.9
4.8	3.0	1.4	0.1	6.0	2.2	4.0	1.0	6.8	3.0	5.5	2.1
4.3	3.0	1.1	0.1	6.1	2.9	4.7	1.4	5.7	2.5	5.0	2.0
5.8	4.0	1.2	0.2	5.6	2.9	3.6	1.3	5.8	2.8	5.1	2.4
5.7	4.4	1.5	0.4	6.7	3.1	4.4	1.4	6.4	3.2	5.3	2.3
5.4	3.9	1.3	0.4	5.6	3.0	4.5	1.5	6.5	3.0	5.5	1.8
5.1	3.5	1.4	0.3	5.8	2.7	4.1	1.0	7.7	3.8	6.7	2.2
5.7	3.8	1.7	0.3	6.2	2.2	4.5	1.5	7.7	2.6	6.9	2.3
5.1	3.8	1.5	0.3	5.6	2.5	3.9	1.1	6.0	2.2	5.0	1.5
5.4	3.4	1.7	0.2	5.9	3.2	4.8	1.8	6.9	3.2	5.7	2.3
5.1	3.7	1.5	0.4	6.1	2.8	4.0	1.3	5.6	2.8	4.9	2.0
4.6	3.6	1.0	0.2	6.3	2.5	4.9	1.5	7.7	2.8	6.7	2.0
5.1	3.3	1.7	0.5	6.1	2.8	4.7	1.2	6.3	2.7	4.9	1.8
4.8	3.4	1.9	0.2	6.4	2.9	4.3	1.3	6.7	3.3	5.7	2.1
5.0	3.0	1.6	0.2	6.6	3.0	4.4	1.4	7.2	3.2	6.0	1.8
5.0	3.4	1.6	0.4	6.8	2.8	4.8	1.4	6.2	2.8	4.8	1.8
5.2	3.5	1.5	0.2	6.7	3.0	5.0	1.7	6.1	3.0	4.9	1.8
5.2	3.4	1.4	0.2	6.0	2.9	4.5	1.5	6.4	2.8	5.6	2.1
4.7	3.2	1.6	0.2	5.7	2.6	3.5	1.0	7.2	3.0	5.8	1.6
4.8	3.1	1.6	0.2	5.5	2.4	3.8	1.1	7.4	2.8	6.1	1.9
5.4	3.4	1.5	0.4	5.5	2.4	3.7	1.0	7.9	3.8	6.4	2.0
5.2	4.1	1.5	0.1	5.8	2.7	3.9	1.2	6.4	2.8	5.6	2.2
5.5	4.2	1.4	0.2	6.0	2.7	5.1	1.6	6.3	2.8	5.1	1.5
4.9	3.1	1.5	0.2	5.4	3.0	4.5	1.5	6.1	2.6	5.6	1.4
5.0	3.2	1.2	0.2	6.0	3.4	4.5	1.6	7.7	3.0	6.1	2.3
5.5	3.5	1.3	0.2	6.7	3.1	4.7	1.5	6.3	3.4	5.6	2.4
4.9	3.6	1.4	0.1	6.3	2.3	4.4	1.3	6.4	3.1	5.5	1.8
4.4	3.0	1.3	0.2	5.6	3.0	4.1	1.3	6.0	3.0	4.8	1.8
5.1	3.4	1.5	0.2	5.5	2.5	4.0	1.3	6.9	3.1	5.4	2.1
5.0	3.5	1.3	0.3	5.5	2.6	4.4	1.2	6.7	3.1	5.6	2.4
4.5	2.3	1.3	0.3	6.1	3.0	4.6	1.4	6.9	3.1	5.1	2.3
4.4	3.2	1.3	0.2	5.8	2.6	4.0	1.2	5.8	2.7	5.1	1.9
5.0	3.5	1.6	0.6	5.0	2.3	3.3	1.0	6.8	3.2	5.9	2.3
5.1	3.8	1.9	0.4	5.6	2.7	4.2	1.3	6.7	3.3	5.7	2.5
4.8	3.0	1.4	0.3	5.7	3.0	4.2	1.2	6.7	3.0	5.2	2.3
5.1	3.8	1.6	0.2	5.7	2.9	4.2	1.3	6.3	2.5	5.0	1.9
4.6	3.2	1.4	0.2	6.2	2.9	4.3	1.3	6.5	3.0	5.2	2.0
5.3	3.7	1.5	0.2	5.1	2.5	3.0	1.1	6.2	3.4	5.4	2.3
5.0	3.3	1.4	0.2	5.7	2.8	4.1	1.3	5.9	3.0	5.1	1.8

Table B.1: Anderson's Iris data as published by Fisher. Measurements are in centimeters (cm).

