7-18-2011

Improved Methods for Cluster Identification and Visualization

Narine Manukyan
University of Vermont

IMPROVED METHODS FOR CLUSTER IDENTIFICATION AND VISUALIZATION IN HIGH-DIMENSIONAL DATA USING SELF-ORGANIZING MAPS.

A Thesis Presented

by

Narine Manukyan

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Master of Science
Specializing in Computer Science

May, 2011

Accepted by the Faculty of the Graduate College, The University of Vermont, in partial fulfillment of the requirements for the degree of Master of Science, specializing in Computer Science.

Thesis Examination Committee:

_______________________________________ **Advisor**
Margaret J. Eppstein, Ph.D.

____
Donna M. Rizzo, Ph.D.

_______________________________________ **Chairperson**
Jeffrey Marshall, Ph.D.

_______________________________________ **Dean, Graduate College**
Domenico Grasso, Ph.D.

Date: March 25, 2011

ABSTRACT

Self-organizing maps (SOMs) are self-organized projections of high dimensional data onto a low, typically two dimensional (2D), map wherein vector similarity is implicitly translated into topological closeness in the 2D projection. They are thus used for clustering and visualization of high dimensional data. However it is often challenging to interpret the results due to drawbacks of currently used methods for identifying and visualizing cluster boundaries in the resulting feature maps. In this thesis we introduce a new phase to the SOM that we refer to as the Cluster Reinforcement (CR) phase. The CR phase amplifies within-cluster similarity with the consequence that cluster boundaries become much more evident. We also define a new Boundary (B) matrix that makes cluster boundaries easy to visualize, can be thresholded at various levels to make cluster hierarchies apparent, and can be overlain directly onto maps of component planes (something that was not possible with previous methods).

The combination of the SOM, CR phase and *B-matrix* comprise an automated method for improved identification and informative visualization of clusters in high dimensional data. We demonstrate these methods on three data sets: the classic 13-dimensional binary-valued “animal” benchmark test, actual 60-dimensional binary-valued phonetic word clustering problem, and 3-dimensional real-valued geographic data clustering related to fuel efficiency of vehicle choice.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my adviser Dr. Margaret J. Eppstein for the continuous support of my Master study and research, for her patience, motivation, enthusiasm, and immense knowledge. Throughout my thesis-writing period, she provided encouragement, sound advice, good teaching, good company, and lots of good ideas. I would like to thank Dr. Donna Rizzo for being an excellent and passionate teacher from whom I have learnt a lot. I would also like to thank Dr. Jeffrey Marshall for being the chair of my defense committee.

Last but not least I would like to thank my parents Kliment Manukyan and Gohar Manukyan, and my husband Brian Hall for their continued support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1. Motivation

Finding patterns in vast multidimensional data sets can be difficult and time-consuming; nonlinear relations between data items may be non-obvious. Algorithms capable of clustering high-dimensional data are one way to address this problem. For example, high dimensional clustering allows content based image retrieval [26]. Using a high-dimensional data clustering algorithm, the images that describe similar objects can be clustered together and then retrieved. This problem has been the object of ongoing research for decades [40]. As digital image and video libraries continue to grow rapidly, being able to scan data by content will continue to grow in importance.

Another example is analysis of gene expression data. DNA microarray technologies together with rapidly increasing genomic sequence information is leading to an explosion in available gene expression data. Currently there is a great need for efficient methods to analyze and visualize these massive data sets. Clustering algorithms can be used for analysis and visualization of gene expression profiles in order to understand which genes are co regulated; such information can help in inferring the structure and function of genetic regulatory networks [46]. A third example is using clustering algorithms for geographical data, for example stream sensitivity (or instability) classification. Watershed planners have long sought decision-making tools for forecasting changes in stream-channels over large spatial and temporal scales. Using clustering algorithms it is possible to assimilate large amounts of disparate data types for use in fluvial hazard management decision-making [3].

All of the above-mentioned high-dimensional clustering problems have been approached using an algorithm called the self-organizing map (SOM), a type of artificial neural network. The biologically inspired SOM algorithm proposed by Kohonen [19, 22], can be used to visualize high-dimensional data and find similarities (clusters) in the data set without a priori knowledge about the number of clusters. The SOM algorithm can identify both linear and non-linear relationships in the data. The self-organizing feature map (SOM) algorithm is a biologically inspired method for constructing a structured representation of data from an often high-dimension input space.

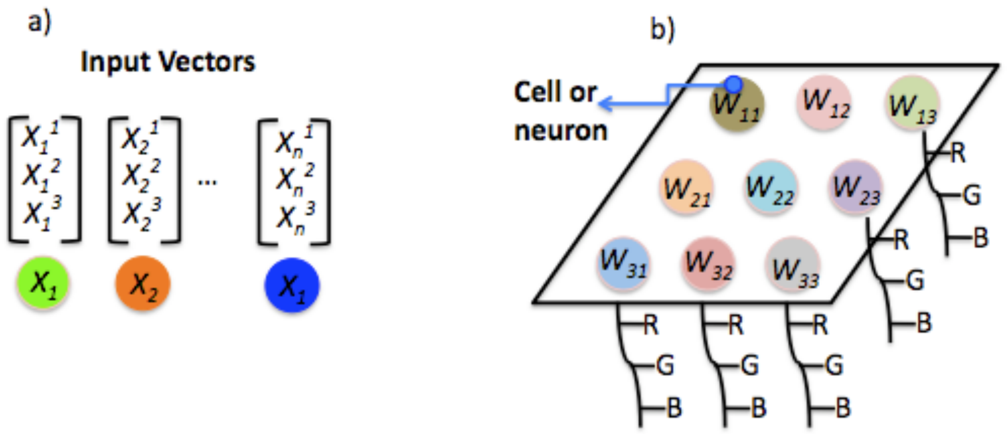


Figure 1.1 a) The 3-dimensional input vectors of RGB values. b) The randomly initialized feature map of weight vectors.

A simple conceptual example that demonstrates the SOM algorithm is the clustering of data representing colors. Assume we have a list of colors, each described with RGB (Red, Green, Blue) values, and we want to cluster them according to these three attributes. The goal is to create a low dimensional 2D projection from the higher dimensional 3D input space in a way that similar colors are topologically close to each other in the 2D projection. The input space X for this problem is the list of 3 dimensional vectors representing RGB values for every color (Fig. 1.1a). The SOM algorithm first creates a grid of cells called neurons, and initially associates randomly chosen three dimensional weight vectors with each cell on the grid (Fig. 1.1 b). In Fig. 1.1 b) the colors of the cells reflect the random numbers that were used for initializing the weight vectors. The weight vectors, as a function of neuron coordinates, are called the *feature map*. Feature maps generated by the SOM algorithm are characterized by the fact that similar input vectors are effectively mapped onto neighboring regions of neurons [10].

In Fig. 1.2 we show an example list of input colors (Fig.1.2. a), the graphical representation of feature maps of colors before (Fig. 1.2b) and after (Fig. 1.2c) training with the SOM algorithm. As you can see, after training with the SOM algorithm the colors in the feature map have self-organized into a map of the input colors. For example the shades of blue are next to each other, as are the shades of green, etc. In this simplified example to show some basic ideas about the SOM algorithm, the number of neurons exactly equals the number of input vectors, but in general this is not necessarily true.

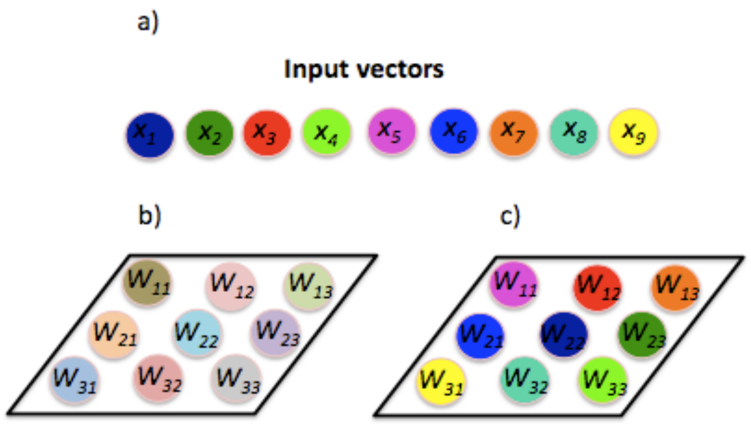

a)
Input vectors
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9
b)
W_{11} W_{12} W_{13} W_{21} W_{22} W_{23} W_{31} W_{32} W_{33}
c)
W_{11} W_{12} W_{13} W_{21} W_{22} W_{23} W_{31} W_{32} W_{33}

Figure 1.2 a) Example list of input RGB colors. b) Initial random RGB colors (weight vectors) on the feature map. c) The feature map after training with the SOM.

There are three essential processes that lead to the formation of the feature map: competition, cooperation and adaption. In the competition process, a distance metric is computed between an input vector and weight vectors of all neurons in the feature map. The neuron that is closest to the input vector is declared to be the winning neuron (winner of the competition) (e.g., W_{33} in Fig. 1.3). A spatial neighborhood is computed around the winning neuron, providing the basis for cooperation among the neighboring neurons (Fig. 1.3 red rectangle). In the adaptation process the "excited" neurons (i.e., the winning neuron and its neighbors) update their individual values to become closer to the input pattern.

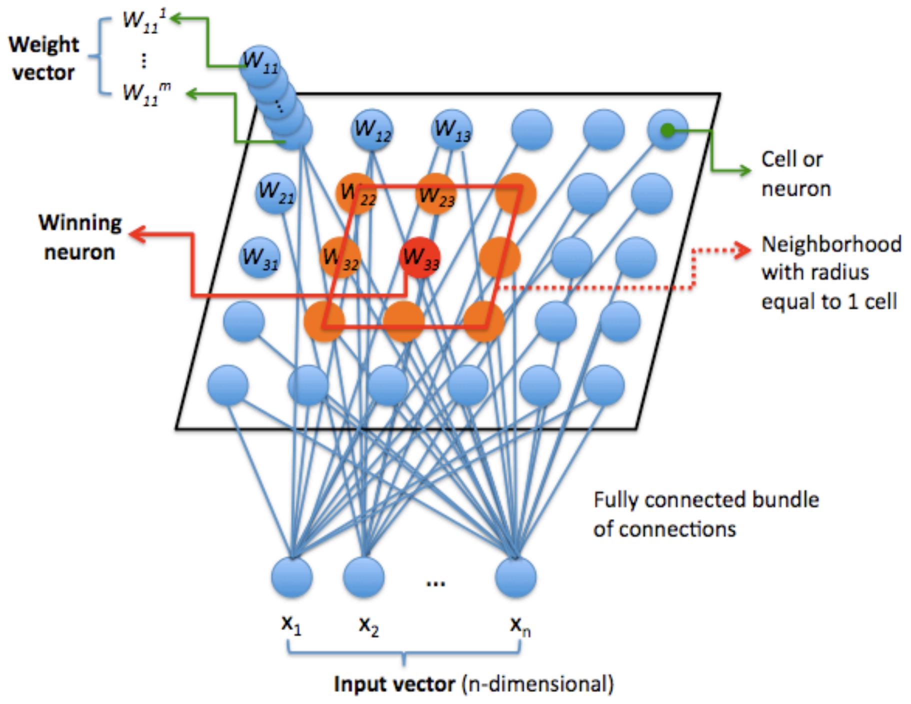


Figure 1.3 Kohonen model of feature mapping.

For years, the low dimensional, ordered representation of data generated by the SOM algorithm has proven useful for variety of technical applications. A few of myriad examples include stream classification [3], geographic pattern detection [1], detection of age-related changes in functional connectivity during rest in autism spectrum disorders [51], assessing the effort of meteorological variables for evaporation estimation [5], detecting skeletal anomalies in aquaculture [41], monitoring chemical processes [47], DNA microarray analysis [31], pattern classification and function approximation [18, 20, 42], knowledge representation [38, 44] and linguistic data classification [16, 43]. The algorithm has also been successfully applied as a model for the development of structural representations in biological neural systems, so-called brain maps [32]. Below we describe some of these applications in more detail.

In geographic pattern detection [1] the authors are addressing regionalization and land-use patterns that are one of the longstanding concerns of geographers. They use a type of SOM to identify relatively homogeneous regions and perform geographic pattern recognition. The main advantage of this method is the possibility of "what if" analyses together with powerful visualization tools and the accommodation of spatial constraints.

Wiggins et al. [51] use a data-driven approach with SOMs to calculate the connectivity rate of posterior-anterior connectivity in healthy individuals and patients with autism spectrum disorders (ASD) during rest. They used individualized resting-state clusters identified by an SOM algorithm to corroborate previous findings of weaker posterior-anterior connectivity in the ASD group and examined age-related changes.

Their findings show that the SOM is a good method for calculating connectivity in a clinical population.

Chang et al. [5] used an SOM to assess the variability of daily evaporation based on meteorological variables. This is an important question as the evaporation affects the distribution of water in the hydrological cycle and plays a key role in water resource management and agriculture. They use daily meteorological data sets from a climate gauge and investigate their multi-collinear relationships. They compare the results with the results of another artificial neural network (back propagation neural network) and show that the SOM performs better.

Ultsch et al. [47] are using an SOM for monitoring chemical processes. In complex nonlinear processes like chemical processes, it is not possible to predict all possible error types in advance. The authors use an SOM algorithm to find dependencies between various process parameters, as well as input and output variables. The SOM also allows them to visualize the process development that helps to estimate the future behavior, and perform efficient fault diagnosis.

Kiviluoto et al. [18] are using an SOM to analyze financial statements and predict bankruptcy. Through qualitative analyses with SOM, companies are classified into healthy and bankrupt-prone ones. They also compare results of the SOM to linear discriminant analysis and learning vector quantization and show that in some instances the SOM outperforms these methods, while in others it does not.

Ritter and Kohonen [39] are using an SOM for detecting “logical similarity” between words from the statistics of their contexts. Inclusion of context in which symbols appear is defined in two ways. In a first demonstration they define the context as a set of attribute values that occur in conjunction with the words. In a second demonstration they define context by the sequence in which the words occur. They analyze simple nouns, verbs, and adverbs in this way. In both of the simulations the words grouped in same categories. The authors also argue that similar processes may be working in the brain.

As the above applications show, the SOM algorithm can be used for variety of problems. Most researchers create a so-called "*U-matrix*", which is essentially a map of spatially windowed averages of vector differences in the feature map. However as we will show, the *U-matrix* is often insufficient for properly identifying clusters.

In this thesis, we introduce a new phase to the SOM that advances the cluster separation by strengthening cluster boundaries; we refer to this new process as the Cluster Reinforcement (CR) phase. The CR phase helps the weights of the same clusters become more similar; consequently cluster boundaries become much more evident. We also define a new Boundary (B) matrix method for visualization of clusters in feature maps. The *B-matrix* method not only makes cluster boundaries sharp and clear, but can also be overlain on maps of component planes. Although the contour plots of the *U-matrix* can also be simultaneously displayed as heat maps of component planes [6, 37], in Chapter 4 we show that the new visualization method is more effective in visualizing clusters. The combination of the SOM, CR phase and *B-matrix* results in highly informative and user-friendly identification and visualization of clusters in high dimensional data.

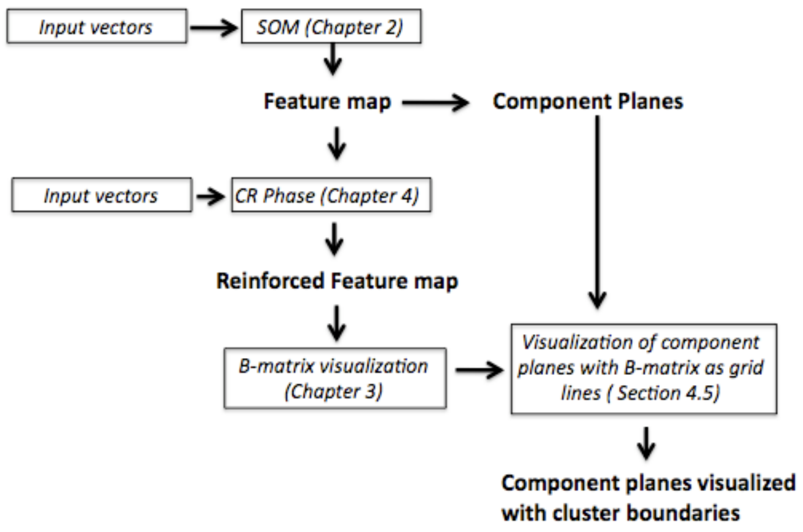

Input vectors
SOM (Chapter 2)
Feature map
Component Planes
Input vectors
CR Phase (Chapter 4)
Reinforced Feature map
B-matrix visualization (Chapter 3)
Visualization of component planes with B-matrix as grid lines (Section 4.5)
Component planes visualized with cluster boundaries

Figure 1.4 Proposed SOM+CR+*B-matrix* algorithm.

In Fig. 1.4 we show an overview of the combined algorithm, along with an indication of where each process is discussed in this thesis. The outline of the thesis is as follows:

In Chapter 2 we review the details and some theoritical aspects of the SOM algorithm. In Chapter 3 we introduce a new visualization method called the *B-matrix*. This motivates the need for an additional phase to the SOM algorithm called the Cluster Reinforcement (CR) phase described in Chapter 4, where we discuss the differences between SOM and SOM+CR for a classic benchmark clustering problem. In Chapter 5 and Chapter 6 we show two real world applications, one with a binary-valued 60-dimensional dataset and one with real-valued 3-dimensional dataset.

CHAPTER 2: SELF-ORGANIZING MAPS

2.1. SOM Algorithm

Thirty years ago Kohonen [19, 20, 22] proposed a stochastic algorithm, that he called a self-organizing process. The basic property of this algorithm is to create low dimensional (typically 2D) topologically ordered projections from high dimensional data. Although this algorithm is very easy to define, its mathematical analysis is difficult and needs sophisticated probabilistic concepts [28]. In this section we will review the algorithm and the main concepts.

In a self-organizing map, so-called "neurons" (in reality these are weight vectors) are placed at the nodes of a one- or two- dimensional lattice. Higher-dimensional maps are also possible but are not as common. The neurons become selectively tuned to various input patterns (stimuli) in the course of a competitive learning process. The weight vectors of spatially adjacent neurons in the lattice become tuned with respect to each other in such a way that a meaningful spatial organization for different input features emerges on the lattice [23, 45] . Therefore a topographic map of input patterns is formed after SOM training in which the special locations of the neurons are indicators of different statistical features in the input patterns. As Kohonen stated:

> "The spatial location of an output neuron in a topographic map corresponds to a particular domain or feature of data drawn from the input space." (Kohonen 1990, page 2) [23].

In Fig. 1.3, we showed the layout of a SOM network in which the output neurons are arranged in a two-dimensional lattice. This kind of topology ensures that each neuron has a set of neighbors. Each neuron in the lattice is fully connected to all the source nodes in the input layer.

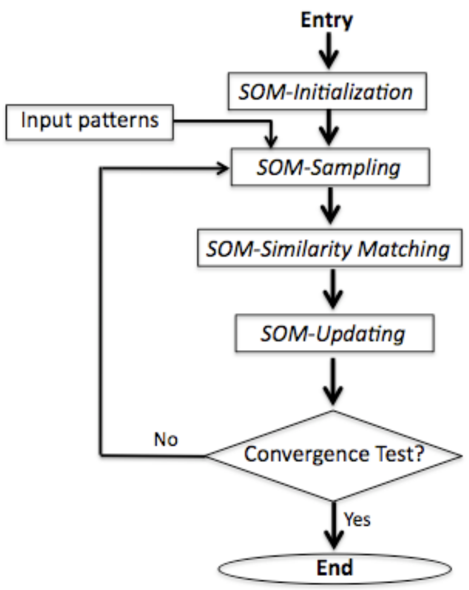


Figure 2.1 This figure shows the algorithm for the SOM.

The basic SOM algorithm is shown in Fig. 2.1 and described below.

1. SOM Initialization. Choose random values for initial *W* weight vectors associated with each neuron.
2. SOM Sampling. Choose a random input vector *x* from the input space *X*. The vector *x* represents the activation pattern that is applied to the lattice. The dimension of vector *x* is equal to $m \times 1$, where *m* is the number of features (elements) in each input vector.

