A CONVERGENCE CRITERION FOR SELF-ORGANIZING MAPS

BY

BENJAMIN H. OTT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

UNIVERSITY OF RHODE ISLAND

2012

MASTER OF SCIENCE THESIS

OF

BENJAMIN H. OTT

APPROVED:

Thesis Committee:

Major Professor Lutz Hamel

Joan Peckham

Nancy Eaton

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2012

ABSTRACT

Currently, only computationally complex, probabilistic models for convergence exist for self-organizing maps (SOMs). The complexity of these algorithms seems to take away from the simple intuitive nature of SOMs in addition to making them computationally unwieldy. The hypothesis proposed is that the basic SOM algorithm is heuristic in nature and will almost always converge on a set of code vectors (or neurons) which reflect the underlying distribution from which the original sample (or training) vectors were drawn. This study shows that this hypothesis is valid for the basic SOM algorithm by imposing a convergence criterion on the basic SOM algorithm. The convergence criterion (or convergence measure) imposed treats the SOM as a conventional two sample test (i.e. one sample being the training data and one sample being the code vectors). If the hypothesis holds, then imposing a population based convergence criterion on SOMs will increase their accuracy and utility by allowing an end user of the maps to differentiate between "good" maps (which converged well) and "bad" maps (which did not converge well), hence allowing only the best maps to be selected for use. The convergence criterion could also be used as an indicator for the appropriate number of training iterations necessary for a given set of training data. For instance, if a convergence check were to show that a given SOM has not yet converged, then the number of training iterations could be increased in the next SOM construction. Alternatively, the learning rate could be increased in order to increase the SOM's ability to capture the variance of the training vectors.

In addition to testing this hypothesis via the calculation of the proposed convergence measure (using the two sample, population based, approach), another existing method of testing SOMs for reliability and organization (i.e. convergence or "goodness") is calculated and compared to the results of the convergence mea-

sure proposed in this study. What has been observed is that the SOM does indeed converge and that the convergence measure proposed in this thesis is more conservative than the existing, and much more computationally intensive, algorithm to which it is compared. Hence, when the convergence criterion proposed herein is satisfied, the reliability and organization criterion are implied. In addition to being more conservative, it also correlates better with lower quantization errors.

ACKNOWLEDGMENTS

As it turns out, a journey such as the writing of a thesis requires a bit of support, and different kinds of support at that. There were some individuals whose support and encouragement made the work within the pages that follow possible. To these people, I am eternally grateful and I consider myself fortunate to know them and to have earned their support.

First and foremost, I would like to thank my loving wife, Rebecca, for her support of my research and for the saint like patience she exhibited those days and nights that I spent researching and writing the contents of this thesis. Secondly, I would like to thank my thesis advisor, Dr. Hamel, for allowing me to be a part of this exceptional project, for challenging me, and for offering sage advice and insightful guidance to me throughout the course of my research. I would also like to thank the members of my research committee and my examining committee, Dr. Peckham, Dr. Eaton and Dr. Dash, for reading and commenting on my thesis and for being a part of my educational journey. Lastly, I would like to thank my family and friends for their encouragement and for providing such a stalwart support network. I would like to distinctly thank my parents for their words of encouragement and for the wonderful example of perseverance that they set for me. I would also like to thank my father-in-law and my mother-in-law for setting great examples and encouraging me throughout my studies.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure **Page**

Figure **Page**

CHAPTER 1

Introduction

Self-organizing maps (SOMs) are a common type of artificial neural network in which training data is run iteratively through a training algorithm. The algorithm matches a given training vector to its optimal fit (best matching unit) in the SOM (see Figure 1) and then acts on the optimal fit and its neighbors using a neighborhood function in order to preserve the general topology, or structure, of the input space (training data). SOMs are used extensively as a method of analysis in a broad variety of fields including bioinformatics, financial analysis, signal processing, and experimental physics as they provide a simple yet effective algorithm for clustering via unsupervised learning [2]. The simple nature of the SOM algorithm and the way in which the visualization of the SOM can be easily and intuitively interpreted make it appealing as an analysis tool. However, with any analysis tool, and especially iterative learning-based tools, questions pertaining to the reliability and the convergence of the tool naturally emerge.