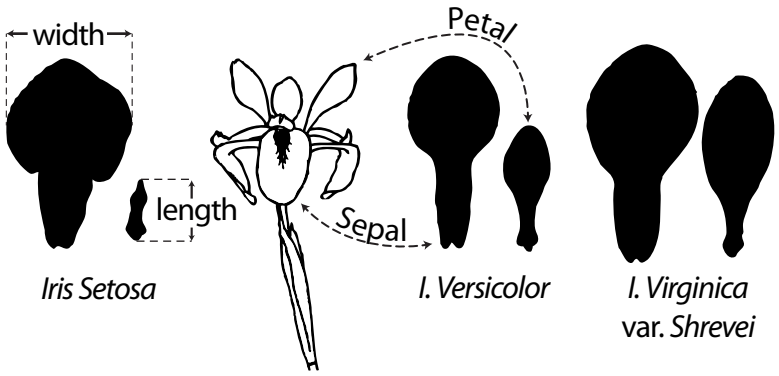

Figure B.33. Iris attributes, redrawn from Anderson's Figures 5 and 8 in *The Species Problem in Iris.*[3] The larger sepals extend downward while the petals extend upward. The solid silhouettes show how the length and width are measured.

List of References

[1] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen*, vol. 7, no. Part II, pp. 179–88, 1936.

[2] E. Anderson, "The irises of the gaspe peninsula," *Bulletin of the American Iris society*, no. 59, pp. 2–5, 1935.

[3] E. Anderson, "The species problem in iris," *Annals of the Missouri Botanical Garden*, vol. 23, no. 3, pp. 457–509, 1936.

[4] J. C. Bezdek, J. M. Keller, R. Krishnapuram, L. I. Kuncheva, and N. R. Pal, "Will the real iris data please stand up?" *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 3, pp. 368–369, 1999.

BIBLIOGRAPHY

"Bibliography of SOM papers," Accessed: 2012-01-22. [Online]. Available: http://www.cis.hut.fi/research/refs/

"Cardiotocography - Wikipedia, the free encyclopedia," Accessed: 2012-02-27. [Online]. Available: http://en.wikipedia.org/wiki/Cardiotocography

"R-Forge: welcome," Accessed: 2011-05-25. [Online]. Available: https://r-forge.r-project.org/

"MATLAB," The Mathworks, Inc.; Natick, Massachusetts, USA, Mar. 2012.

Anderson, E., "The irises of the gaspe peninsula," *Bulletin of the American Iris society*, no. 59, pp. 2–5, 1935.

Anderson, E., "The species problem in iris," *Annals of the Missouri Botanical Garden*, vol. 23, no. 3, pp. 457–509, 1936.

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

Bezdek, J. C., Keller, J. M., Krishnapuram, R., Kuncheva, L. I., and Pal, N. R., "Will the real iris data please stand up?" *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 3, pp. 368–369, 1999.

Bouckaert, R. R., Frank, E., Hall, M., and Kirkby, R., "WEKA manual for version 3-7-5," Accessed: 2012-02-08, Oct. 2011, the University of Waikato. [Online]. Available: http://prdownloads.sourceforge.net/weka/WekaManual-3-7-5.pdf?download

Brownlee, J., "WEKA classification algorithms," Accessed: 2012-02-09, May 2011. [Online]. Available: http://sourceforge.net/projects/wekaclassalgos/

Chambers, J., "How S4 methods work," Accessed: 2010-05-13, Aug. 2006. [Online]. Available: http://developer.r-project.org/howMethodsWork.pdf

Chambers, J., *Software for Data Analysis: Programming with R*, 1st ed. Springer, July 2008.

Ellis, G., Bertini, E., and Dix, A., "The sampling lens: making sense of saturated visualisations," in *CHI '05 extended abstracts on Human factors in computing systems*, CHI EA '05. New York, NY, USA: ACM, 2005, pp. 1351–1354.

Fisher, R. A., "The use of multiple measurements in taxonomic problems," *Ann. Eugen*, vol. 7, no. Part II, pp. 179–88, 1936.

Frank, A. and Asuncion, A., "UCI machine learning repository," Accessed: 2011-02-18, 2010, university of California, Irvine, School of Information and Computer Sciences. [Online]. Available: http://archive.ics.uci.edu/ml

Gastner, M. T. and Newman, M. E. J., "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.