3. SOM Similarity Matching. Find the best-matching (winning) neuron $i(x)$ by calculating the minimum-distance Euclidean criterion between the selected input vector x and the weights of all l neurons in the lattice:

$$i(x) = \arg\min_j \left\| x(n) - w_j \right\|, \; j = 1,2,...,l \qquad (2.1)$$

4. SOM Updating. Adjust the synaptic weight vectors of all neurons by using the update equations (2.2)-(2.5).

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_1}\right) \qquad (2.2)$$

$$h_{j,i(x)}(t) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2(t)}\right) \qquad (2.3)$$

$$\eta(t) = \eta_0 \exp\left(-\frac{t}{\tau_2}\right) \qquad (2.4)$$

$$w_j(t+1) = w_j(t) + \eta(t) h_{j,i(x)}(t)(x(t) - w_j(t)) \qquad (2.5)$$

where $t = 0,1,2,3...$, is the time step, $\sigma(t)$ given in equation (2.2) is the standard deviation of the Gaussian topological neighborhood function $h_{j,i(x)}$ given in equation (2.3), σ_0 is the initial value of σ and τ_1 and τ_2 are time constants. The neighborhood function shrinks with time, due to the fact that $\sigma(t)$ decreases. The Gaussian topological neighborhood function $h_{j,i(x)}$ is a typical choice of neighborhood function that satisfies the following two requirements:

a) It is symmetric about the maximum point defined by $d_{j,i} = 0$, where $d_{j,i}$ is the topological distance between neurons *j* and *i* on the grid, given by $d_{j,i} = \|r_j - r_i\|$, where r_j defines the position of excited neuron *j* and r_i defines the position of winning neuron *i*.

b) The size of the topological neighborhood decreases monotonically with increasing lateral distance $d_{j,i}$, decaying to zero for $d_{j,i} \to \infty$ (this is necessary condition for convergence).

In equation (2.4), $\eta(t)$ denotes the learning-rate parameter that is varied over time. It starts with initial value η_0 and decreases exponentially with increasing time. Finally equation (2.5) describes the weight update function.

5. SOM Convergence Test. Iterate steps 2 through 5 until convergence (e.g., when changes in the feature map fall below some specified tolerance).

As the algorithm shows there are three essential processes involved in the formation of the self-organizing map: competition (find the winning neuron), cooperation (identify a neighborhood of neurons) and adaptation (update of all neurons in the neighborhood).

2.2. Theoretical Aspects of the SOM Algorithm

As discussed in Chapter 1, many real world applications of the SOM have been found to be useful. However, a general theory of the algorithm has not yet been achieved. It is not clear under what conditions the algorithm may be guaranteed to converge or whether the algorithm works by performing a stochastic gradient descent on some potential function [52], and problems of important practical interest, like the number and type of the algorithm's stationary states, convergence speed as a function of the algorithms' parameters and the avoidance of sub-optimal representations, are not solved [11]. Some of these questions have been addressed to some extent for simple cases [10]. As far as we know, the only case where a complete analysis has been achieved is for scalar inputs with a linear (1D) network [9].

A sketch of a proof for self-organization and convergence was provided in Kohonen's original papers [19, 22] and in his books [21, 24] in 1984 and 1991. The first complete proofs of both self-organization and convergence were established (for uniform distribution of the inputs and a restricted neighborhood function) by Cottrell and Fort in 1987 [8]. Sufficient conditions for self-organization, are given in [13], for the case of decreasing neighborhood whose width is such that at the first iteration more then half of the neurons are updated. These results were later generalized to a wider class of input distributions by Bouton and Pagès [4] and to a more general neighborhood by Erwin *et al.* [10], who have sketched the extension of the proof of self-organization and studied the role of the neighborhood function. Finally, Fort and Pagès in 1994 [15], 1995 [14], 1997 [2] (with Benaim) have achieved the proof of the almost sure convergence towards a unique state, after self-organization, for a very general class of neighborhood functions. Flanagan in 2001 [12] gave some results in higher dimensions, but these remain incomplete.

2.3. SOM Visualization

Interneuron distance has been traditionally used for cluster detection in the SOM literature. A unified distance matrix, called the *U-matrix* was proposed by Ultsch and Siemon [50] to illustrate the clustering of input vectors in the SOM. Other variations of the *U-matrix* were later introduced in [25] and [30].

In their 1990 paper, Ultsch and Siemon introduced the idea of using $2n\text{-}1 \times 2n\text{-}1$ matrix to store distances between the weight vectors in the SOM output of size $n \times n$ [50]. It was an interesting idea, but the presentation was incomplete, the explanation of the method had many typographical mistakes, and is thus unclear, and we have not found any use of this method in the literature.

In a later paper [48] Ultsch (1993) describes the *U-matrix* using a *U-height* function (2.6). He defines the *U-height* of a neuron *i* as the sum of all data distances from the weight vector of neuron *i* to the weight vectors of the neurons in $NN(i)$, where $NN(i)$ represents the nearest neighbors of the neuron *i* (e.g., a Moore neighborhood in a rectangular 2D lattice). More specifically, let *i* be a neuron on the map, $NN(i)$ be the set of nearest neighbors of *i* on the map, $w(i)$ the weight vector associated with neuron n_i, then

$$U\_height(n) = \sum_{j \in NN(i)} d(w(i), w(j)) \quad (2.6)$$

where *d(x,y)* is the distance function used in the SOM algorithm to construct the map. The *U-matrix* is an $n \times n$ matrix of the *U_heights* of each of the grid positions of the neurons on the map [49]. The *U-matrix* is usually displayed as a heat map, three-dimensional surface [48, 49], or contour map [6].

This implementation of the *U-matrix* has become the standard tool for the display of the distance structures of the input data on SOM [23]. It has been used in a number of applications including geographic pattern detection [1], stream classification [3], clustering microbial community profile data [36], monitoring chemical processes [47], and DNA microarray analysis [31]. The *U-matrix* helps to visualize the emergence of structural features of the distances within the data space. Outliers, as well as possible cluster structures, can often be recognized for high dimensional data spaces. Unfortunately since the *U-matrix* is $n \times n$ that means that the sum of the neighborhood distances for each neuron is placed in the position of that neuron, which makes it difficult to see the separation of neurons close to cluster boundaries. In Chapter 3 of this thesis we illustrate this problem and introduce a new method called the Boundary (B) matrix that helps to overcome this limitation of the *U-matrix*.

There have been other similar methods reported in the literature for visualizing SOM output. In [25] the visualization of the SOM is implemented by constructing an image where the grey scale of each pixel represents the maximum distance in the feature space of the corresponding unit to its four nearest neighbors in the 2D rectangular grid (Von Neumann neighborhood). The larger the distance, the lighter the grey value is [25].

Another similar method is median interneuron distance (MID) matrix [30]. Each MID entry is the median of the Euclidean distances between the corresponding pointer and all pointers belonging to a star-shaped, fixed radius neighborhood containing typically eight units (Moore neighborhood). The median can be seen as a conservative choice; more radical options based on extremes can also be implemented like max, min, etc.

All of these methods are using the same idea of measuring distances between neighboring neurons and using sum, max or median functions to plot them in a grey scale. All of these methods have the limitation of putting the resulting information in an $n \times n$ grid; as mentioned above this can result in an unclear separation clusters.

CHAPTER 3: BOUNDARY MATRIX

In this chapter, we introduce a new method for identifying cluster boundaries in Self-Organizing Maps (SOM), which we refer to as the Boundary (B) matrix. We then show how the *B-matrix* makes clusters easier to identify than the traditionally utilized *U-matrix* [48] discussed in Chapter 2.

3.1. Construction of the *B-matrix*

Let's assume we have an $n \times n$ matrix W of spatially organized weight vectors each of size $1 \times m$ and we wish to identify clusters of similar adjacent weight vectors. The matrix W is a 2D representation of the multidimensional input space, such as would be created by an SOM, where weight vectors of neurons are spatially organized in the grid by similarity. In order to improve visual identification of potential clusters in W, we propose a toroidal *B-matrix* of size $2n \times 2n$. Assuming that matrix indexing runs from 0 to $2n$-1, the $n \times n$ elements of the *B-matrix* that have both indices even correspond to the original $n \times n$ elements in W. The remaining entities of B correspond to the boundaries between adjacent elements in W. Thus, the positions with two even indices are assigned placeholder values of *NaN* (Not a Number) and all other positions are filled with distances between adjacent elements in W, as illustrated in Fig. 2. In these schematics, we use the notation $\tilde{W}_{i,j}$ to identify the position in the *B-matrix* corresponding to W. In Fig. 3.1a, the horizontal arrows denote the distances between horizontal elements, the vertical arrows denote the distances between vertical elements and the crossing arrows denote the mean of the distances between the elements on the diagonals. We illustrate a simplified example of this calculation in Fig. 3.1b, using

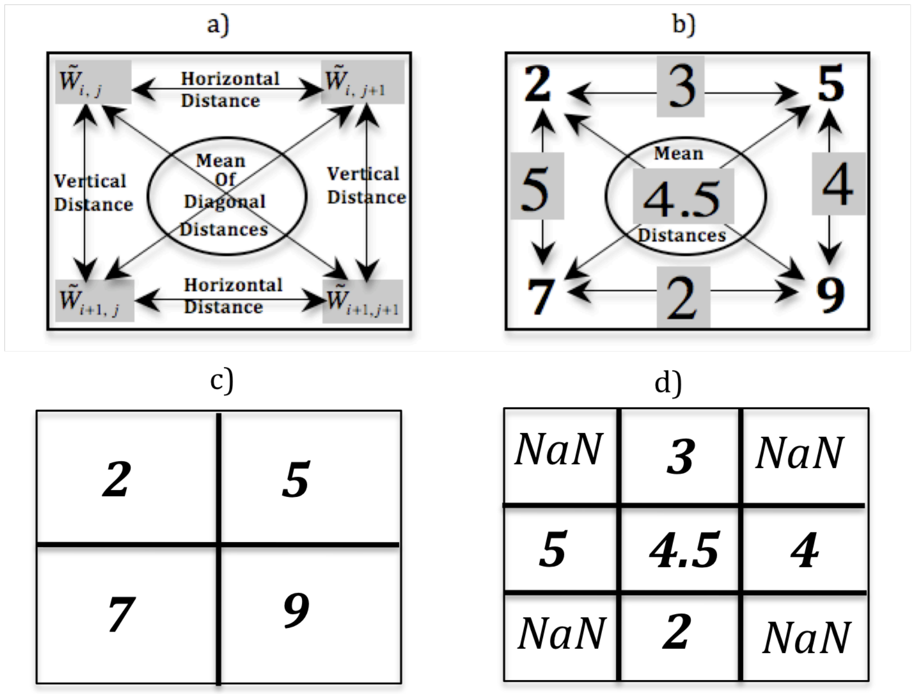


Figure 3.1 a) Illustration of distances between 4 adjacent elements of *W* (shaded) used to compute the elements of the *B-matrix* (not shaded); b) An example calculation of the 5 elements of the *B-matrix* (shaded) computed from representative adjacent values of *W* (not shaded). c) shows the *W* matrix for this example. d) shows the *B-matrix* for this example

sample integers for scalar weight vectors of length $m = 1$. For simplicity we show only 2×2 elements of W (Fig. 3.1 a,c) and the resulting 3×3 elements of the *B-matrix* (Fig. 3.1 b,d).

As the schematics in Fig. 3.1 show, the *B-matrix* is extended relative to W in such a way that the Moore neighborhood (*i.e.,* the 8 nearest neighbors) around each element $B_{i,j}$ (for i and j even) are distance values between the corresponding element and its Moore neighbors in W. The generalized formula for construction of B is thus:

$$B_{i,j} = \begin{cases} NaN \text{ for even}(i), \text{even}(j) \\ \text{Dist}\left(W_{\text{floor}(i/2),\text{floor}(j/2)}, W_{\text{mod}(\text{floor}(i/2)+1,n),\text{floor}(j/2)}\right) \text{ for odd}(i), \text{even}(j) \\ \text{Dist}\left(W_{\text{floor}(i/2),\text{floor}(j/2)}, W_{\text{floor}(i/2),\text{mod}(\text{floor}(j/2)+1,n)}\right) \text{ for even}(i), \text{odd}(j) \\ \text{mean}\begin{pmatrix} \text{Dist}\left(W_{\text{floor}(i/2),\text{floor}(j/2)}, W_{\text{mod}(\text{floor}(i/2)+1,n),\text{mod}(\text{floor}(j/2)+1,n)}\right), \\ \text{Dist}\left(W_{\text{mod}(\text{floor}(i/2)+1,n),\text{floor}(j/2)}, W_{\text{floor}(i/2),\text{mod}(\text{floor}(j/2)+1,n)}\right) \end{pmatrix} \text{ for odd}(i), \text{odd}(j) \end{cases} \quad (3.1)$$

Where indices i,j satisfy $0 \le i \le 2n\text{-}1$ and, $0 \le j \le 2n\text{-}1$, floor(x) rounds down to the nearest integer, mod(x,y) is x modulo y, and Dist(x,y) is any distance function between vectors x and y; in all subsequent applications we use Euclidian distance for Dist(x,y). For example, consider the following weight matrix W of scalar weight vectors (*i.e.*, vectors of size $m \times 1$ where m = 1), specifically constructed to have four clusters of similar weights {1 2 3}, {10 11 12}, {16 18}, {26 27 28}, shown in different colors:

$$W = \begin{bmatrix} 1 & 3 & 26 & 26 & 26 \\ 1 & 3 & 27 & 28 & 28 \\ 2 & 2 & 18 & 18 & 18 \\ 10 & 10 & 16 & 18 & 18 \\ 11 & 12 & 16 & 16 & 18 \end{bmatrix} \quad (3.2)$$

The corresponding *B-matrix* is shown below, where the *NaN* values at positions corresponding to the elements of W from equation (3.2) are represented as dashes.

$$B = \begin{bmatrix} - & 2 & - & 23 & - & 0 & - & 0 & - & 25 \\ 0 & 2 & 0 & 23.5 & 1 & 1.5 & 2 & 2 & 2 & 26 \\ - & 2 & - & 24 & - & 1 & - & 0 & - & 27 \\ 1 & 1 & 1 & 20 & 9 & 9.5 & 10 & 10 & 10 & 21.5 \\ - & 0 & - & 16 & - & 0 & - & 0 & - & 16 \\ 8 & 8 & 8 & 11 & 2 & 1 & 0 & 0 & 0 & 12 \\ - & 0 & - & 6 & - & 2 & - & 0 & - & 8 \\ 1 & 1.5 & 2 & 5 & 0 & 1 & 2 & 1 & 0 & 7.5 \\ - & 1 & - & 4 & - & 0 & - & 2 & - & 7 \\ 10 & 9.5 & 9 & 13.5 & 10 & 10 & 10 & 9 & 8 & 16 \end{bmatrix} \qquad (3.3)$$

The *B-matrix* of equation (3.3) is visualized in a grey scale in Fig. 3.2, with *NaN* values shown in white. Darker regions indicate that the distance between the weights in *W* is large and lighter regions indicate that corresponding vectors of *W* are closer. The elements with two even indices, corresponding to the elements of *W*, are overlain with the corresponding values of *W* (shown in blue marked with asterisks), while the remaining elements are overlain with the numeric values of the *B-matrix* (colored in red).

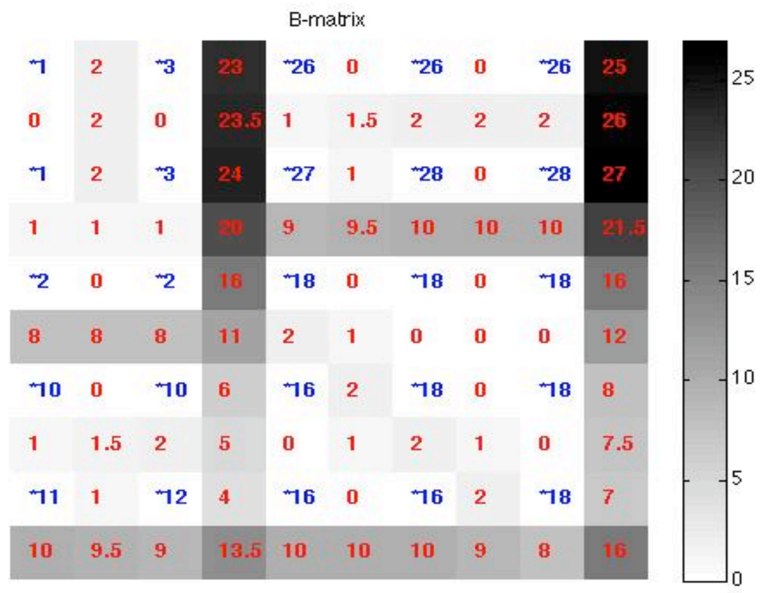


Figure 3.2 Visualization of *B*-matrix from equation (3.3); non-zero values of *B* are shown in red, and corresponding values of *W* are indicated with an asterisk in blue.

In contrast, Fig. 3.3 shows the traditional *U-matrix* for the same weights *W* from equation (3.2). For each element *i,j* in *U*, we calculate the average Euclidian distance from W_{ij} to it's 8 nearest neighbors (Moore neighborhood) as discussed in Chapter 2. Note that, unlike the $2n \times 2n$ *B-matrix*, the *U-matrix* is $n \times n$.

For the readers' convenience, we place these example *B*- and *U*- matrices side by side in Fig. 3.4. The four clusters of *W*: {1 2 3}, {10 11 12}, {16 18}, {26 27 28} are readily identifiable in the *B-matrix* as separated by dark boundaries (Fig. 3.4 right, circled), whereas in the *U-matrix* (Fig. 3.5 left) these clusters are not at all apparent.

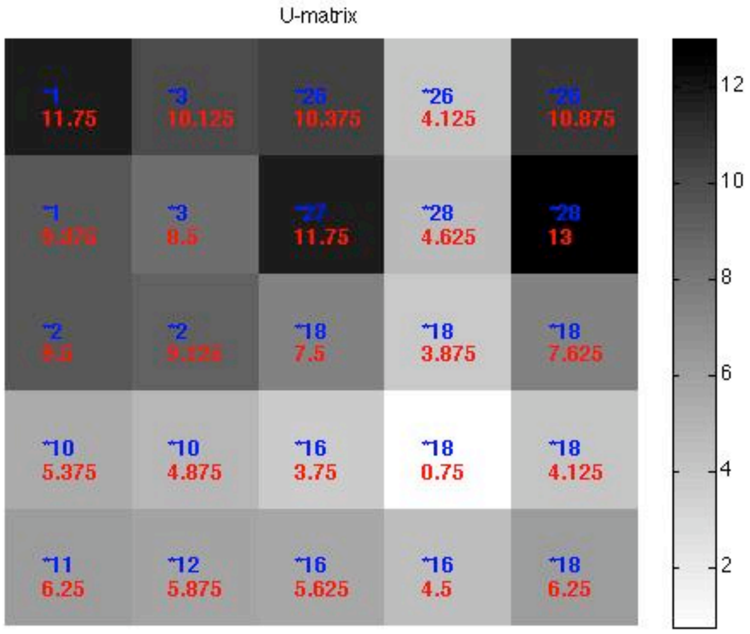


Figure 3.3 Visualization of the *U-matrix*; values of *U* are shown in red, and corresponding values of *W* from equation (3.2) are indicated with an asterisk in blue.

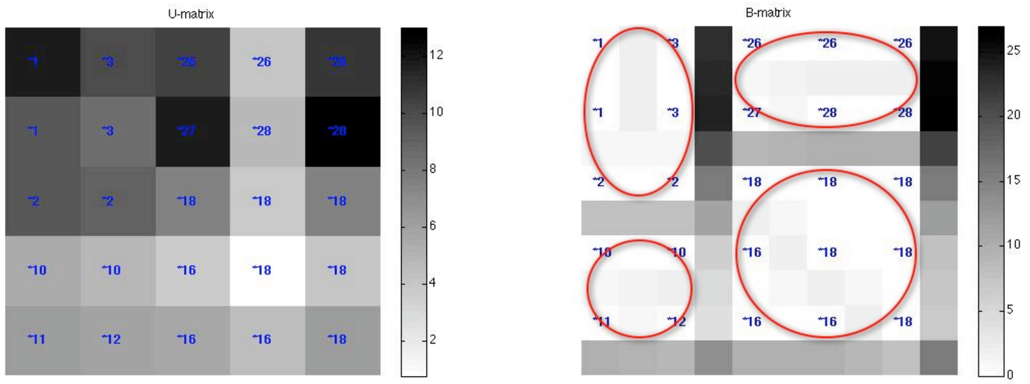


Figure 3.4 *U-matrix* (left) and *B-matrix* (right) corresponding to the *W* shown in equation 8 and shown here in blue and asterisk.