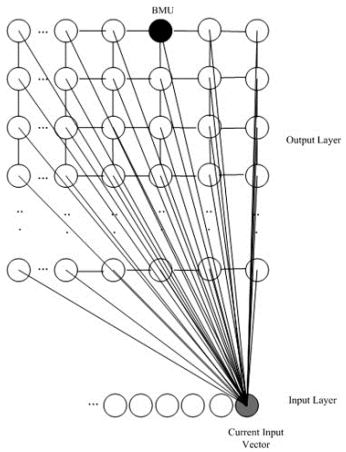

Figure 1: The optimal fit, or best matching unit (BMU) in the SOM, is the code vector which is most like the given input vector (i.e. which has the smallest Euclidean distance to the input vector). The above pictorial was given in [1].

Several effective measures have been developed in order to analyze the effectiveness of a given SOM. One such measure, quantization error, is the error function proposed by Kohonen and is the de facto measure of the efficacy of a given SOM at modeling the characteristics intrinsic to the sample data used for training[2]. The quantization error of a given training vector is the smallest distance between that training vector and any neuron in the SOM. To get a feel for how well a given SOM represents a data set, one can sum the quantization error over all of the training data vectors or one can calculate the average quantization error of the training data vectors. The goal of the SOM then, is to minimize this value. Attempting to minimize the quantization error can lead to overfitted models which are ineffective at classifying future samples because as the model fits the training data better and better, it may end up modeling noise in the training data which is not characteristic of the general population from which the training data sample was drawn. Also, how does one determine when the quantization error is "good enough"? One can make the quantization error arbitrarily small by increasing the complexity of the model by adding neurons to the map.

Another approach is to modify the algorithm itself so that statistical measures or other objective analysis techniques can be imposed. Bishop's generative topographic mapping (GTM) [3] and Verbeek's generative self-organizing map (GSOM) [4] seem to fall into this category. The GTM and GSOM attempt to model the probability density of a data set using a smaller number of latent variables (i.e. the dimensionality of the latent space is less than or equal to that of the data space). A non-linear mapping is then generated which maps the latent space into the data space. The parameters of this mapping are learned using an expectation-maximization (EM) algorithm [3, 5]. Algorithms in the class of the GTM and the GSOM should be viewed as alternates to the SOM as opposed to modifications of

it, even though they share properties similar to the SOM. Other scholars have taken an energy function approach, imposing energy functions on the neurons of the SOM and then attempting to minimize these energy functions [6, 7], again seeming to take away from the SOM's simplicity. Both of these approaches, namely altering the algorithm or imposing energy functions on the SOM, seem to take away from the SOM's appeal as a simple, fast algorithm for visualizing high dimensional data, especially since the alterations tend to be much more complicated than the SOM learning algorithm itself.

Yet another approach is to calculate the significance of neighborhood stabilities in order to analyze whether or not inputs close in the sample space remain close when projected on the SOM. In addition, checks are performed in order to determine whether or not the neighborhood relationships are preserved from map to map [i.e. if many SOMs are created using bootstrap samples of the training data, and the training data is projected onto each of the SOMs, do the training vectors appear to be randomly mapped within a defined neighborhood or is their neighborhood relation significant][8]. While providing a sound set of statistical tools to analyze SOMs, this reliability and stability based approach, is computationally unwieldy and drastically increases the amount of time associated with the analysis of a given data set. The time cost comes from the creation of many maps using the bootstrapped samples of training data (typically 100-200 maps; Efron recommends using at least 200 samples when bootstrapping statistics [9]) and the analysis of each pair of training data over each map after all of the maps have been trained.

Here, we propose a population based approach for analyzing SOM convergence. Yin and Allison [10] showed that the neurons in a SOM will converge on the probability distribution of the training data in the limiting case (i.e. if the

SOM were trained forever). They imposed the assumption that at each iteration $t = 1, 2, ..., \infty$, there were no "dead" neurons in the map. In other words, each neuron was assumed to be updated an infinite number of times. However, the key insight is that the SOM, as Kohonen had claimed earlier, will in effect, model the probability density of the training data in a fairly general setting. If we operate under the assumption that the SOM will model the probability density of the training data, then we can perform a simple two sample test in order to see if the SOM has effectively modeled the training data. In other words, we can treat the neurons of the map as one sample, treat the training data as another sample, and then proceed to perform tests in order to see if it appears that the samples were drawn from the same probability space. Thus, the SOM may be viewed as a sample building algorithm. This population based approach lends to a fast convergence criterion, based on standard statistical methods, which does not modify the original algorithm, hence preserving its appeal as a simple and fast analysis tool.