Gusein-Zade, S. M. and Tikunov, V. S., "A new technique for constructing continuous cartograms," *Cartography and Geographic Information Science*, vol. 20, pp. 167–173, July 1993.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H., "The WEKA data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, p. 1018, Nov. 2009.

Hamel, L. and Brown, C., "Improved interpretability of the unified distance matrix with connected components," in *Proceeding of the 7th International Conference on Data Mining*. Las Vegas Nevada, USA: CSREA Press, July 2011, pp. 338–343.

Harrie, L., Sarjakoski, L. T., and Lehto, L., "A variable-scale map for small-display cartography," *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, vol. 34, no. 4, pp. 237–242, 2002.

Himberg, J., "Enhancing the SOM based data visualization by linking different data projections," in *Proceedings of the International Symposium on Intelligent Data Engineering and Learning (IDEAL'98)*, Hong Kong, Oct. 1998, p. 427434.

House, D. H. and Kocmoud, C. J., "Continuous cartogram construction," in *Proceedings of the conference on Visualization '98*. Research Triangle Park, North Carolina, United States: IEEE Computer Society Press, 1998, pp. 197–204.

Hsu, C. C., "Generalizing self-organizing map for categorical data," *IEEE Transactions on Neural Networks*, vol. 17, no. 2, pp. 294–304, 2006.

Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., Takatsuka, M., and Bui, M., "Parallel batch training of the Self-Organizing map using OpenCL," in *Neural Information Processing. Models and Applications*, Wong, K. W., Mendis, B. S. U., and Bouzerdoum, A., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6444, pp. 470–476.

Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., Plzlbauer, G., Rauber, A., and Dittenbach, M., "A vector field visualization technique for self-organizing maps," in *Advances in Knowledge Discovery and Data Mining*, Ho, T. B., Cheung, D., and Liu, H., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, vol. 3518, pp. 399–409.

Kaski, S., Kangasz, J., and Kohonen, T., "Bibliography of Self-Organizing map (SOM) papers: 1981-1997," *Neural Computing Surveys*, vol. 1, pp. 102–350, 1998.

Keahey, T. A., "Visualization of high-dimensional clusters using nonlinear magnification," *Visual Data Exploration and Analysis VI*, vol. 3643 of SPIE, 1999.

Keim, D. A., North, S. C., and Panse, C., "CartoDraw: a fast algorithm for generating contiguous cartograms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 1, pp. 95–110, 2004.

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

Kohonen, T., *Self-Organizing Maps*, 3rd ed., Springer Series in Information Sciences. Berlin, Heidelberg, New York: Springer, 2001, no. 30.

Koua, E. L., "Using self-organizing maps for information visualization and knowledge discovery in complex geospatial datasets," in *Proceedings of 21st International Cartographic Renaissance (ICC)*, 2003, pp. 1694–1702.

Kraaijveld, M. A., Mao, J., and Jain, A. K., "A nonlinear projection method based on Kohonen's topology preserving maps," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 548–559, May 1995.

Lay, D. C., *Linear algebra and its applications*, 3rd ed. Addison Wesley, Sept. 2005.

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

Myklebust, G. and Solheim, J. G., "Parallel self-organizing maps for actual applications," in *International Conference on Neural Networks*, vol. 2. IEEE, 1995, pp. 1054–1059 vol. 2.

Newman, M. E. J., "Cart: Computer software for making cartograms," Accessed: 2011-06-17, Nov. 2006. [Online]. Available: http://www-personal.umich.edu/~mejn/cart/

Newman, M., "Images of the social and economic world," Accessed: 2012-02-26. [Online]. Available: http://www-personal.umich.edu/~mejn/cartograms/

Oja, M., Kaski, S., and Kohonen, T., "Bibliography of self-organizing map (SOM) papers: 1998-2001 addendum," *Neural Computing Surveys*, vol. 3, no. 1, pp. 1–156, 2003.

Pampalk, E., Rauber, A., and Merkl, D., "Using smoothed data histograms for cluster visualization in Self-Organizing maps," in *Artificial Neural Networks ICANN 2002*, Dorronsoro, J. R., Ed., vol. 2415. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 871–876.

Park, Y., Crghino, R., Compin, A., and Lek, S., "Applications of artificial neural networks for patterning and predicting aquatic insect species richness in running waters," *Ecological Modelling*, vol. 160, no. 3, pp. 265–280, Feb. 2003.

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

Pll, M., Honkela, T., and Kohonen, T., "Bibliography of self-organizing map (SOM) papers: 2002-2005 addendum," Helsinki University of Technology, Helsinki, Tech. Rep. TKK-ICS-R23, 2007.