3.2. Discussion

The *U-matrix* is the canonical tool that has been used for the display of the distances between weights on self-organizing maps [47]. However, as illustrated here, it may fail to detect clusters in some data sets [49]. In this section, we have proposed the *B-matrix* as an alternative to the *U-matrix* and used a simple example to illustrate how the *B-matrix* separates the boundaries between the clusters from the cluster elements themselves, thus making clusters evident, even in cases where the *U-matrix* fails. In Chapter 4, we will show an alternative way of visualizing the *B-matrix* that facilitates overlaying cluster boundaries onto $n \times n$ matrices, such as the component planes of *W*. In Chapters 5 and 6 we show how the *B-matrix* can be used to identify clusters in two real world problems, one with binary weight vectors and one with real-valued weight vectors.

CHAPTER 4: CLUSTER REINFORCEMENT PHASE

In this chapter, we introduce an additional phase in Self-Organizing maps (SOM) for strengthening cluster boundaries, which we refer to as the Cluster Reinforcement (CR) phase. We use a classical problem (animal clustering) [39] to show how the CR phase makes the cluster boundaries sharper and more easily detectable than with the SOM algorithm alone. We show how the *B-matrix* (see Chapter 3) facilitates the visualization of these cluster boundaries, especially when overlain onto the SOM component planes where the B-values are visualized by the thickness of grid lines. In the following chapters (Chapters 5 and 6) we show applications of the CR phase.

4.1. Self-Organizing Maps With Cluster Reinforcement Phase

Starting from an initial state of complete disorder, the SOM algorithm gradually leads to an organized representation of activation patterns (weight vectors) drawn from the input space, assuming the parameters of the algorithm are selected properly (see Chapter 2). Initially, topological ordering of weight vectors takes place. Ones this has occurred, weight vectors continue to be fine-tuned until convergence occurs. However, identification of self-organized clusters in the SOM algorithm can be difficult, as the cluster boundaries may be unclear and it is difficult to see how the SOM component planes relate to clusters. We introduce an additional phase Cluster Reinforcement (CR) phase to the SOM that helps the elements of the same clusters become more similar, thus resulting in a stronger separation between clusters. This cluster sharpening proceeds in an unsupervised manner; without prior knowledge of the domain or number of clusters to be identified. By overlaying the *B-matrix* of the SOM + CR onto the component planes of the SOM, one is able to clearly visualize clusters. Thresholding the B-values at different levels permits hierarchical cluster visualization. Preliminary experimentations showed that, after successful convergence of the SOM (in 1000-2000 iterations), the CR phase achieved good results in fewer than 100 iterations. Consequently, in all results reported in this thesis we simply use 100 iterations as our convergence criterion for the CR phase.

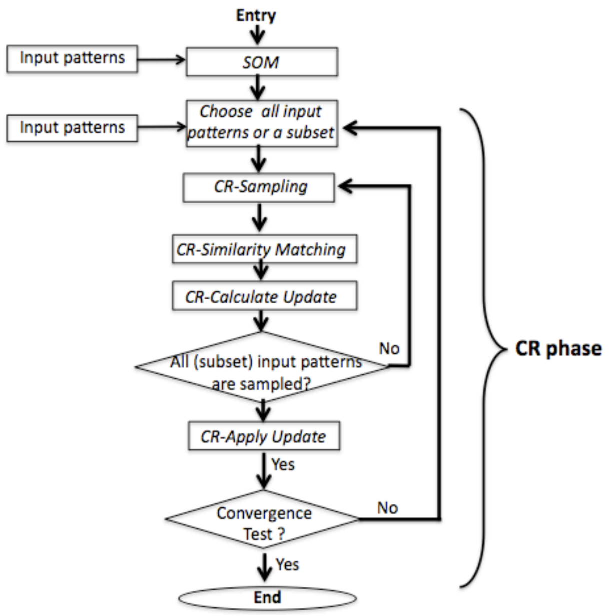


Figure 4.1 This diagram shows the algorithm for CR phase.

In Chapter 2, we gave the description of the algorithm for the SOM. Although the CR phase algorithm is structurally similar to the basic SOM algorithm, there are important differences. In Fig. 4.1 we show the CR phase algorithm, applied in tandem with the traditional SOM.

1. CR Initialization. The CR phase is run subsequent to the SOM, so the converged feature map from the SOM is used as the initial feature map in the CR phase.
2. CR Sampling. Pick all vectors, one x at a time, from the input space X (if dealing with large data sets, pick a random subset of input patterns).
3. CR Similarity Matching. Find the best-matching (winning neuron) $i(x)$ by using the minimum-distance Euclidean criterion between the weights:

$$i(x) = \arg\min_{j} \left\| x(n) - w_j \right\|,\ j = 1,2,...,l \qquad (4.1)$$

   where l is the number of neurons in the lattice.
4. CR Updating. Adjust the synaptic weight vectors of all neurons by using the update formula

$$\sigma(n) = \sigma_0 + \frac{t}{t_{\max}}\left(\sigma_{final} - \sigma_0\right) \qquad (4.2)$$

$$h_{j,i(x)}(t) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2(t)}\right)\ where\ d_{j,i} = \left\| e_j - e_i \right\| \qquad (4.3)$$

$$\eta = \eta_0 + \frac{t}{t_{\max}}\left(\eta_{final} - \eta_0\right), \quad t = 0,1,2,..., \qquad (4.4)$$

$$\Delta W_{j,i(x)}^{t+1} = \eta(t) h_{j,i(x)}(t)\alpha(x(t) - w_j(t)) \qquad (4.5)$$

   where $\sigma(t)$ in equation (4.2) is the standard deviation of the weight-based Gaussian neighborhood function $h_{j,i(x)}$ given in equation (4.3), σ_0 is the initial value of σ, and $t_{\max}$ is a predefined number of iterations. The neighborhood function shrinks with time, due to the fact that $\sigma(t)$ decreases linearly. The weight-based neighborhood function $h_{j,i(x)}$ satisfies the following two requirements:

a) It is symmetric about the maximum point defined by $d_{j,i} = 0$, where $d_{j,i}$ is the weight-based distance between neurons *j* and *i* on the grid, given by $d_{j,i} = \| e_j - e_i \|$, where e_j defines the value of the excited weight vector *j* and e_i defines the value of the winning weight vector *i*.

b) The size of the weight-based neighborhood decreases monotonically with increasing lateral distance $d_{j,i}$, decaying to zero for $d_{j,i} \to \infty$ (this is a necessary condition for convergence).

In equation (4.4) $\eta(t)$ denotes the learning-rate parameter that starts with initial value η_0 and then decreases linearly with increasing time. Equation (4.5) describes the weight update for neuron *j* due to the input vector *x* at time $t+1$. We accumulate all updates (for the weights of all *l* neurons on the lattice due to all input vectors) and then apply them with a synchronous update in the outer loop (Fig. 4.1).

$$W^{t+1}_{j,i(x)} = W^{t}_{j,i(x)} + \Delta W^{t+1}_{j,i(x)}, \quad j = 1,2,\ldots,l \ \ and \ \ \forall x \in X \qquad (4.6)$$

5. <u>CR Convergence Test</u>. Iterate steps 2 through 5 until some convergence criterion has been reached (for example for some predefined number of iterations).

4.2. Explanation of the CR phase.

Let's assume we have an $n \times n$ matrix W of spatially organized weight vectors each of size $1 \times m$, and we wish to identify clusters of similar adjacent weight vectors The matrix W is a 2D representation of the multidimensional input space, such as created by an SOM, where weight vectors are spatially organized by similarity. To facilitate identification of potential clusters in W, we will use the additional CR phase. In this phase we define a new neighborhood function as given in (4.3). In equation (4.3), σ is the standard deviation of the Gaussian weight-based neighborhood. For the CR phase, we use a linear decay function for σ given in (4.2). In equation (4.3) the vector e_j defines the weight vector of excited neuron j and e_i defines the weight vector of winning neuron i. Equation (4.3) is similar to the neighborhood function of SOM given in equation (2.3), with the exception that in the SOM algorithm $d_{i,j}$ is the *topological distance* between the neurons given by $d_{i,j} = \left\| r_j - r_i \right\|$, where the discrete vector r_j defines the *position* of excited neuron j and r_i defines the discrete *position* of the winning neuron i. In contrast in equation (4.3) of the CR phase, the distance is based on the *values of the weight* vectors themselves ($d_{j,i} = \left\| e_j - e_i \right\|$). The CR phase neighborhood function includes neurons with similar weight vectors, regardless of their position in the grid so is no longer necessarily symmetric around the winning neuron, as in the SOM.

After the first phase of SOM the weight vectors are topologically ordered but boundaries between clusters may be diffuse. During the CR phase the neurons with similar weight vectors are adjusted further towards each other, which results in smaller distances between the members of the same cluster and larger distances between the members of different clusters. This phenomenon sharpens and clarifies the cluster boundaries.

In Fig. 4.2, we illustrate how the neighborhood function for the SOM neighborhood given in equation (2.3) (left) and for the CR phase given in equation (4.3) (right) change over time (y-axes), assuming the lattice radius is 3 nodes. The SOM algorithm starts with an initial σ_0 value equal to the "radius" of the lattice and decays over time exponentially (see equation (2.2) in Chapter 2). For the CR phase, σ starts with an initial value 0.1 and decays linearly (rather then exponentially) to 0.01 (equation (4.2)), since empirical results showed that linear decay of σ for the CR phase yielded better results. This is due to the fact that in the SOM the weight vectors that are in the neighborhood of the winning neuron get updated inversly proportional to the difference between their weights and the weight of the winning neuron, while in the CR phase the weights are updated directly proportional to the difference between their weights and the weight of the winning neuron, since the neighborhood function itself is weight-based. Given this fact, the exponential decay of the neighborhood function is more appropritate for the SOM algorithm, to facilitate a transition from self-organization to fine-tunning of the self-organized neurons. In the CR phase, a larger neighborhood can be considered for more iterations, as neurons are updated in

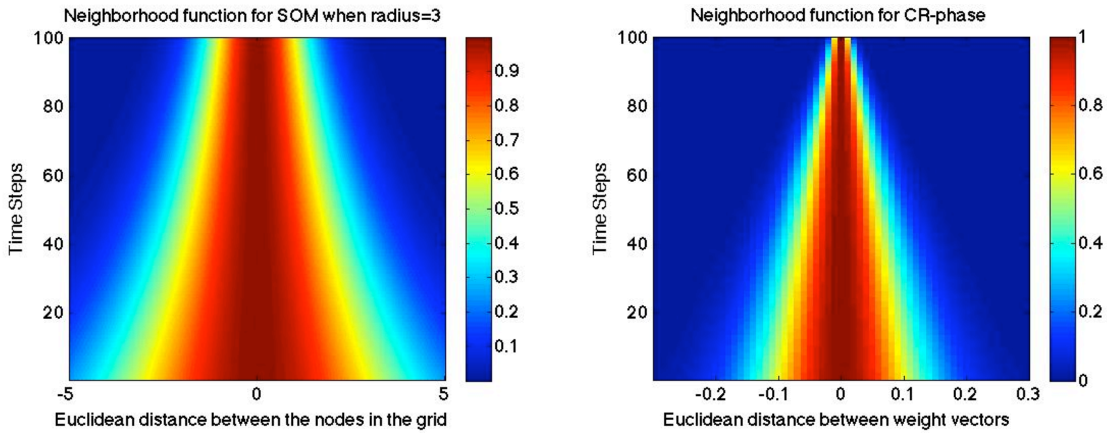


Figure 4.2 Visualization of neighborhood function for SOM (on left) and CR phase (on right).

proportion to their closeness to the winning node, so will not disrupt the self-organization. In Fig. 4.3 we show a few examples to demonstrate the differences between the neighborhood function for the SOM and CR phases. In the left panel we show examples of an SOM neighborhood around the winning neuron for iterations 1750 (Fig. 4.3a,b), 1800 (Fig. 4.3c,d) and 2000 (Fig. 4.3e,f). On the right panel we show what the CR phase neighborhood *would* look like for the same iterations, even though in practice the CR phase follows the SOM and thus would never be applied to iterations 1750 or 1800. In Figures 4.3a) amd 4.3b) the winning node is marked with an asterisk. Note that in the SOM, the neighborhoods are symetrically centered around the winning node (Fig. 4.3a,c,e), whereas in the CR phase neighborhoods may be assymetrical (Fig. 3.3b,d,f). As we increase the number of iterations (Fig. 4.3c,d,e,f), the topological neighborhood (left) shrinks in a predictable manner for all winning neurons, while the CR neighborhood changes depending on the distances between each weight vector and the weight of the winning neuron. Fig. 4.3e,f demonstrate a typical starting situation for the CR phase, where the topological neighborhood has been reduced to 1 (only the winning node) in the SOM.

In Fig. 4.4, we show the weight-based neighborhood at iteration 10 of the CR phase (Fig. 4.4a) and the updates of the weight vectors at that iteration for all 13 dimensions (Fig. 4.4 b-n). Note that the size of the updates applied to the neurons are different for different feature dimensions. This is due to the fact that each weight vector in the neighborhood of the winning neuron updates its weight for each dimension relative to the difference of the weight of that neuron and the winning neuron in that dimension.

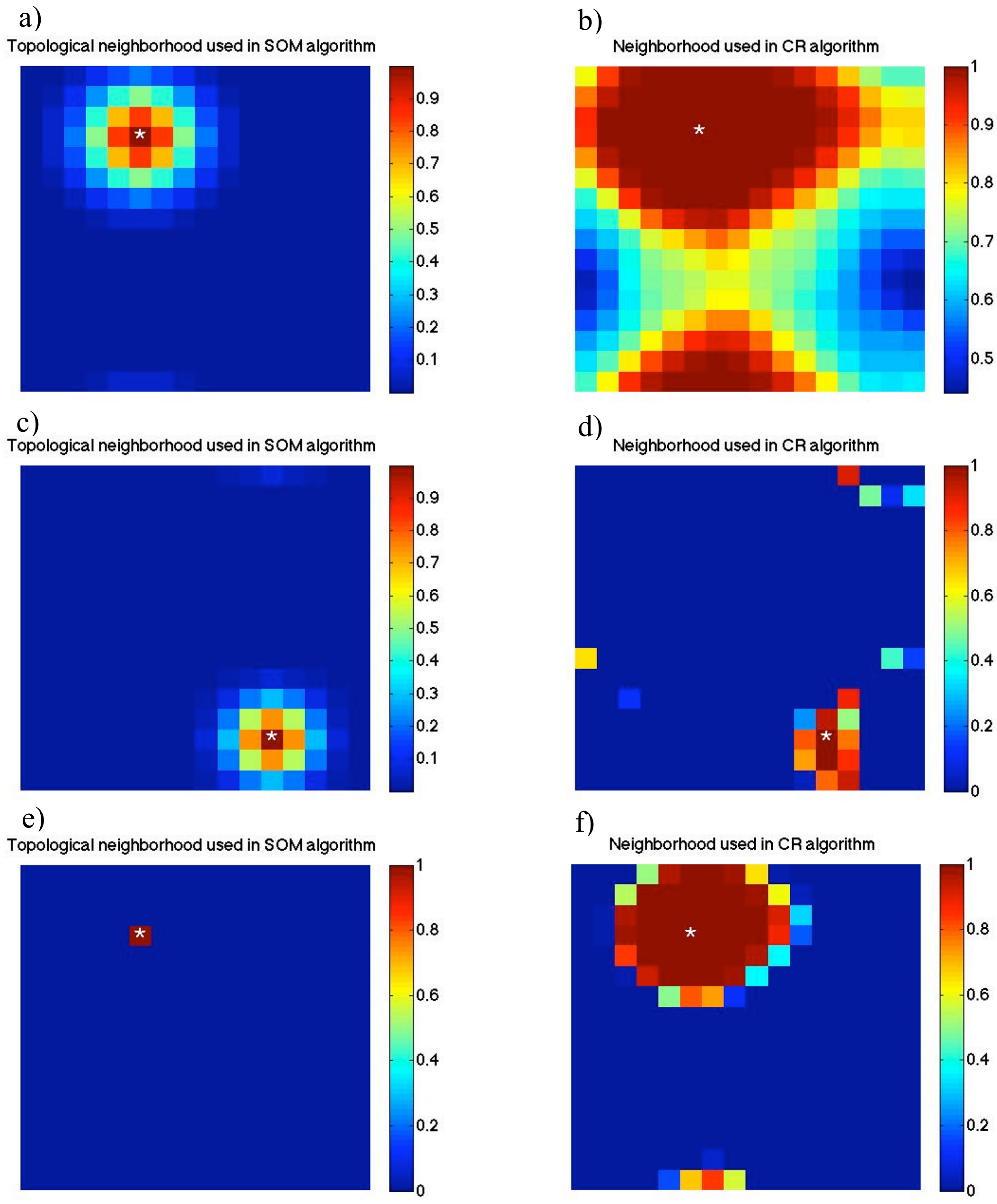


Figure 4.3 Examples of topological neighborhood (left panel) and weight-based neighborhood (right panel). a) and b) show the neighborhood for iteration number 1750, c) and d) for iteration number 1800, e) and f) for iteration number 2000.

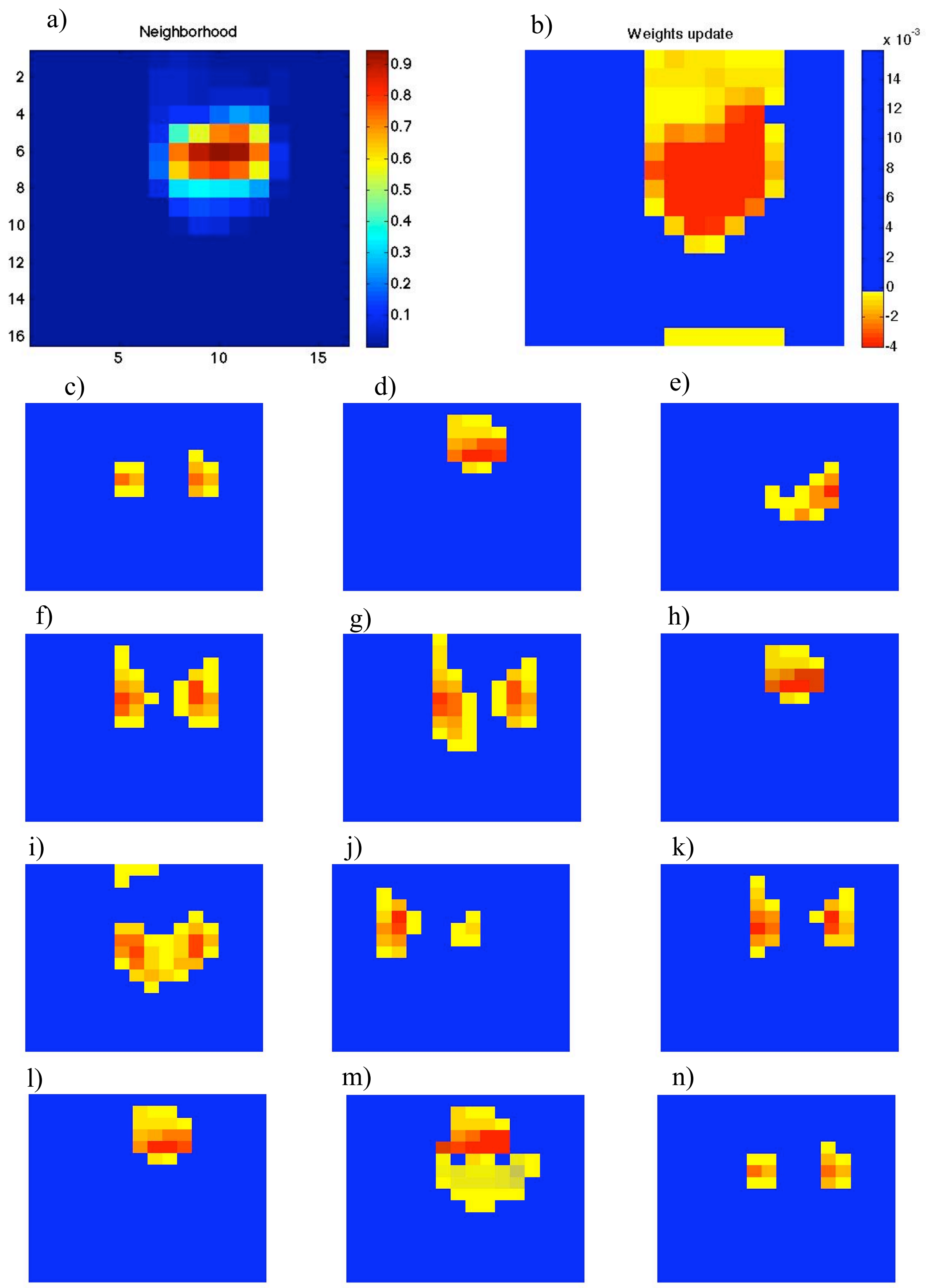


Figure 4.4 a) Shows the neighborhood at step 10 in CR phase; b) through n) show how the weights are updated for all 13 attributes of weight vectors for that step. The color scale of c) through n) are the same as shown in b).

4.3. Differences between SOM algorithm and Cluster Reinforcement Phase Algorithm.

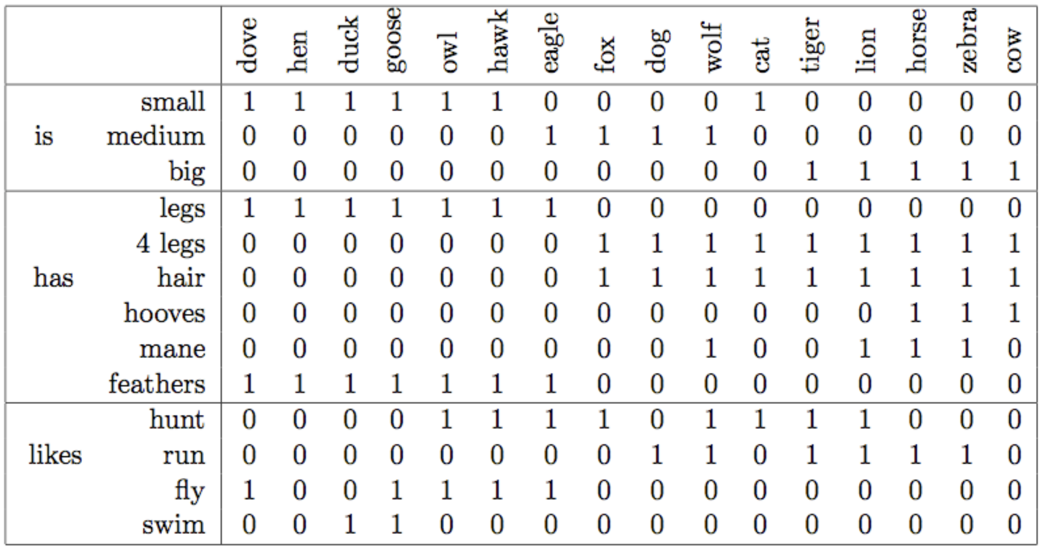
Although many aspects of the SOM and CR algorithms are similar, there are essential differences that are highlighted here:

In the initialization step (CR step 1), we start with trained SOM weight vectors W instead of random weight vectors (as in SOM step 1). In the sampling step (CR step 2) instead of drawing a sample x from input space (as in SOM step 2), we take all the input vectors and then chose each input element once for sampling; when using large datasets one can sample from a randomly selected subset of the input records, rather then the entire input space (e.g., see Chapter 6). The similarity matching step (CR step 3) is as in the SOM algorithm. In the CR phase, we elect to use synchronous updates rather than asynchronous updates (as in the SOM) in the inner loop, to avoid bias based on sampling order and obtain consistent results. In the updating step (CR step 4), we use the weight-based neighborhood function given in equation (4.3) with an additional scaling parameter α (see equation 14). Readers who wish to add a CR phase to their SOM can easily copy and edit their SOM updating code, as long as they made the required changes summarized above.

4.4. Demonstration of CR phase on a Benchmark Problem

We use the classic Benchmark animal clustering problem [39] to show the cluster sharpening features of the CR phase. Consider the set of data given in Table 1, which pertains to a number of different animals. Each column of the table is a binary feature vector x_a describing an animal, based on the presence (=1) or absence (=0) of each of

Table 1. Animal names and their attributes



		dove	hen	duck	goose	owl	hawk	eagle	fox	dog	wolf	cat	tiger	lion	horse	zebra	cow
is	small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
has	legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
likes	hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

13 different attributes given in the rows. For example the binary feature vector x_s for the animal hen is

$$x_a = [\ 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0\ ]^T$$

The attribute pairs "feather" and "two legs", as well as "four legs" and "hair", are completely correlated in this dataset, while "two legs" and "four legs" are mutually exclusive. Also please note that "hawk" and "owl" as well as "horse" and "zebra" have identical feature vectors. The animal species designation is specified by a column vector x_s, whose *k*th element, representing animal k = 1,2,…16, is given a fixed value of 0.2; the remaining elements are all set equal to zero. For example the animal species vector x_s for hen is

$$x_s = [\ 0 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ]^T$$

The value 0.2 is chosen to ensure this does not dominate the attribute vector. The input vector x for each animal is a vector of 29 elements, representing a concatenation of the attribute vector x_a and the species vector x_s, as shown by

$$x = \begin{bmatrix} x_s \\ x_a \end{bmatrix} = \begin{bmatrix} x_s \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ x_a \end{bmatrix} \qquad (4.7)$$

Finally, each input vector x is normalized to unit length. We adopt this implementation of species designation and normalization in order to stay consistent with previous publications [23, 30, 45]. However, as we will discuss later, this introduces some undesirable artifacts into the results of the SOM.

In Fig. 4.5a and 4.6a we show the resulting *U-matrix* with animal names written over the winning neurons using different random seeds. After 2000 iterations of SOM training (including the convergence phase) in Fig. 4.5b and 4.6b we show the *B-matrix*, defined in Chapter 3, for the same map. Note that the grid is toroidal in both dimensions. As one can see in Fig. 4.5b) and 4.6 b), after the SOM many of the cluster boundaries using the *B-matrix* method have multiple lines. In Fig. 4.5c,d) and 4.6c,d) we show the *U-matrix* (on left) and *B-matrix* (on right) respectively after additional 100 iterations of the CR phase. While the CR phase sharpens cluster boundaries in the *U-matrix* (e.g., compare Fig 4.5a to Fig 4.5c), this improvement is even more evident in the *B-matrix* (e.g., compare Fig. 4.5b to Fig. 4.5d). In Fig. 4.5d) and 4.6 d), one can readily identify clusters at various levels of separation, where the darker lines denote stronger separation. There is clear separation between the birds and mammals, and there is a smaller separation within some clusters, such as between {horse, zebra, cow} and {lion, tiger, cat, wolf, fox, dog}. The results are consistent with the data in Table 1.

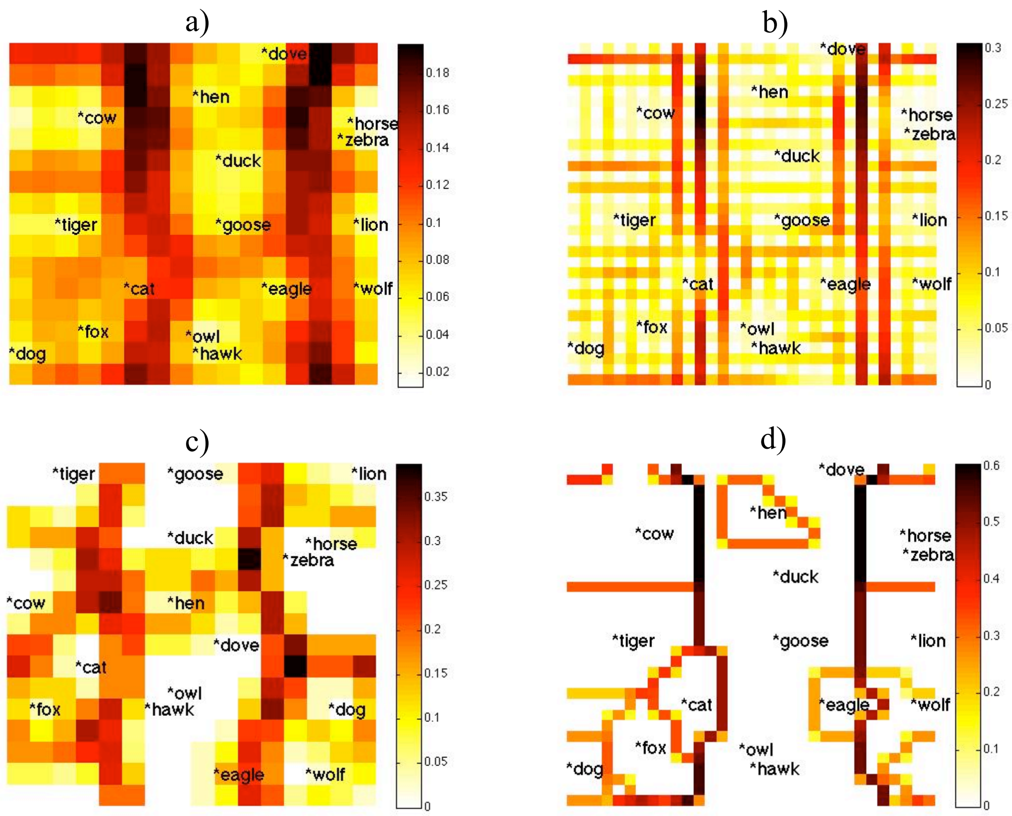
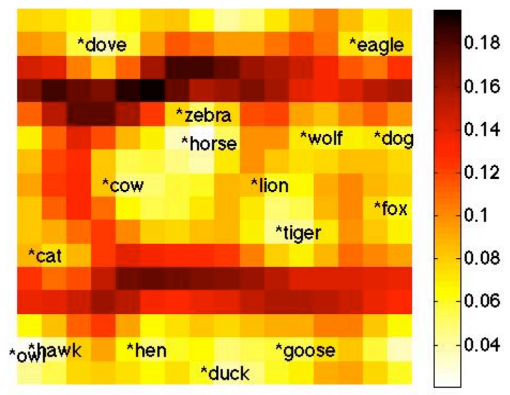
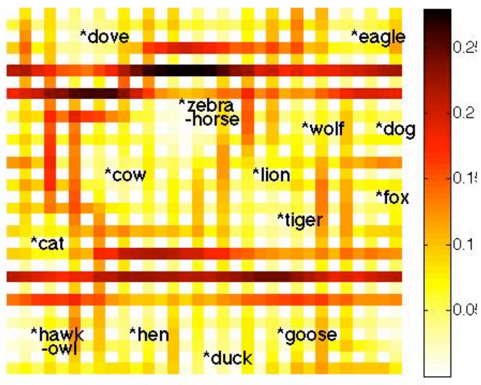


Figure. 4.5 a) *U-matrix* of Animal Data after SOM training. b) *B-matrix* of Animal Data after SOM training. c) *U-matrix* of Animal Data after CR phase. d) *B-matrix* of Animal Data after CR phase.

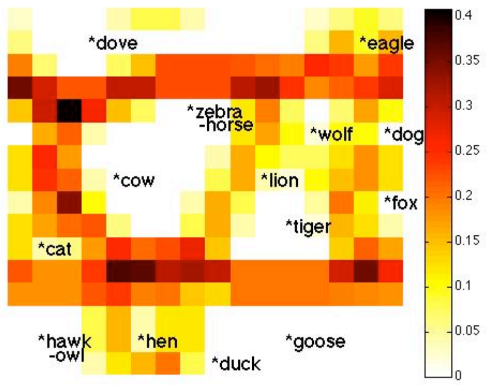
a)



b)



c)



d)

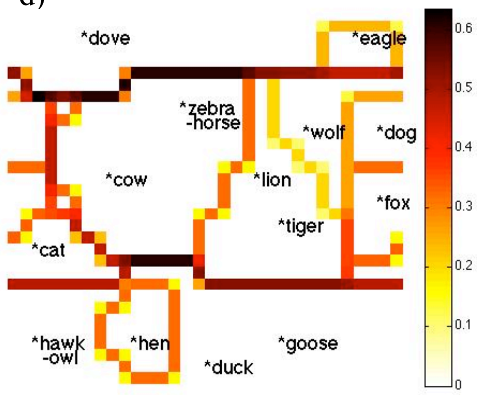


Figure 4.6 a) *U-matrix* of Animal Data after SOM training. b) *B-matrix* of Animal Data after SOM training. c) *U-matrix* of Animal Data after CR phase. d) *B-matrix* of Animal Data after CR phase. Same as Fig. 11, but with different random seed.

Notice in both of these runs that hen has an intermediate strength boundary around it within the bird cluster (Fig. 4.5 and 4.6). In this case, the boundary around the hen is an artifact of the species designation and normalization of data. In Fig. 4.7 we show raw attribute distances between animals before normalization. For the reader's convenience rows are labeled with animals names. Each entry shows the 1-norm (city block distance) between the pairs of animals. At the end of each row, we show the sum of the row elements in colored text. The second row represents the animal "hen". The sum of the distances between hen and all the other animals in the list is 80 (it is colored blue). It is the third smallest sum in the sums column.

	dove	*hen*	*duck*	*goose*	*owl*	*hawk*	*eagle*	*fox*	*dog*	*wolf*	*cat*	*tiger*	*lion*	*horse*	*zebra*	*cow*	***SUM***
dove	0	1	2	1	1	1	3	8	8	10	6	9	10	9	9	8	***86***
hen	1	0	1	2	2	2	4	7	7	9	5	8	9	8	8	7	**80**
duck	2	1	0	1	3	3	5	8	8	10	6	9	10	9	9	8	***92***
goose	1	2	1	0	2	2	4	9	9	11	7	10	11	10	10	9	***98***
owl	1	2	3	2	0	0	2	7	9	9	5	8	9	10	10	9	***86***
hawk	1	2	3	2	0	0	2	7	9	9	5	8	9	10	10	9	***86***
eagle	3	4	5	4	2	2	0	5	7	7	7	8	9	10	10	9	***92***
fox	8	7	8	9	7	7	5	0	2	2	2	3	4	5	5	4	***78*** ***78***
dog	8	7	8	9	9	9	7	2	0	2	4	3	4	5	5	4	***86***
wolf	10	9	10	11	9	9	7	2	2	0	4	3	2	5	5	6	***94***
cat	6	5	6	7	5	5	7	2	4	4	0	3	4	5	5	4	***72***
tiger	9	8	9	10	8	8	8	3	3	3	3	0	1	4	4	3	***84***
lion	10	9	10	11	9	9	9	4	4	2	4	1	0	3	3	4	***92***
horse	9	8	9	10	10	10	10	5	5	5	5	4	3	0	0	1	***94***
zebra	9	8	9	10	10	10	10	5	5	5	5	4	3	0	0	1	***94***
cow	8	7	8	9	9	9	9	4	4	6	4	3	4	1	1	0	***86***

Figure 4.7. This matrix demonstrates the distance between the animals before normalization.

In Fig. 4.8 we show the attribute distances between animals after inclusion of the species vector x_a and normalization. Hen now has the highest value in the sums list (20.88). This occurs due to the fact that hen has the fewest (only 3) non-zero attributes (Table 1), so the species designation value of 0.2 becomes disproportionally large relative to its attribute vector. This is the reason why we see a separate sub-cluster around hen with the birds cluster (Fig. 4.5 and 4.6). While this highlights the importance of choosing one's data representation carefully, it does not reflect negatively on the CR phase or *B-matrix* visualization, which is the focus of this thesis.

	dove	*hen*	*duck*	*goose*	*owl*	*hawk*	*eagle*	*fox*	*dog*	*wolf*	*cat*	*tiger*	*lion*	*horse*	*zebra*	*cow*	***SUM***
dove	0	0.46	0.48	0.38	0.38	0.38	0.76	1.9	1.9	1.92	1.43	1.91	1.92	1.91	1.91	1.9	***19.55***
hen	0.46	0	0.46	0.75	0.75	0.75	1.13	1.89	1.89	1.91	1.41	1.9	1.91	1.9	1.9	1.89	***20.88***
duck	0.48	0.46	0	0.38	0.76	0.76	1.14	1.9	1.9	1.92	1.43	1.91	1.92	1.91	1.91	1.9	***20.7***
goose	0.38	0.75	0.38	0	0.38	0.38	0.77	1.91	1.91	1.93	1.53	1.92	1.93	1.92	1.92	1.91	***19.93***
owl	0.38	0.75	0.76	0.38	0	0	0.38	1.53	1.91	1.61	1.14	1.54	1.61	1.92	1.92	1.91	***17.75***
hawk	0.38	0.75	0.76	0.38	0	0	0.38	1.53	1.91	1.61	1.14	1.54	1.61	1.92	1.92	1.91	***17.75***
eagle	0.76	1.13	1.14	0.77	0.38	0.38	0	1.14	1.53	1.28	1.53	1.54	1.61	1.92	1.92	1.91	***18.97***
fox	1.9	1.89	1.9	1.91	1.53	1.53	1.14	0	0.48	0.63	0.48	0.76	0.95	1.14	1.14	0.95	***18.35***
dog	1.9	1.89	1.9	1.91	1.91	1.91	1.53	0.48	0	0.63	0.95	0.76	0.95	1.14	1.14	0.96	***19.98*** ***19.98***
wolf	1.92	1.91	1.92	1.93	1.61	1.61	1.28	0.63	0.63	0	0.95	0.64	0.32	0.96	0.96	1.27	***18.54***
cat	1.43	1.41	1.43	1.53	1.14	1.14	1.53	0.48	0.95	0.95	0	0.76	0.95	1.14	1.14	0.95	***16.95***
tiger	1.91	1.9	1.91	1.92	1.54	1.54	1.54	0.76	0.76	0.64	0.76	0	0.32	0.77	0.77	0.76	***17.80***
lion	1.92	1.91	1.92	1.93	1.61	1.61	1.61	0.95	0.95	0.32	0.95	0.32	0	0.64	0.64	0.95	***18.22***
horse	1.91	1.9	1.91	1.92	1.92	1.92	1.92	1.14	1.14	0.96	1.14	0.77	0.64	0	0	0.38	***19.60***
zebra	1.91	1.9	1.91	1.92	1.92	1.92	1.92	1.14	1.14	0.96	1.14	0.77	0,64	0	0	0.38	***19.60***
cow	1.9	1.89	1.9	1.91	1.91	1.91	1.91	0.95	0.95	1.27	0.95	0.76	0.95	0.38	0.38	0	***19.95***

Figure 4.8 This matrix demonstrates the distances between the animals after including species designation and normalization.

In Fig. 4.9 we show the *B-matrix* of Animal Data for different scaling parameters (α, from equation (4.5)) values from 0.25 to 8. We observed that when the scaling parameter value is half of the width of the grid (i.e., equal to 8 in this grid) the results were better than for smaller numbers. Previous experimentation showed that results were not further improved for higher α; therefore for computational efficiency, we use half of the width of the grid for α in all subsequent experiments.