Plzlbauer, G., Rauber, A., and Dittenbach, M., "Advanced visualization techniques for self-organizing maps with graph-based methods," *Advances in Neural NetworksISNN 2005*, pp. 813–813, 2005.

Plzlbauer, G., Rauber, A., and Dittenbach, M., "A vector field visualization technique for self-organizing maps," in *Advances in Knowledge Discovery and Data Mining*, 2005, pp. 399–409.

R Foundation for Statistical Computing,, "The comprehensive R archive network," Accessed: 2011-05-18. [Online]. Available: http://cran.r-project.org/

Raisz, E., "The rectangular statistical cartogram," *Geographical Review*, vol. 24, no. 2, pp. 292–296, Apr. 1934.

Rossi, F., "R-Forge: yasomi: Self-Organising Map in R: Project home," Accessed: 2012-02-27, Feb. 2011. [Online]. Available: https://r-forge.r-project.org/projects/yasomi/

Sammon, John W., J., "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. C-18, no. 5, pp. 401–409, May 1969.

Sarkar, D., *Lattice: Multivariate Data Visualization with R.* New York: Springer, 2008, ISBN 978-0-387-75968-5.

Silva, B. and Marques, N., "A hybrid parallel SOM algorithm for large maps in data-mining," in *Proceedings of the Portuguese Conference on Artificial Intelligence*, 2007.

Steinberg, S., "Front cover," *The New Yorker*, p. OFC, Mar. 1976.

Team, R. D. C., *R: A Language and Environment for Statistical Computing*, Vienna, Austria, 2011, ISBN 3-900051-07-0.

Temple Lang, D., "Rcartogram: Interface to mark newman's cartogram software," Accessed: 2011-06-16, Nov. 2008, R package version 0.2-2. [Online]. Available: http://www.omegahat.org/Rcartogram/

Tobler, W. R., "Pseudo-cartograms," *Cartography and Geographic Information Science*, vol. 13, no. 1, pp. 43–50, 1986.

Tobler, W. R., "Geographic area and map projections," *Geographical Review*, vol. 53, no. 1, pp. 59–78, Jan. 1963.

Tobler, W. R., "A continuous transformation useful for districting," *Annals, New York Academy of Sciences*, no. 219, pp. 215–220, 1973.

Trutschl, M., Grinstein, G., and Cvek, U., "Intelligently resolving point occlusion," in *IEEE Symposium on Information Visualization, 2003. INFOVIS 2003.* IEEE, Oct. 2003, pp. 131–136.

Ultsch, A., "Self-Organizing neural networks for visualisation and classification," in *Information and classification: concepts, methods, and applications.* University of Dortmund: Springer Verlag, 1993, pp. 307–313.

Ultsch, A., "Maps for the visualization of high-dimensional data spaces," in *Proc. Workshop on Self organizing Maps*, 2003, p. 225230.

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

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

Vesanto, J., Himberg, J., Alhoniemi, E., and Parhankangas, J., "Self-organizing map in MATLAB: the SOM toolbox," CiteSeerX, Tech. Rep., 1999.

Vesanto, J., Himberg, J., Siponen, M., and Simula, O., "Enhancing SOM based data visualization," in *Proceedings of the International Conference on Soft Computing and Information/Intelligent Systems (IIZUKA'98)*, Iizuka, Japan, Oct. 1998, p. 6467.

Ward, M. O., "A taxonomy of glyph placement strategies for multidimensional data visualization," *Information Visualization*, vol. 1, no. 3-4, pp. 194–210, Dec. 2002.

Wehrens, R. and Buydens, L., "Self- and super-organising maps in R: the kohonen package," *J. Stat. Softw.*, vol. 21, no. 5, 2007.

Yan, J., "som: Self-Organizing map," Accessed: 2011-06-16, 2010, R package version 0.3-5. [Online]. Available: http://CRAN.R-project.org/package=som

Yang, M. H. and Ahuja, N., "A data partition method for parallel self-organizing map," in *International Joint Conference on Neural Networks*, vol. 3. IEEE, 1999, pp. 1929–1933 vol. 3.

Yin, H., "Data visualisation and manifold mapping using the ViSOM," *Neural Networks: The Official Journal of the International Neural Network Society*, vol. 15, no. 8-9, pp. 1005–1016, Nov. 2002, PMID: 12416690.

Zumbrunn, T., "R-Forge: diffusion-based cartograms," Accessed: 2010-12-01, Sept. 2010. [Online]. Available: https://r-forge.r-project.org/projects/cart/