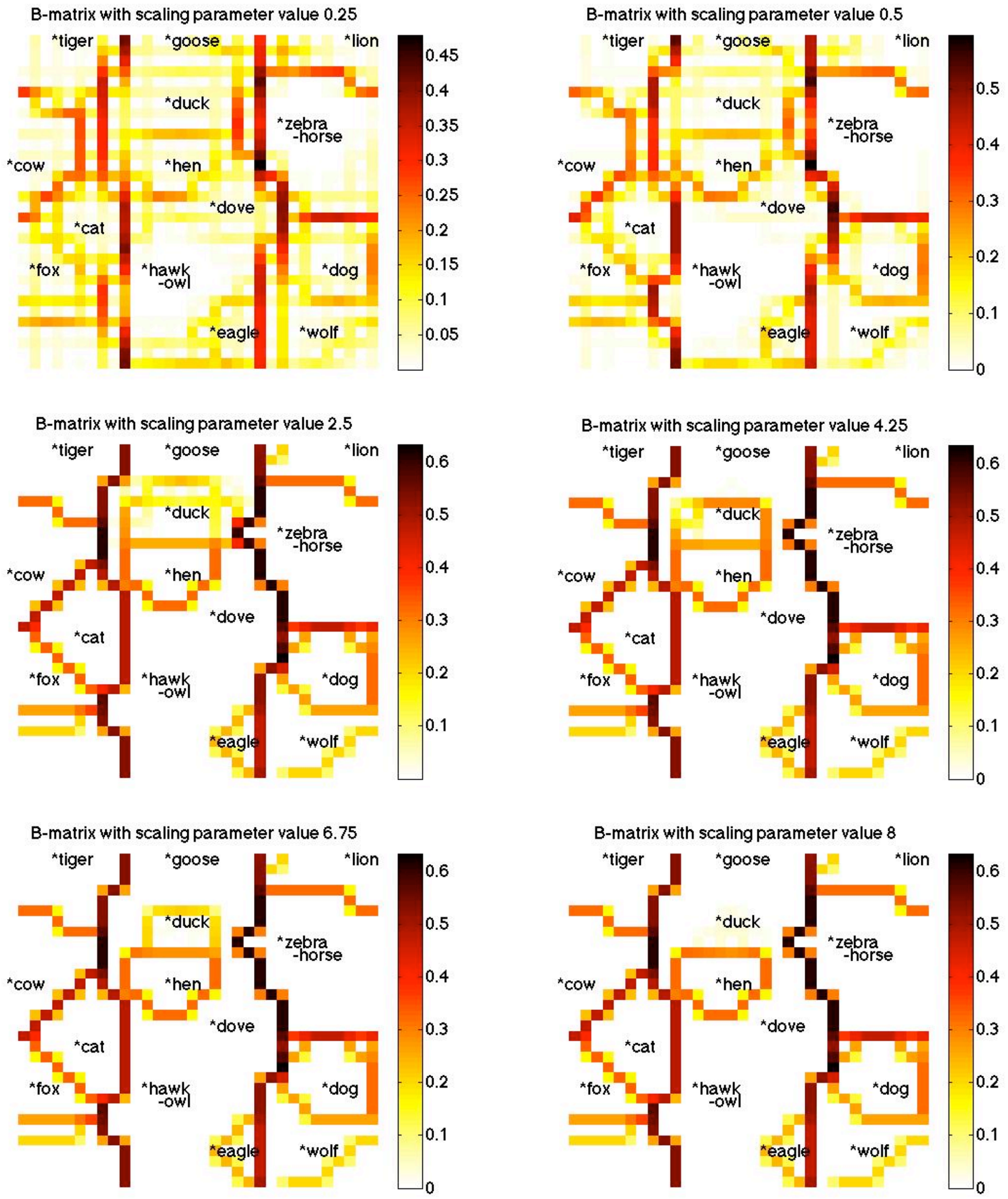


Figure 4.9 Shows B-matrix of Animal Data for different scaling parameters from 0.25 (upper left) to 8 (lower right). See figure titles.

4.5. Alternative visualization of component planes in self-organizing maps using lines of different weights to highlight the separation between the clusters.

In this section we introduce a new method for visualizing component planes from the SOM after applying the CR phase (for cluster sharpening) and generating the $2n \times 2n$ *B-matrix*. To demonstrate this visualization method, we will use the small example we introduced in Chapter 3 where we had an $n \times n$ matrix W of spatially organized scalar weights, (i.e., vectors each of size $m \times 1$ where m = 1), where weight vectors are spatially organized by similarity. We will also compare this visualization method with traditional contour visualization of the *U-matrix* generated from SOM. For the readers convenience, we repeat the example W matrix that we used in Chapter 3.

$$W = \begin{bmatrix} 1 & 3 & 26 & 26 & 26 \\ 1 & 3 & 27 & 28 & 28 \\ 2 & 2 & 18 & 18 & 18 \\ 10 & 10 & 17 & 18 & 18 \\ 11 & 11 & 17 & 17 & 18 \end{bmatrix} \qquad (4.8)$$

Having the W matrix of spatially organized weight vectors we first calculate the *B-matrix*, which we show in equation (4.9) with dashes at positions corresponding to the elements of W from equation (4.8). As mentioned in Chapter 3, in the *B-matrix* we use three kinds of distances; horizontal distances (between the horizontal elements of W, vertical distance (between the vertical elements) and diagonal distances (which are the mean distances between the four diagonally positioned elements (see Fig. 3.1a in Chapter 3).

$$B = \begin{bmatrix} - & 2 & - & 23 & - & 0 & - & 0 & - & 25 \\ 0 & 2 & 0 & 23.5 & 1 & 1.5 & 2 & 2 & 2 & 26 \\ - & 2 & - & 24 & - & 1 & - & 0 & - & 27 \\ 1 & 1 & 1 & 20 & 9 & 9.5 & 10 & 10 & 10 & 21.5 \\ - & 0 & - & 16 & - & 0 & - & 0 & - & 16 \\ 8 & 8 & 8 & 11 & 2 & 1 & 0 & 0 & 0 & 12 \\ - & 0 & - & 6 & - & 2 & - & 0 & - & 8 \\ 1 & 1.5 & 2 & 5 & 0 & 1 & 2 & 1 & 0 & 7.5 \\ - & 1 & - & 4 & - & 0 & - & 2 & - & 7 \\ 10 & 9.5 & 9 & 13.5 & 10 & 10 & 10 & 9 & 8 & 16 \end{bmatrix} \tag{4.9}$$

To draw lines that contain information about the clusters in the W matrix, we create a grid of $n \times n$ cells representing the elements of W. For each horizontal and vertical distance in the *B-matrix*, we draw a grid line between the elements of W with the thickness of the line proportional to the horizontal or vertical distance. For each of the diagonal distances in the *B-matrix* we draw a dot with the thickness of the dot proportional to the value of the diagonal distance. We then apply a threshold parameter θ, which allows us to control the level of cluster resolution. Only distances larger then θ are drawn as lines or dots. We show the algorithm of this visualization technique in Fig. 4.10.

In Fig. 4.11 we show the visualization of the W matrix (or one of its component planes), using the algorithm given above with two different thresholds and in Fig. 4.12 we show hierarchical visualization of cluster boundaries in the Animals Dataset using four different thresholds.

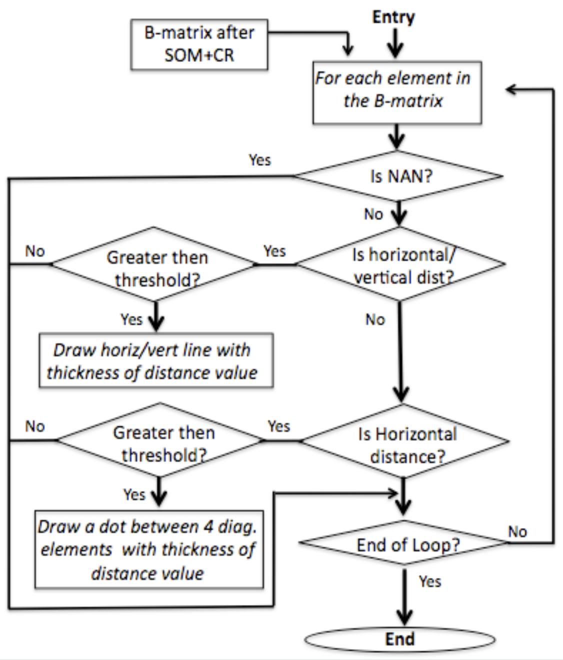

Entry
B-matrix after SOM+CR
For each element in the B-matrix
Is NAN?
Yes
No
Is horizontal/ vertical dist?
Yes
No
Greater then threshold?
No
Yes
Draw horiz/vert line with thickness of distance value
Is Horizontal distance?
Yes
Greater then threshold?
No
Yes
Draw a dot between 4 diag. elements with thickness of distance value
End of Loop?
No
Yes
End

Figure 4.10 Algorithm for visualizing B-matrix as grid lines.

This method allows us to show cluster boundaries in the *B-matrix* with grid lines superimposed on component planes, resulting in a highly informative visualization. The reader can easily see how various clusters are related via the underlying attributes, displayed in each component plane. This is not possible when using the SOM and *U-matrix* alone for two reasons. Primarily, prior to the CR phase the boundaries are too diffuse to be displayed as single grid lines, and secondly, the $n \times n$ values in the *U-matrix* do not correspond to the boundaries between $n \times n$ values in the weight matrix, so can not be displayed as grid lines. To illustrate this point in Fig. 4.13 we show the contour plot visualization of SOM/*U-matrix* (Fig. 4.13a), SOM/*B-matrix* (Fig. 4.13b), SOM+CR/*U-matrix* (Fig. 4.13c), and SOM+CR/*B-matrix* (Fig. 4.13d). In Fig. 4.14a) we show the contour plot overlain on one component plans for the SOM/*U-matrix* and in

Fig. 4.14b) we show the visualization of the same component plane using SOM+CR/*B-matrix* grid lines.

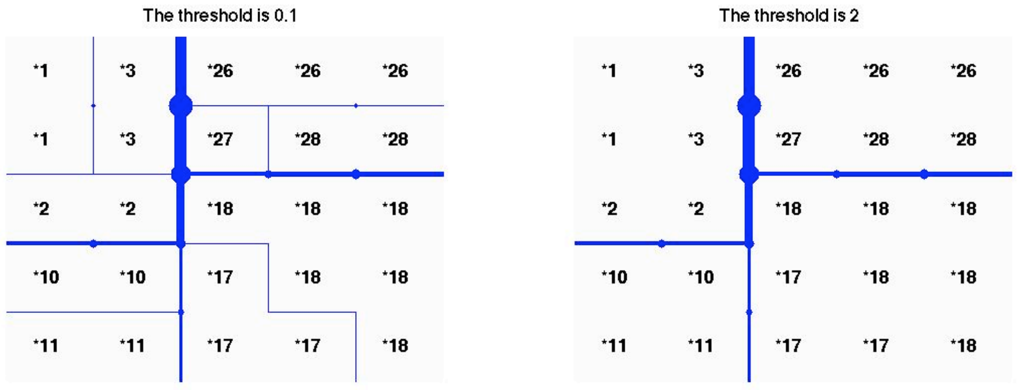


Figure 4.11 Visualization of *B-matrix* using lines when threshold is 0.1 (left) and 2 (right).

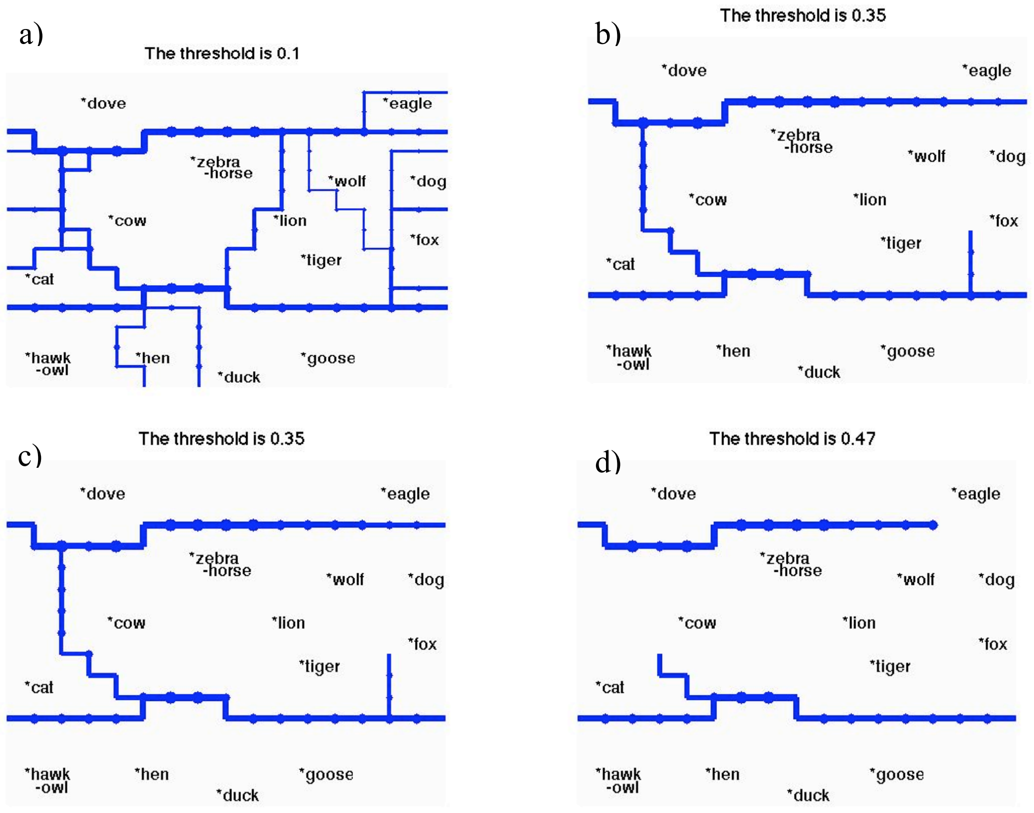


Figure 4.12 Hierarchical Visualization of *B-matrix* using lines for Animals Data set using four different thresholds (see figure titles).

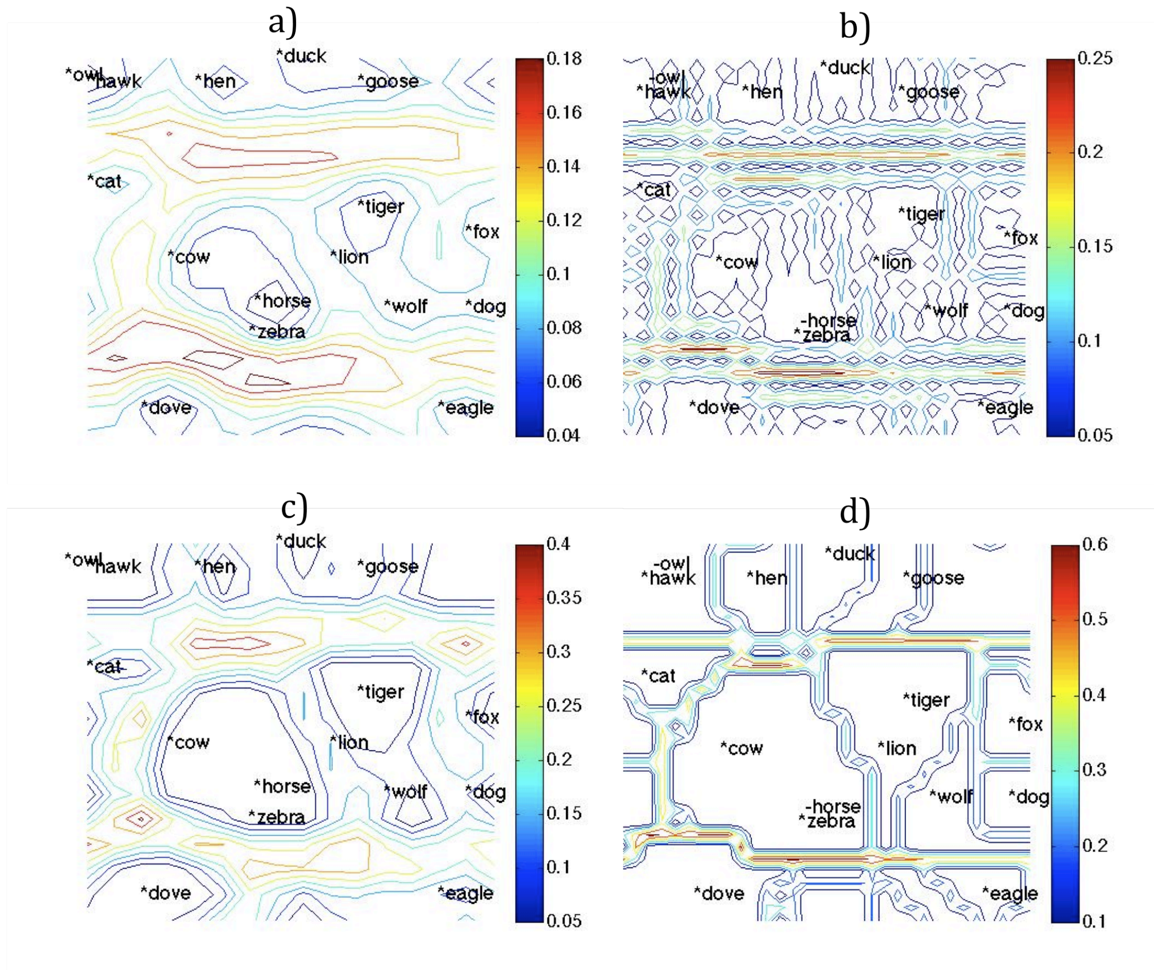


Figure 4.13 a) Contour plot for SOM/Umatrix, b) contour plot for SOM/*B-matrix*, c) contour plot for SOM+CR/*U-matrix*, d) contour plot for SOM+CR/*B-matrix*.

In Fig. 4.15 we point out that it is most informative to display the component planes resulting from the SOM *before* the CR phase with grid lines from the *B-matrix* resulting from *after* the CR phase, as shown for one component plane in Fig. 4.15a). For comparison, we show the same component plane *after* the CR phase in Fig. 4.15 b). Because the purpose of the CR phase is to make clusters more internally homogeneous with less diffuse boundaries (Fig. 4.15b), this can result in loss of information about within cluster heterogeneities (Fig. 4.15a). In Fig. 4.16 and Fig. 4.17 we show the remaining 12 component planes of the Animal Data from the SOM (before the CR phase) with the *B-matrix* (after the CR phase) shown as gridlines (threshold $\theta = 0.1$).

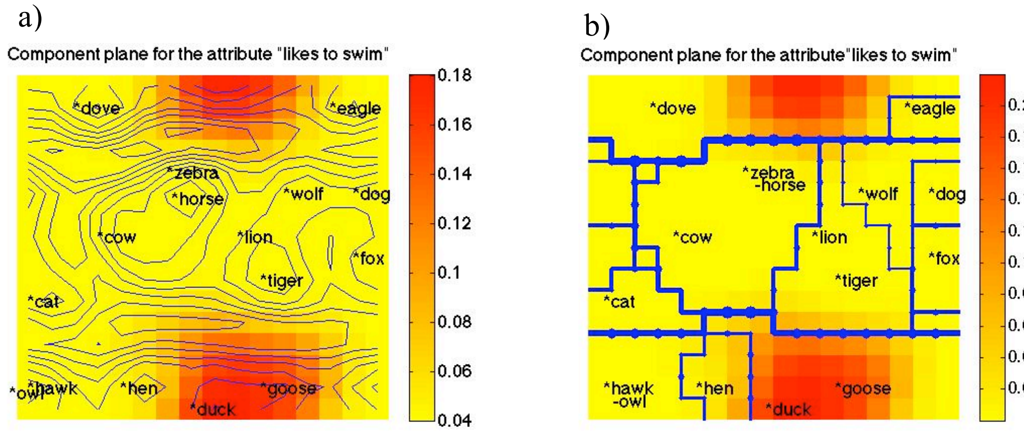


Figure 4.14a) Component plane for attribute "likes to swim" overlain with contour plot for SOM/*U-matrix*. b) same component plane of SOM overlain with *B-matrix* shown as grid lines, with a threshold of 0.1.

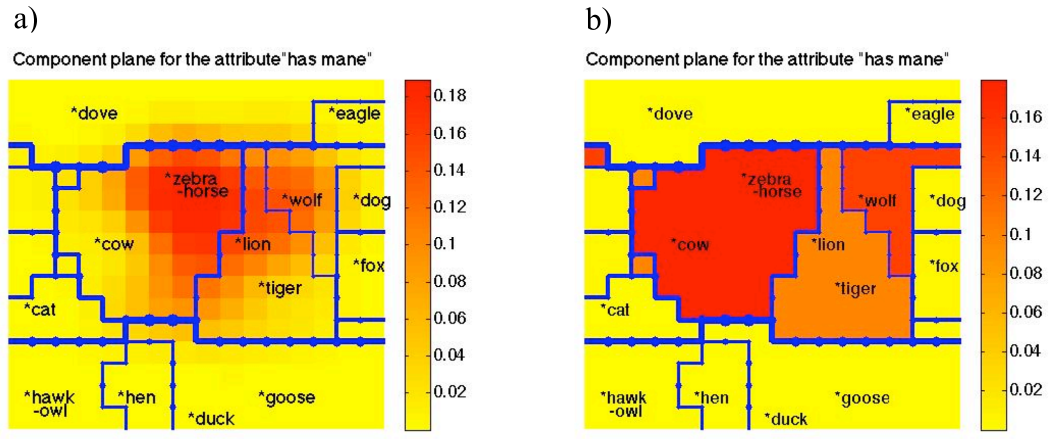


Figure 4.15 a) The component plane for the attribute "likes to swim " after SOM training overlain with *B-matrix* after SOM+CR training. b) The component plane for the same attribute after SOM+CR training overlain with *B-matrix* after SOM+CR training. Threshold is 0.1.

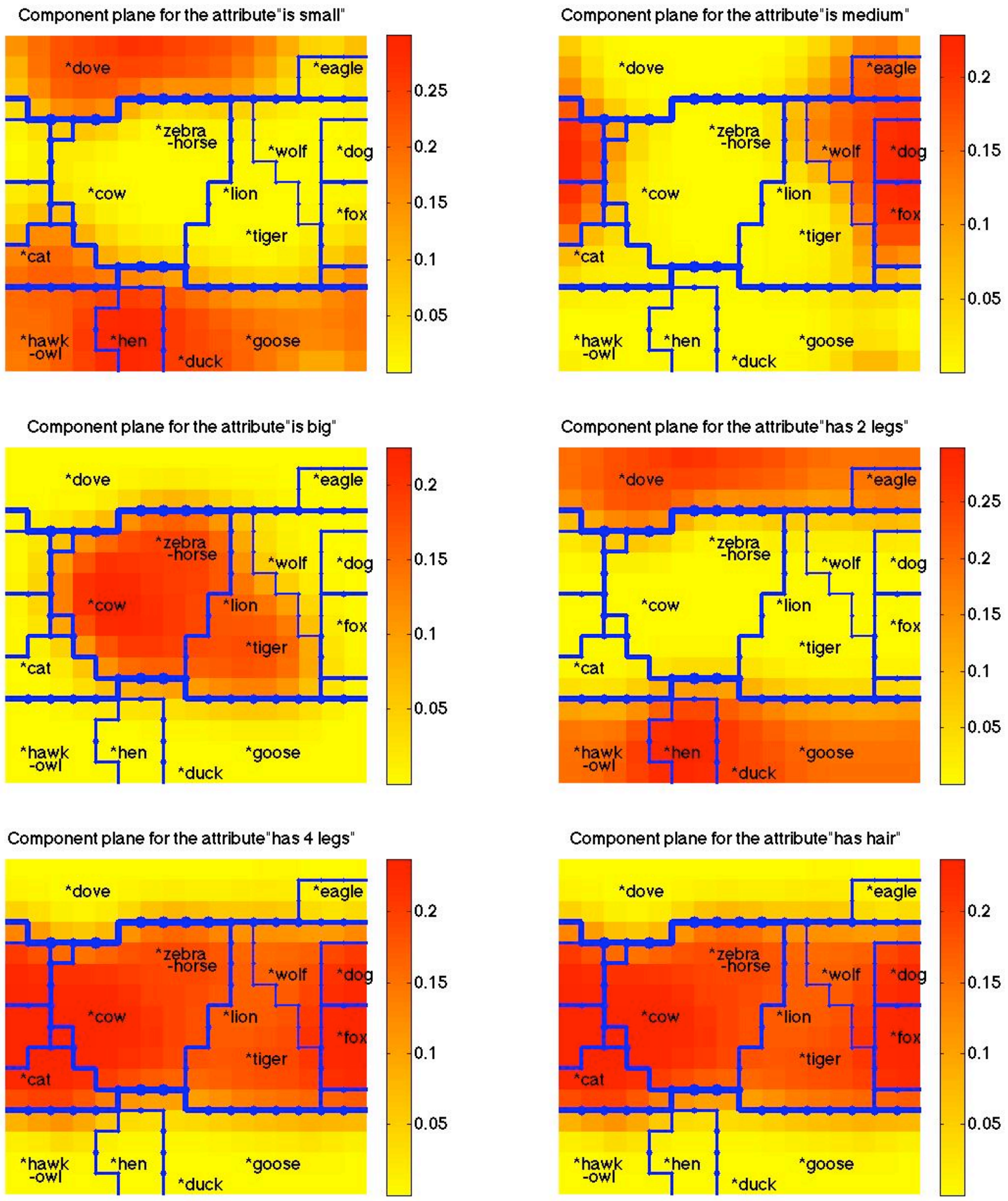


Figure 4.16 Component planes for attributes "is small", "is medium", "is big", "has 2 legs", "has 4 legs", "has hair" of Animal Data. The threshold is 0.1

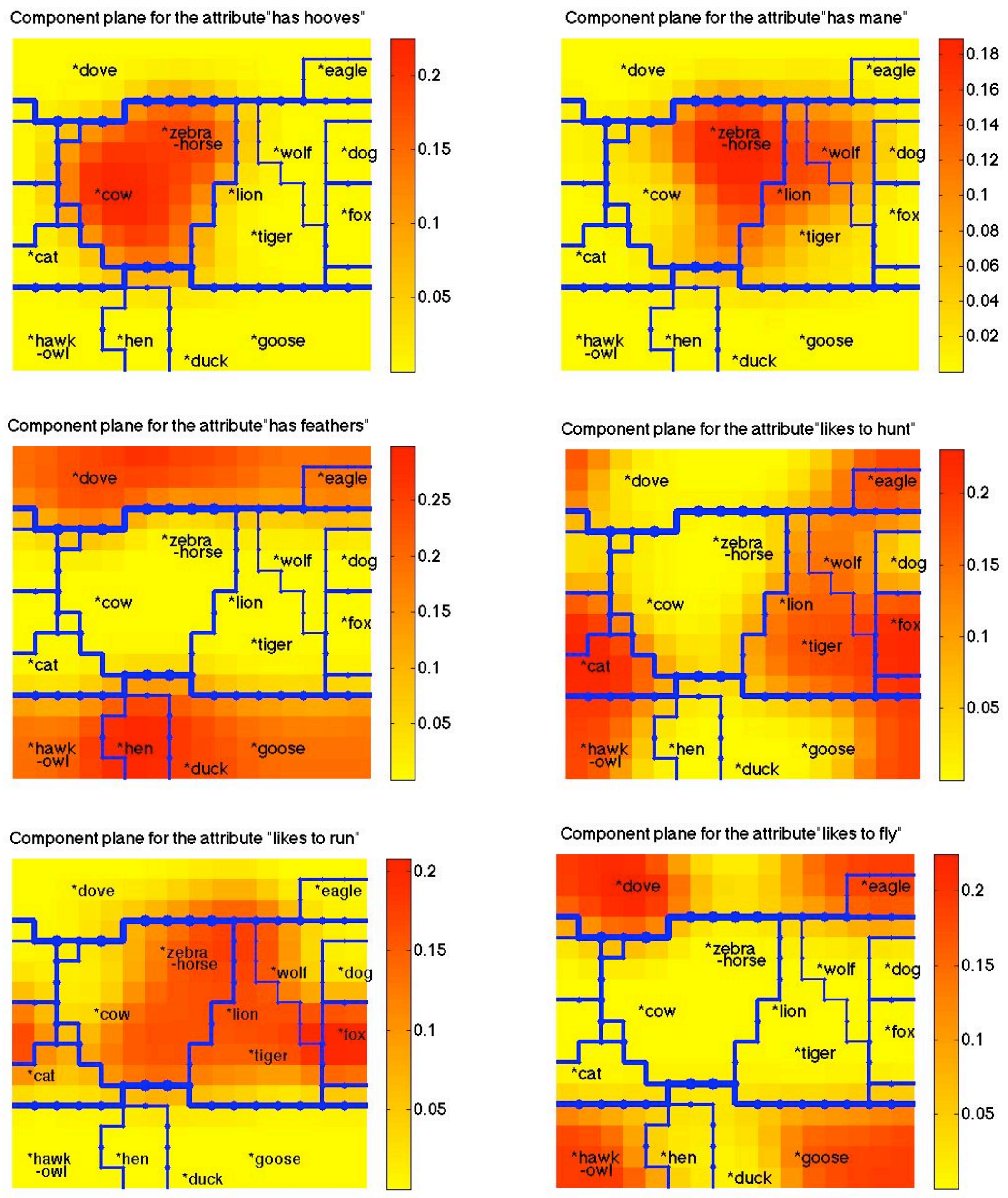


Figure 4.17 Component planes for all attributes "has hooves", "has mane", "has feathers", "likes to hunt", "likes to run", "likes to fly" of Animal Data. The threshold is 0.1.

CHAPTER 5: CLUSTERING WORDS BY PHONETIC CHARACTERISTICS

In this chapter we present a real world application of the SOM+CR phase with *B-matrix* visualization for data with binary values. In a Dichotic Hearing Test (DHT), auditory processing disorders are diagnosed by presenting pairs of words simultaneously through headphones to the patient. Currently only adhoc methods have been established for selecting test words or pairs of test words, using the phonetic features of words. One interesting question to answer is what words or pairs of words are more clinically predictive.

We use a dataset of 200 words, each described with 60 phonetic characteristics. The source of this dataset is from a DHT currently in use [17]. After SOM+CR, the resulting feature map that gives a 2D representation of the 60 dimensional space that can be used to identify which words belong to the same clusters and which clusters are adjacent.

5.1. The Dataset

The dataset consists of 200 one syllable words that have been manually characterized with binary phonetic features of up to 5 consonants. In Table 2 we show some samples from the dataset.

Table 2. A sample from the dataset of 200 words. The first column (attributes) represents the phonetic features of the words. The first row (words) shows 14 sample words from the dataset.

Attributes/words	globe	bed	bird	dad	food	head	mud	need	red	judge	knife	rough	chef	soft
C1nasal	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1fricative	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1stop	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1approximant	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1voiced	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1bilabial	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1labiodental	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1dental	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1alveolar	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1palatoalveolar	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1velar	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1laryngeal	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2nasal	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2fricative	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2stop	1	0	0	0	0	0	0	0	0	0	0	0	0	0
C2approximant	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2voiced	1	0	0	0	0	0	0	0	0	0	0	0	0	0
C2bilabial	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2labiodental	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2dental	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2alveolar	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2palatoalveolar	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2velar	1	0	0	0	0	0	0	0	0	0	0	0	0	0
C2laryngeal	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C3nasal	0	0	0	0	0	0	1	1	0	0	1	0	0	0
C3fricative	0	0	0	0	1	1	0	0	0	1	0	0	1	1
C3stop	0	1	1	1	0	0	0	0	0	1	0	0	0	0
C3approximant	1	0	0	0	0	0	0	0	1	0	0	1	0	0
C3voiced	1	1	1	1	0	0	1	1	1	1	1	1	0	0
C3bilabial	0	1	1	0	0	0	1	0	0	0	0	0	0	0
C3labiodental	0	0	0	0	1	0	0	0	0	0	0	0	0	0
C3dental	0	0	0	0	0	0	0	0	0	1	0	0	0	0
C3alveolar	1	0	0	1	0	0	0	1	0	1	1	0	0	1
C3palatoalveolar	0	0	0	0	0	0	0	0	1	0	0	1	1	0
C3velar	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C3laryngeal	0	0	0	0	0	1	0	0	0	0	0	0	0	0
C4nasal	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C4fricative	0	0	0	0	0	0	0	0	0	1	1	1	1	1
C4stop	1	1	1	1	1	1	1	1	1	1	0	0	0	0
C4approximant	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C4voiced	1	1	1	1	1	1	1	1	1	1	0	0	0	0
C4bilabial	1	0	0	0	0	0	0	0	0	0	0	0	0	0
C4labiodental	0	0	0	0	0	0	0	0	0	0	1	1	1	1
C4dental	0	0	0	0	0	0	0	0	0	1	0	0	0	0
C4alveolar	0	1	1	1	1	1	1	1	1	1	0	0	0	0
C4palatoalveolar	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C4velar	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C4laryngeal	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C5nasal	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In order to familiarize the reader with different phonetic features that we used for clustering, we give a brief explanation about the meanings of the attributes.

Only consonants of each word are taken into consideration. The attributes are labeled in column 1 as follows: The first through fifth consonants of any given word are denoted by attributes starting with C1, C2, C3, C4, C5 accordingly. The rest of the attribute name describes one of 12 different features of the consonants. One interesting topic in phonetics is describing sounds and finding how the words fall into patterns using these descriptions. This is accomplished by describing how speech sounds are made and sorting them into groups. In order to form consonants, the airstream through the vocal tract must be obstructed in some way. Consonants can therefore be classified according to the place and manner of this obstruction [27]. The 12 different groups of consonants in our data set are defined below:

> **"Nasal** – Consider the consonants at the ends of "rang, ran, ram." When you say these consonants by themselves, note that the air is coming out through the nose. In the formation of these sounds, the point of articulatory closure moves forward, from velar in "rang," through alveolar in "ran," to bilabial in "ram." In each case, the air is prevented from going out through the mouth, but is able to go out through the nose because the soft palate, or velum, is lowered…
>
> **Fricative – (**Close approximation of two articulators so that the airstream is partially obstructed and turbulent airflow is produced). The mechanism involved in making these slightly hissing sounds may be likened to that involved when the wind whistles around a corner. The consonants in "fie, vie" (labiodental), "thigh, thy" (dental), "sigh, zoo" (alveolar), and "shy" (palato-alveolar) are examples of fricative sounds…

Stop – (Complete closure of the articulators involved so that the airstream cannot escape through the mouth)…. This kind of sound occurs in the consonants in the words “pie, buy”(bilabial closure), “tie, dye” (alveolar closure), and “kye, guy” (velar closure)…

Approximant – (An articulation in which one articulator is close to another, but with- out the vocal tract being narrowed to such an extent that a turbulent airstream is produced). In saying the first sound in "yacht" the front of the tongue is raised toward the palatal area of the roof of the mouth, but it does not come close enough for a fricative sound to be produced. The consonants in the word "we" (approximation between the lips and in the velar region) and, for some people, in the word "raw" (approximation in the alveolar region) are also examples of approximants…

Voiced - Sounds produced when the vocal cords are vibrating are said to be voiced, as opposed to those in which the vocal cords are apart, which are said to be voiceless. In order to hear the difference between a voiced and a voiceless sound, try saying a long v sound… and compare it with a long f sound…

Bilabial – (Made with the two lips). Say words such as "pie, buy, my" and note how the lips come together for the first sound in each of these words…

Labiodentals – (Lower lip and upper front teeth). Most people, when saying words such as “fie, vie,” raise the lower lip until it nearly touches the upper front teeth …

Dental - (Tongue tip or blade and upper front teeth). Say the words "thigh", "thy". Some people (most speakers of American English) have the tip of the tongue protruding between the upper and lower front teeth; others (most speakers of British English) have it close behind the upper front teeth. Both these kinds of sounds are normal in English, and both may be called dental…

Alveolar – (Tongue tip or blade and the alveolar ridge). Again there are two possibilities in English. You may pronounce words such as "tie, die, nigh, sigh, zeal, lie" using the tip of the tongue or the blade of the tongue…

Palatoalveolar – (Tongue blade and the back of the alveolar ridge). Say words such as "shy, she, show." During the consonants, the tip of your tongue may be down behind the lower front teeth, or it may be up near the alveolar ridge, but the blade of the tongue is always close to the back part of the alveolar ridge…

Velar – (Back part of the tongue and soft palate). The consonants that have the farthest back place of articulation in English are those that occur at the end of "hack", "hang" and "hag" …" (Ladefoged 1975, pages 2-9)[27].

It is important to mention that some consonants can have several features at the same time. For example the words "buy" and "pie" are "bilabial" and "stop". In our dataset if the given feature is true for the given word then the value 1 is used, otherwise 0 is used.

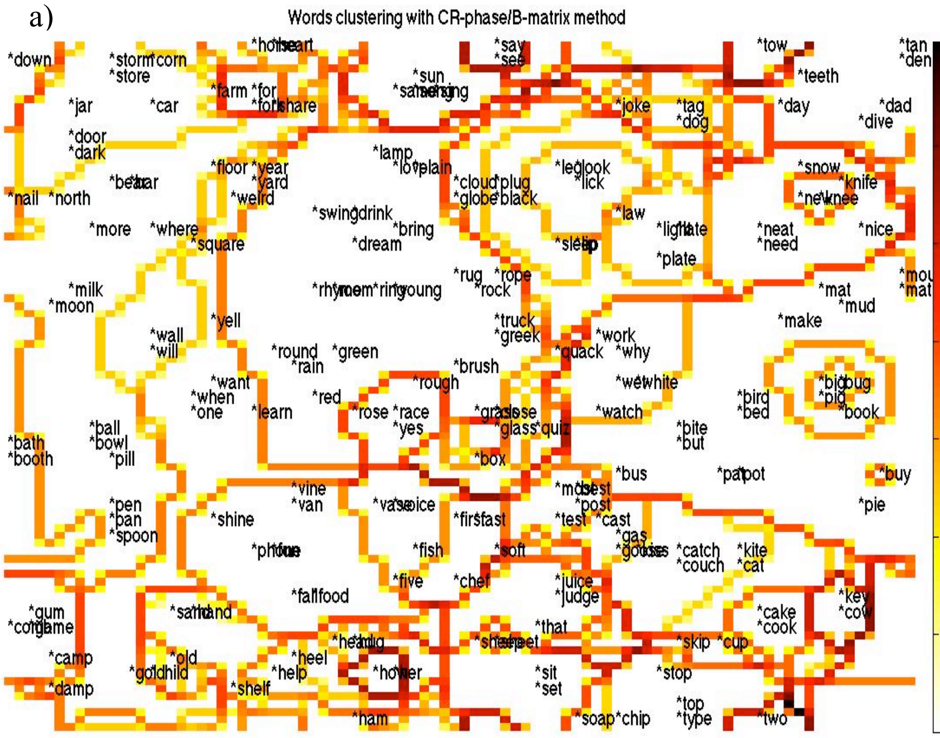
5.2. Results

In this section we show the results of the clustering algorithm for the words phonetic dataset. In Fig 5.1 we show the *B-matrix* of the dataset after training with SOM for 2000 steps and CR phase for 100 iterations. A weight matrix of size 45×45 was used. When using different random seeds (e.g., Fig. 5.1a,b) the words tend to cluster similarly and furthermore the clusters that are close to each other in the first map (Fig. 5.1a) are close to each other in the second map (Fig. 5.1b). This means that the clustering is robust to random initialization, despite reducing the data from 60 dimensions to a 2D map. In Fig. 5.2-5.4 we show a few example component planes of the feature map.

5.3. Discussion

The results of this experiment provide a 2D representation of the 60 dimensional space that can be used to identify which words belong to the same clusters and how close are they in the input space. Audiologists could perhaps use these results to select word pairs to diagnose specific types of DHT auditory processing disorders or answer interesting questions like how close are the clusters that contain the words that a patient failed to recognize, which can then be used to give a failure coefficient. Examining the test words in the component planes, it might be possible to diagnose distinct types of auditory disorders (e.g., if the patient always fails on words that belong to the same or adjacent clusters that differ in specific component planes).

a)



b)

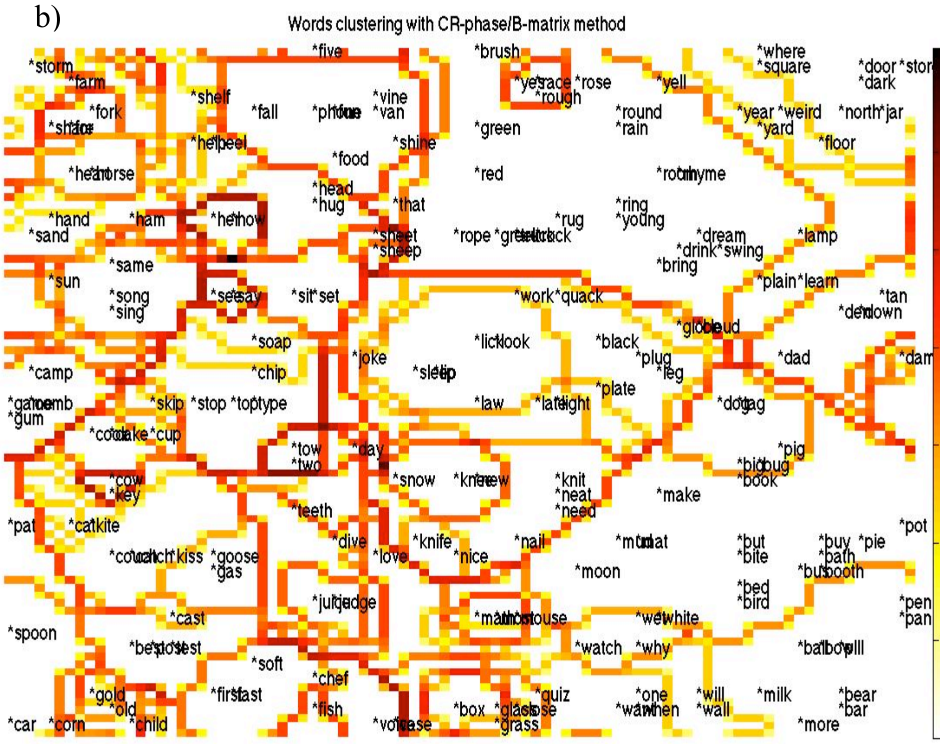
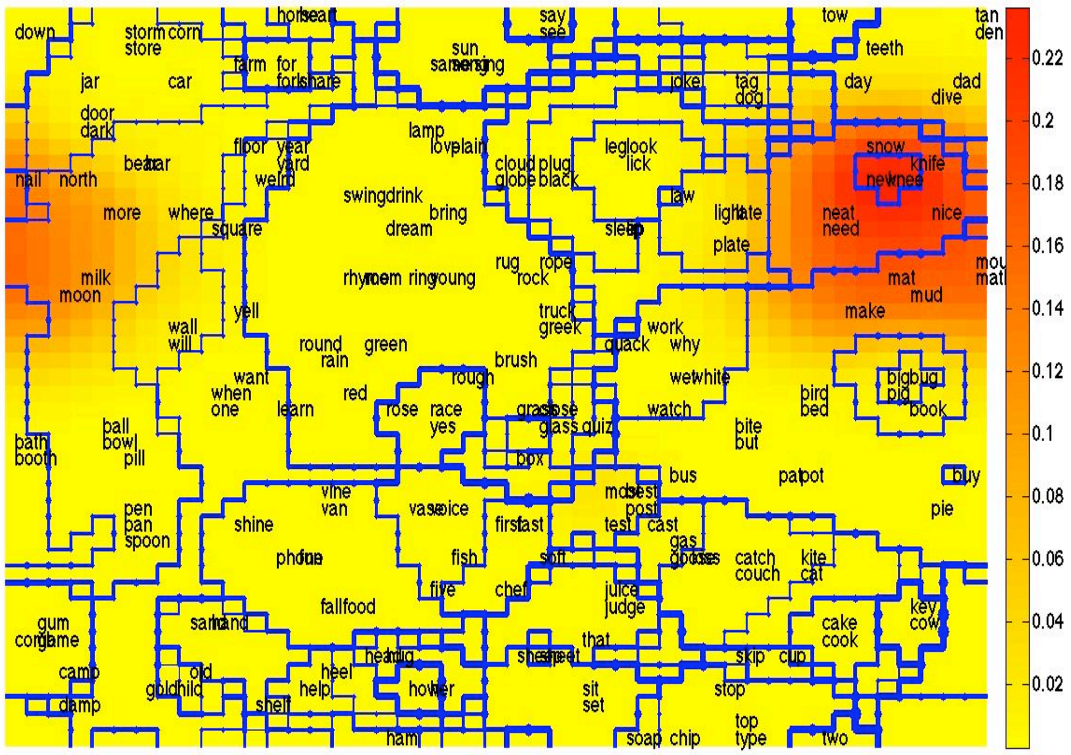


Figure 5.1 The B-matrixes of Word Data for two random seeds, after 2000 iterations of SOM training and 100 iterations of CR training.

Component plane for attribute
C3nasal



Component plane for attribute
C3stop

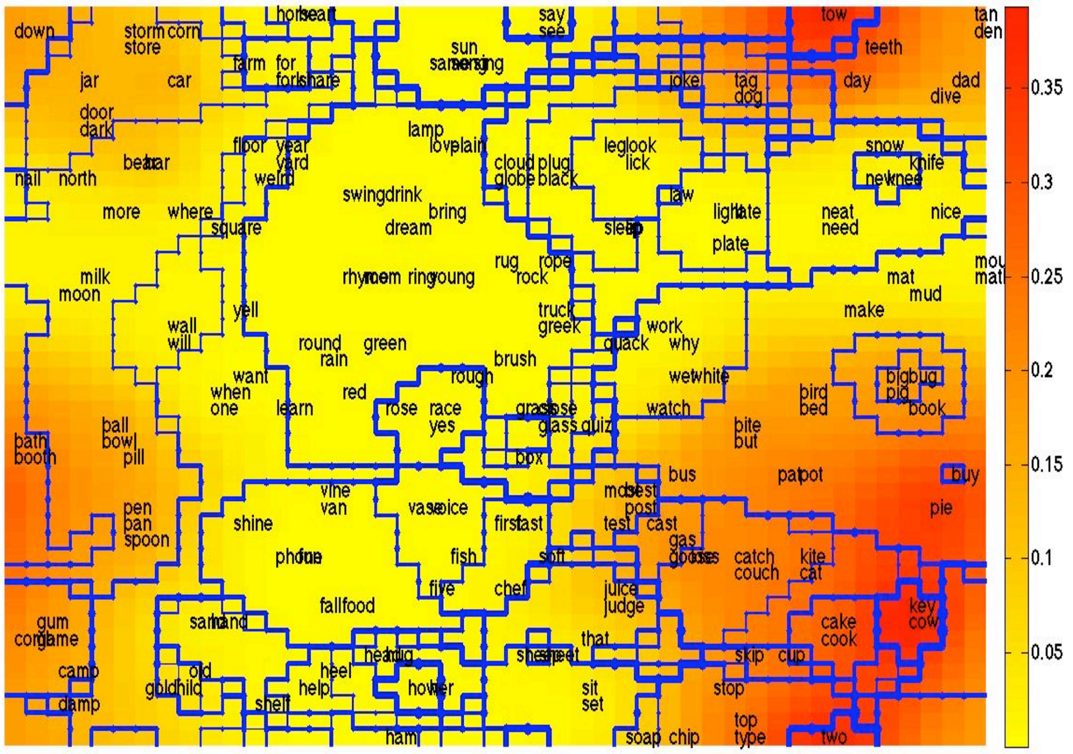


Figure 5.2. Component planes for attributes "C3nasal" and "C3stop".

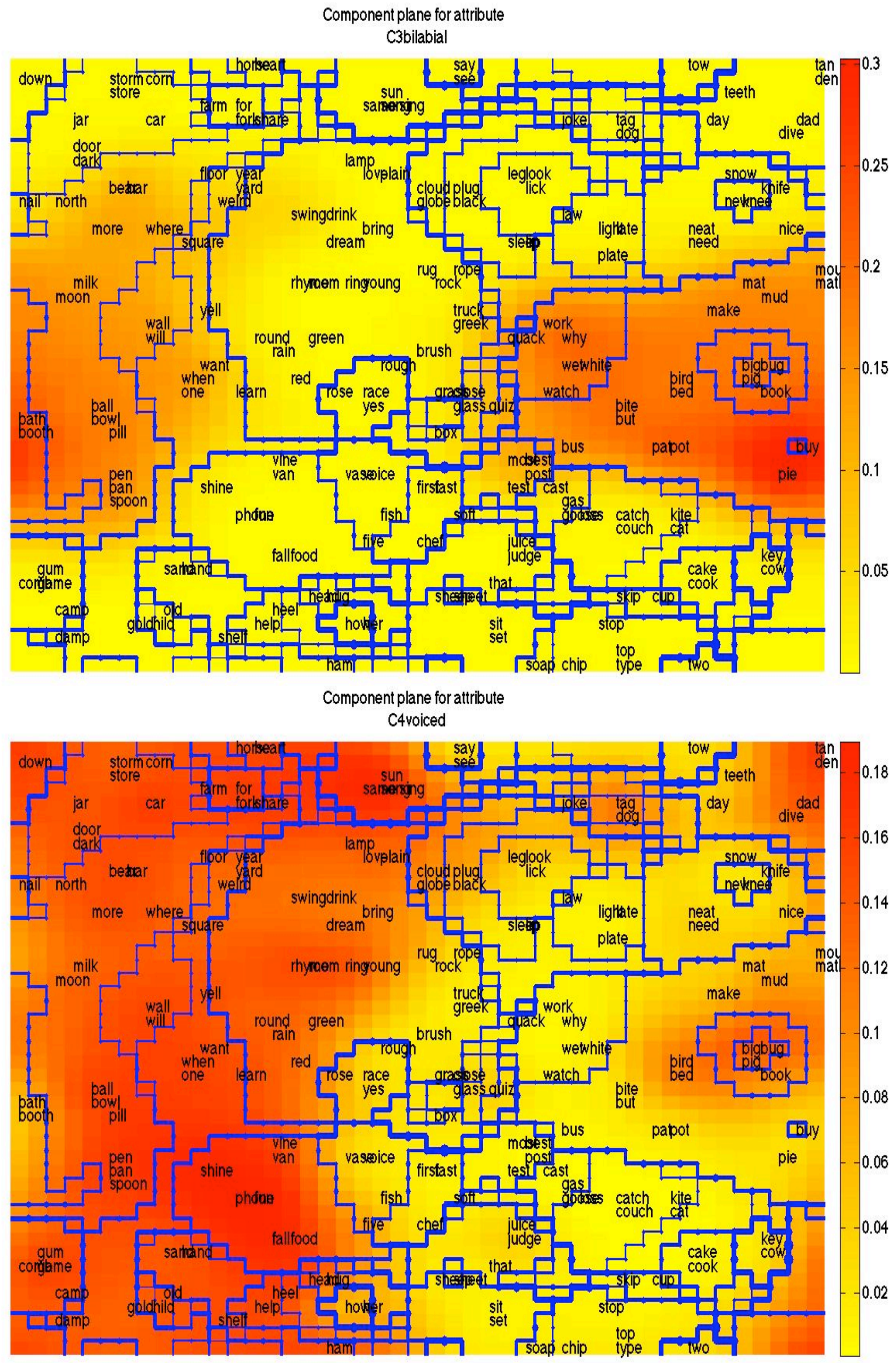
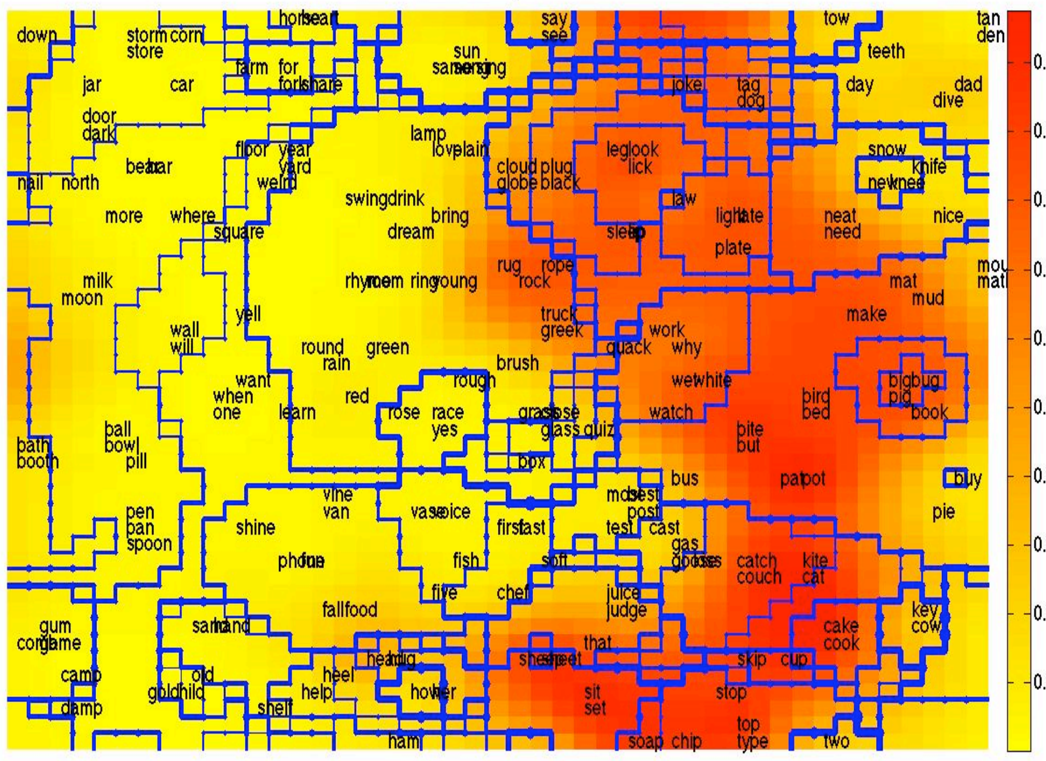


Figure 5.3 Component planes for attributes "C3bilabial" and "C4voiced".

Component plane for attribute
C4stop



Component plane for attribute
C3approximant

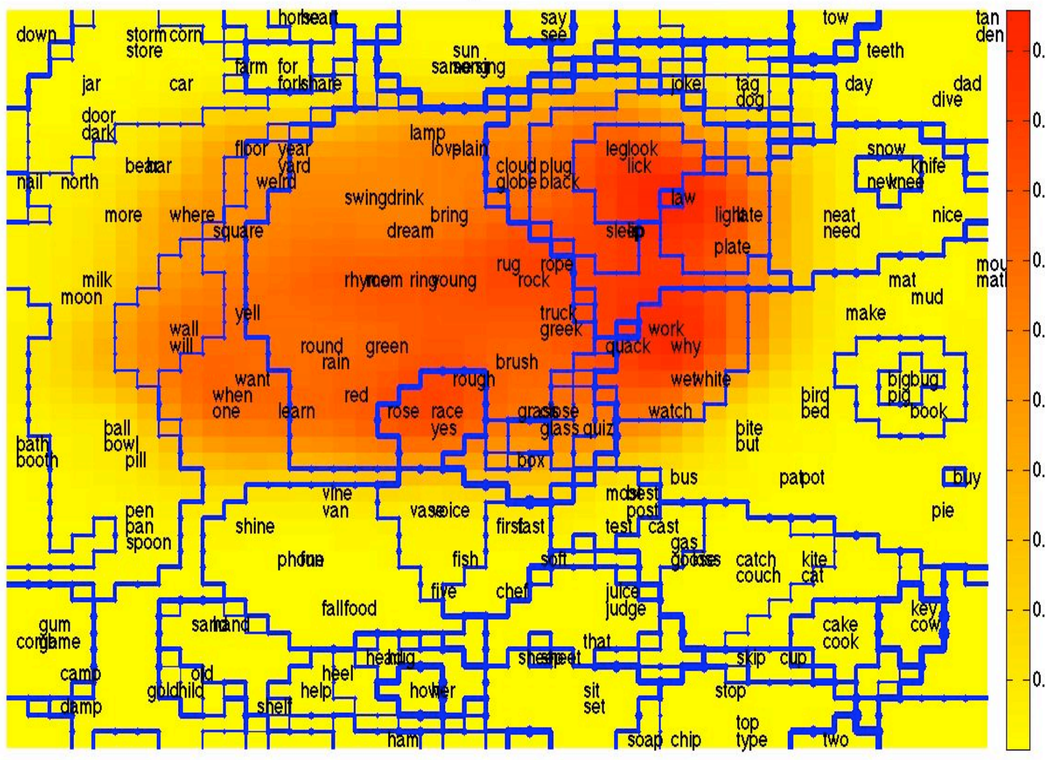


Figure 5.4. Component planes for attributes "C4stop" and "C3approximant".

CHAPTER 6: GEOGRAPHIC DATA CLUSTERING RELATED TO FUEL EFFICIENCY OF VEHICLE CHOICE.

In this chapter, we apply the SOM+CR phase with *B-matrix* visualization to real valued geographic data to find possible nonlinear correlations between various attributes of vehicle efficiency. First, we collected data about registered vehicles locations, vehicle mileage information, and elevation information, and then we used a sliding window approach to find average values of above-mentioned attributes in different locations of the state of Vermont. Finally we applied the SOM+CR phase with *B-matrix* visualization to this averaged data to analyze possible correlations.

6.1. The Dataset

This dataset was collected from several sources (see Fig. 6.1). One of the sources was from the Department of Motor Vehicles (DMV) title database. This database contained the registered address, make, model and year of each registered vehicle titled in the state of Vermont. It was a total of *600,000* records. For this experiment we are only interested in passenger vehicles, so we eliminated the records that represented big trucks, boats, snowmobiles, etc. After this cut we had *410,046* records for passenger vehicles. We decided to eliminate the records where addresses were registered as P.O. Box addresses, since these do not accurately reflect the spatial location of the vehicle registration. Furthermore, we eliminated all the vehicles that were registered in other states. After the above-mentioned eliminations the database contained *321,487* data records, which is 78.5% of all passenger vehicles titled currently in Vermont.

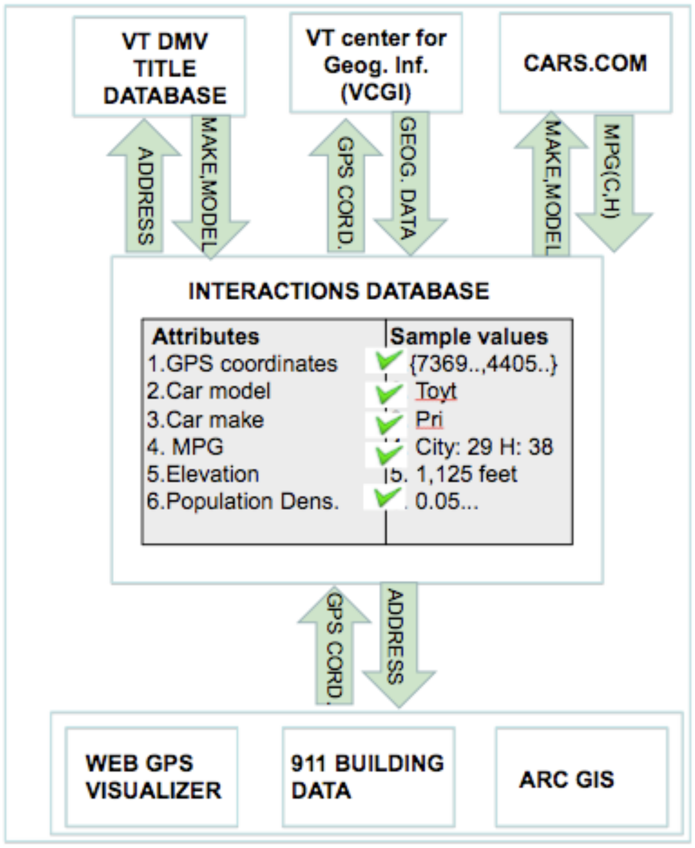


Figure 6.1 This diagram shows all different data sources that were used to collect data.

The next step was using available tools to convert addresses of the form <street, apt, state, zip code> to the form <latitude, longitude>. We used three different sources obtain a complete mapping. First we mapped the addresses from the 911 Building data from the Vermont Center for Geographic Information (VCGI), which contained <latitude, longitude> of most of the buildings in Vermont. As this data set was not complete, we then used a geographical software program called ARC GIS for the addresses that were missing from the first source and finally we used the Web GPS visualizer [34] for the addresses that were missing from both of above-mentioned source (about 100). The latter source was the most complete, because it contained a large database of addresses derived from Google Maps [33], but it took about 50 seconds to retrieve each record. Thus it would have take about 133 hours, which is 14.5 days, if we used it to convert the entire database. In Fig. 6.2 we show the locations of registered lightweight vehicles in the state of Vermont.

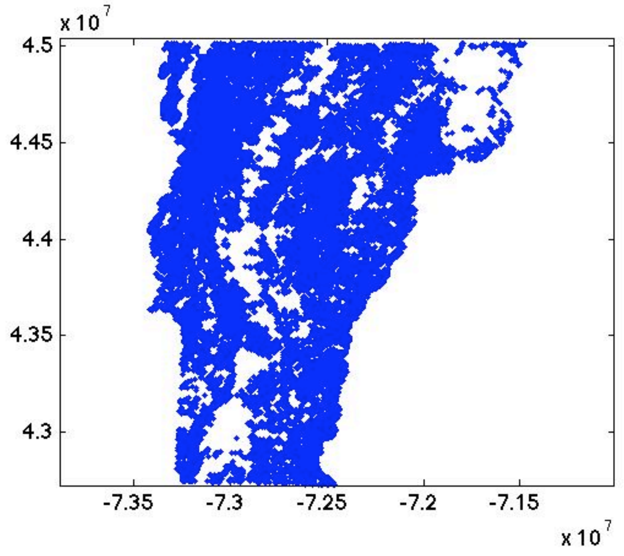


Figure 6.2 Locations of registered lightweight vehicles in the state of Vermont.

To get vehicle mileage information we used the programming language Ruby [29] to mine this information (with permission) from the web site cars.com [7]. This code allowed us to collect city and highway mileage information for each make, model and year of vehicle that exists on this web site, which is one of the largest online sources of vehicle types. One of the obstacles that we had to overcome was the fact that, in the DMV database, the make and model of the cars were reported with short abbreviated and there was no consistency in the abbreviations used. For example the make “Toyota” was reported as “Toyt”, “Toy”, etc. The information that we collected from the web page had the full names of make and model (e.g., Toyota Corolla). We thus had to use some tricks with wild cards and regular expressions to match each of the abbreviated names with full names of make and model. After the last step, we had the latitude and longitude of the locations of passenger vehicles in Vermont and the mileage (MPG) information for them

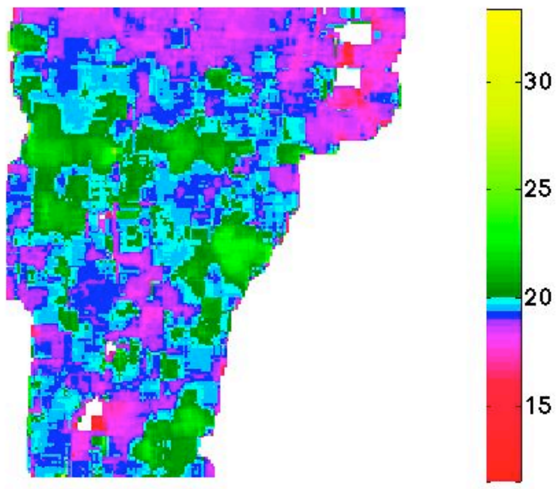


Figure 6.3. Visualization of MPG data for Vermont State.

(see Fig. 6.3). Please note that we adjusted the color scale in Fig. 6.3 to highlight the differences and we used averaging with the spatial window technique (window size = 0.1 decimal latitude/longitude degrees), resulting in 24,311 data points. Having this information we were also able to find the location of each hybrid vehicle in Vermont (see Fig. 6.4).

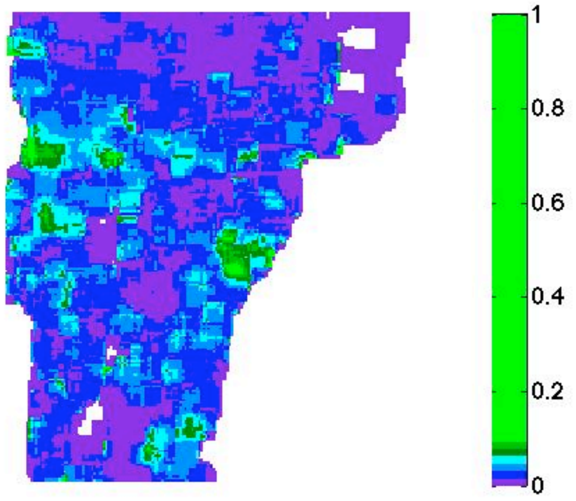


Figure 6.4 Visualization of proportion of hybrids in Vermont.

We also obtained the elevation of each registered vehicle location (Fig. 6.5). For example, we wanted to see if people living in higher elevations tend to have lower mileage cars, due to the fact that lower mileage cars tend to have higher power. This data was obtained through VCGI [35] and the Transportation Research Center at the University of Vermont.

Using the locations of registered vehicles we were also able to determine the density of cars (number of cars per unit area) as shown in Fig. 6.6. In the future we intend to obtain other relevant attributes, such as political make up of various special locations, that we hope may shed some light on consumer vehicle choices.

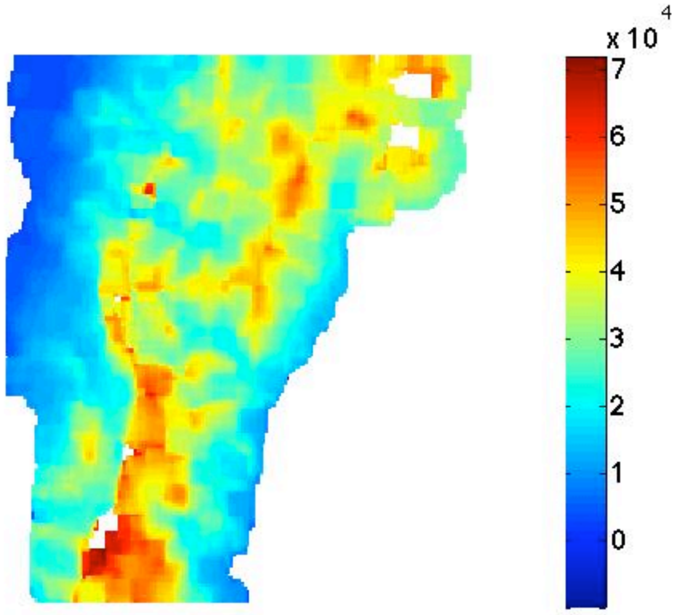


Figure 6.5 Visualization of elevation data for the state of Vermont.

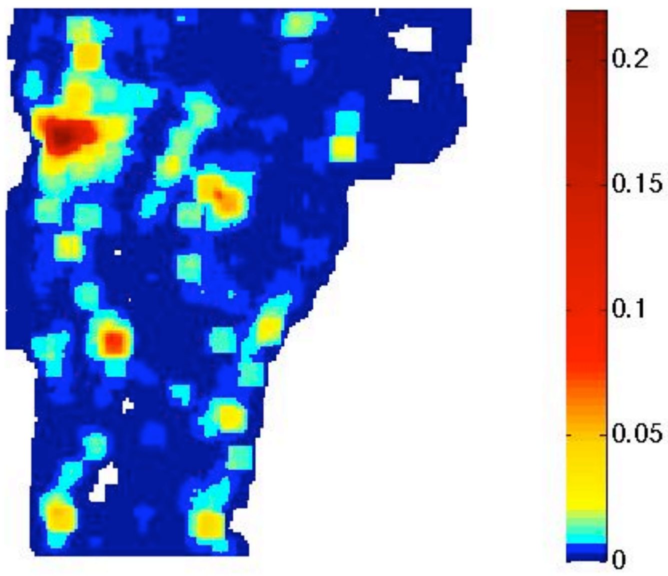


Figure 6.6 Vehicle density map, that shows the number of vehicles per unit area.

6.2. Results

In this section we show preliminary results of the SOM+CR phase with *B-matrix* visualization applied to the real-valued geographical data discussed in Section 6.1. We use three attributes: vehicle density, vehicle city mileage and elevation each normalized to a range from 0 to 1. After training with SOM+CR phase on *24,311* input vectors (obtained using a spatial window of size 0.1 decimal latitude/longitude degrees) the *B-*

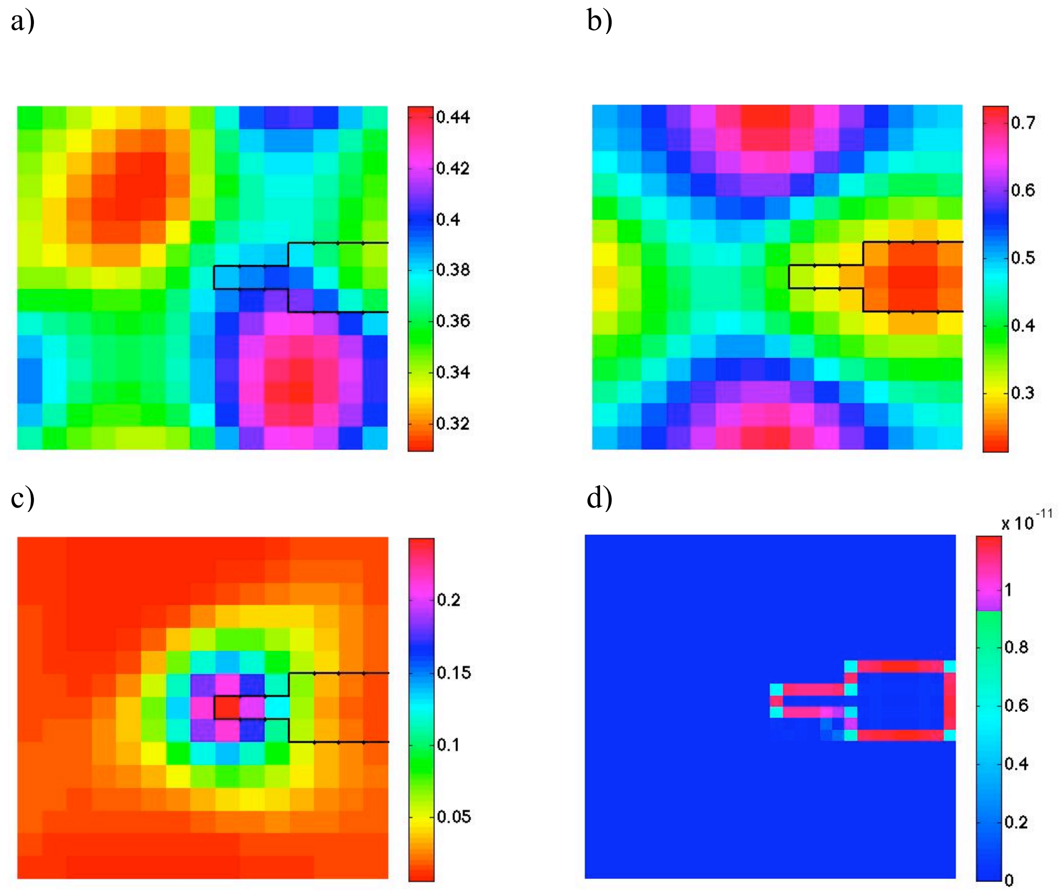


Figure 6.7. a) The component plane of attribute city mileage, b) the component plane of attribute elevation, c) the component plane of attribute vehicle density, d) the B-matrix of geographical data. In figures a, b, c the threshold is 0.9*10^-11.

matrix of the feature map is the 2D representation of these three. For this experiment we used a gird of 15×15 for weight vectors. The SOM was trained for 2000 steps, the CR phase was trained for 10 iterations, using a random subset of 200 input patters for each CR step. In Fig. 6.7 we show the component planes and *B-matrix* of the geographical data.

The results show that there are essentially no correlations between the attributes of this preliminary data set. The one cluster that was identified is essentially delineating low elevation portion of the data. This occurred because after SOM training the ranges of normalized mileage and density were reduced to much smaller than that of normalized elevation, due to the non-normal left-skewed distribution of the raw data for mileage and density. We hope to obtain additional features that may relate to consumer vehicle efficiency choice in the future. Nevertheless, this example demonstrates that the SOM+CR and *B-matrix* visualization works for attributes of real-valued numbers.

CHAPTER 7: CONCLUSION

7.1. Overview

Self-organizing maps have been proven to be useful tools for clustering and visualization of high-dimensional data. Nevertheless it is often challenging to interpret the results due to drawbacks of currently used methods for identifying cluster boundaries in the resulting feature maps. In this thesis, we introduced an additional CR phase to the SOM algorithm that results in sharper boundaries between the clusters. We also introduce a new form of distance matrix (*B-matrix*) that facilitates the visualization of cluster boundaries. By $2n \times 2n$ display of the B-values as the thickness of grid lines they can be overlain onto $n \times n$ component planes. A threshold parameter θ is used to threshold the displayed lines to allow hierarchical control of the visual level of cluster resolution. These advanced visualization methods were shown to perform better then regular SOM/*U-matrix* method for a benchmark animal clustering problem. We then showed real world applications of these methods for a 60-dimensional binary-valued phonetic word clustering problem and a 3-dimensional real-valued geographic data clustering problem related to fuel efficiency of vehicle choice. The word clustering application may help to design and interpret dichotic auditory processing tests. While no clustering was observed in the geographical data related to fuel efficiency, we anticipate that adding additional features may yield more interesting results.

References

1. Bação, F., V. Lobo, and M. Painho, *Geo-self-organizing map (Geo-SOM) for building and exploring homogeneous regions.* Geographic Information Science, 2004: p. 22-37.
2. Benaïm, M., J.C. Fort, and G. Pagès, *Convergence of the one-dimensional Kohonen algorithm.* Advances in Applied Probability, 1998: p. 850-869.
3. Besaw, L.E., et al., *Stream classification using hierarchical artificial neural networks: A fluvial hazard management tool.* Journal of Hydrology, 2009. **373**(1-2): p. 34-43.
4. Bouton, C. and G. Pagès, *Convergence in distribution of the one-dimensional Kohonen algorithms when the stimuli are not uniform.* Advances in Applied Probability, 1994. **26**(1): p. 80-103.
5. Chang, F.J., L.C. Chang, and G.R. Kaoand, *Assessing the effort of meteorological variables for evaporation estimation by self-organizing map neural network.* Journal of Hydrology, 2010. **384**(1-2): p. 118-129.
6. Choe, Y. and R. Miikkulainen, *Contour integration and segmentation in a self-organizing map of spiking neurons.* Biological Cybernetics, 2004. **90**(2): p. 75-88.
7. Classified Ventures, L., *cars.com.* June 1998.
8. Cottrell, M. and J.C. Fort, *Étude d'un processus d'auto-organisation.* Ann. Inst. Henri Poincar´e, 1987. **23**(1): p. 1-20.
9. Cottrell, M., J.C. Fort, and G. Pagès, *Theoretical aspects of the SOM algorithm.* Neurocomputing, 1998. **21**(1-3): p. 119-138.
10. Erwin, E., K. Obermayer, and K. Schulten, *Self-organizing maps: ordering, convergence properties and energy functions.* Biological Cybernetics, 1992. **67**(1): p. 47-55.
11. Erwin, E., K. Obermayer, and K. Schulten, *Self-organizing maps: Stationary states, metastability and convergence rate.* Biological Cybernetics, 1992. **67**(1): p. 35-45.
12. Flanagan, J.A., *Self-organized criticality and the self-organizing map.* Physical Review E, 2001. **63**(3): p. 36130.
13. Flanagan, J.A., *Sufficient conditions for self-organisation in the one-dimensional SOM with a reduced width neighbourhood.* Neurocomputing, 1998. **21**(1-3): p. 51-60.
14. Fort, J.C. and G. Pages, *On the as convergence of the Kohonen algorithm with a general neighborhood function.* The Annals of Applied Probability, 1995. **5**(4): p. 1177-1216.
15. Fort, J.C. and G. Pagès, *About the convergence of the generalized Kohonen algorithm.* 1994.
16. Honkela, T., *On the Use of Self-Organizing Map (SOM) in Linguistic Visualization*, 1996.
17. Kelley, K., *Phonological characteristics of words*, 2011: University of Vermont.

18. Kiviluoto, K., *Predicting bankruptcies with the self-organizing map.* Neurocomputing, 1998. **21**(1-3): p. 191-201.
19. Kohonen, T., *Analysis of a simple self-organizing process.* Biological Cybernetics, 1982. **44**(2): p. 135-140.
20. Kohonen, T. *Automatic Formation of TopologicalMaps of Patterns in a Self-Organizing System*. in *Proc. 2SCIA, Scand. Conf. on Image Analysis*. 1981.
21. Kohonen, T., *Self-Organization and Associative Memory. Springer-Verlag.* New York, 1984.
22. Kohonen, T., *Self-organized formation of topologically correct feature maps.* Biological Cybernetics, 1982. **43**(1): p. 59-69.
23. Kohonen, T., *The Self-Organizing Map.* Proceedings of the IEEE, 1990. **78**: p. 1464-1480.
24. Kohonen, T., *Self-Organizing Maps: Optimization Approaches In: Artificial Neural Networks, T.* Elsevier Science (North-Holland), 1991.
25. Kraaijveld, M.A., J. Mao, and A. Jain, *A nonlinear projection method based on Kohonen's topology preserving maps.* Neural Networks, IEEE Transactions on, 2002. **6**(3): p. 548-559.
26. Laaksonen, J., et al., *PicSOM-content-based image retrieval with self-organizing maps.* Pattern Recognition Letters, 2000. **21**(13-14): p. 1199-1207.
27. Ladefoged, P. and K. Johnson, *A course in phonetics*. Vol. 127. 1975: Harcourt Brace Jovanovich New York.
28. M. Cottrell, J.C.F., G. Pagès, *Theoretical aspects of the SOM algorithm.* Neurocomputing, 1998. **21**: p. 119-138.
29. Matsumoto, Y.M. *Ruby programming language*. mid-1990s.
30. Merkl, D. and A. Rauber, *Alternative ways for cluster visualization in self-organizing maps*, in *In Proc. of the Workshop on Self-Organizing Maps (WSOM97*1997.
31. Nikkilä, J., et al., *Analysis and visualization of gene expression data using self-organizing maps.* Neural Networks, 2002. **15**(8-9): p. 953-966.
32. Obermayer, K., H. Ritter, and K. Schulten, *A principle for the formation of the spatial structure of cortical feature maps.* Proceedings of the National Academy of Sciences of the United States of America, 1990. **87**(21): p. 8345-8349.
33. Online Source, w.p. Http://maps.google.com/maps. 2005.
34. Online Source, w.p. Http://www.gpsvisualizer.com/. 2002.
35. Online Source, w.p., Http://www.vcgi.org/. 1994.
36. Pearce, A.R., P.J. Mouser, D.M. Rizzo, *Characterizing a landfill leachate contamination plume by clustering microbial community profile data.* Water Resources Research, 2011.
37. Peura, M. and J. Iivarinen, *Efficiency of simple shape descriptors.* Advances in Visual Form Analysis, World Scientific, Singapore, 1997: p. 443–451.
38. Ritter, H. and T. Kohonen, *Self-organizing semantic maps.* Biological Cybernetics, 1989. **61**(4): p. 241-254.
39. Ritter, H. and T. Kohonen, *Self-organizing semantic maps.* Biological Cybernetics, 1989. **61**: p. 241-254.

40. Rui, Y., T.S. Huang, and S.F. Chang, *Image Retrieval: Current Techniques, Promising Directions, and Open Issues* 1.* Journal of visual communication and image representation, 1999. **10**(1): p. 39-62.
41. Russo, T., et al., *Application of the Self-Organizing Map to the study of skeletal anomalies in aquaculture: The case of dusky grouper (Epinephelus marginatus Lowe, 1834) juveniles reared under different rearing conditions.* Aquaculture, 2010.
42. Sarkaria, S.S., A.J. Harget, and E. Claridge, *Shape recognition using the Kohonen self-organising feature map.* Pattern Recognition Letters, 1992. **13**(3): p. 189-194.
43. Scholtes, J.C., *Neural Networks in Natural Language Processing and INformation Retrieval*, 1993, Universiteit van Amsterdam: the Netherlands.
44. Scholtes, J.C., *Unsupervised learning and the information retrieval problem*, in *IEEE International Joint Conference on Neural Networks*1991, Int. Neural Networks Soc, IEEE: New York, NY, USA. p. 95-100.
45. Simon, H., *Neural networks: a comprehensive foundation.* 1999.
46. Törönen, P., et al., *Analysis of gene expression data using self-organizing maps.* FEBS letters, 1999. **451**(2): p. 142-146.
47. Ultsch, A. *Self organized feature maps for monitoring and knowledge acquisition of a chemical process*. 1993.
48. Ultsch, A. *Self-Organizing Neural Networks for Visualization and Classification*. in *Information and Classification*. 1993. Springer.
49. Ultsch, A., *U*-matrix: a tool to visualize clusters in high dimensional data*2003: Fachbereich Mathematik und Informatik.
50. Ultsch, A. and H.P. Siemon. *Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis*. in *Proceedings of International Neural Networks Conference (INNC)*. 1990. Kluwer Academic Press.
51. Wiggins, J.L., et al., *Using a self-organizing map algorithm to detect age-related changes in functional connectivity during rest in autism spectrum disorders.* Brain Research, 2010.
52. Yin, H. and N.M. Allinson, *On the distribution and convergence of feature space in self-organizing maps.* Neural Computation, 1995. **7**(6): p. 1178-1187.