In this thesis, we first discuss some background material which is fundamental to the convergence measure proposed here. Secondly, we introduce a conventional convergence approach which examines the stability of neighborhood relationships on the SOM. This stability based approach is the one proposed by Cottrell et al. This approach is based on measures imposed on the map, which is typical for the currently existing convergence criteria. Measures are imposed solely on the map without comparing the neurons of the map to the training data in order to see if the neurons adequately model the training data. Third, we discuss the convergence measure proposed in this thesis. This measure uses conventional statistical measures to compare the sample formed by the neurons of the SOM to the sample formed by the training data in order to determine if it appears that the neurons

adequately model the training data (i.e. checks are performed in order to see if it looks like the two samples are drawn from the same underlying probability space). Each feature is checked and credit is given for each feature which is adequately modeled by the SOM. Finally, results from experimentation comparing the conventional measure to the convergence measure proposed in this body of work are reported.

CHAPTER 2

Background

A basic understanding of Self-Organizing Maps (SOMs), the Bootstrap, and Principal Components are necessary in order to provide the framework for the convergence criterion later proposed in this work.

2.1 Self-Organizing Maps

Self-organizing maps are a common type of artificial neural network in which training data is run repetitively through a training algorithm which matches each training vector to its optimal fit in the SOM and then acts on the optimal fit and its neighbors using a neighborhood function in order to preserve the general topology, or structure, of the input space (training data). The code vectors, or neurons in the network, can be initialized in any way and the SOM should ultimately converge [10]. The basic SOM algorithm uses an optimal fit approach based on the Euclidean distance to find the code vector upon which a given training vector should act. Then, the optimal fit code vector and any code vectors within its neighborhood are altered slightly using a scale factor (the learning rate - included in the neighborhood function) times the difference between the given code vector and the training vector. SOMs provide a simple yet effective algorithm for clustering via unsupervised learning. As the name might indicate, clustering algorithms are a method of unsupervised learning which have the goal of grouping similar data together (i.e. data which are very similar should be grouped, or clustered, together). Unsupervised learning essentially means that the algorithm is not given any indicator regarding the group or cluster with which each training vector is associated. In the case of the SOM algorithm, after the algorithm has been run, each training vector is typically mapped onto the SOM and the associated neuron

is labeled with an ID indicating the cluster to which the vector belonged.

The SOM algorithm may be mathematically defined as follows. Let x denote a training vector. The optimal fit, or best matching, code vector can be found using the following simple test [2]:

$$c = \arg\min_i ||x - m_i|| \tag{1}$$

or, equivalently,

$$||x - m_c|| = \min_i ||x - m_i|| \tag{2}$$

where the m_i represent the neurons of the SOM. Next, the code vectors are updated according to the following equation [2]:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{3}$$

where t is an integer specifying the iteration (i.e. t=0,1,2,...), $m_i(t+1)$ is the updated code vector, $m_i(t)$ is the code vector prior to the update, $x(t)$ is a randomly chosen training vector, and $h_{ci}(t)$ is a neighborhood function. In the context of this work, the neighborhood function $h_{ci}(t)$ is given by the following equation:

$$h_{ci}(t) = \begin{cases} \alpha(t) & \text{if } m_i \text{ is in the neighborhood, } N_c(t), \text{ of } m_c \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\alpha(t)$ is a monotonically decreasing function of time t and $N_c(t)$ also decreases with time. What this means is that code vectors in the neighborhood of m_c, the optimal fit code vector, will be updated to be a multiple of $\alpha(t)$ closer to the training vector x. The neighborhood $N_c(t)$ should be very wide in the beginning of training and shrink monotonically with time until the optimal fit neuron is the

only neuron in neighborhood defined by $N_c(t)$ [10]. A good global ordering may then be formed [10].

The algorithm is run for many iterations while $\alpha(t)$ and $N_c(t)$ decrease in order to ensure a reasonable organization of the map. Typically, the number of iterations, t, has to be reasonably large in order to have confidence that the resulting SOM is reliably organized (i.e. the training data needs to be run through the algorithm a substantial number of times). Depending on the complexity of the training data sample, t may need to be in the 100,000s or greater in order to have confidence in the resulting maps.

2.2 Bootstrap

The basic bootstrap provides a non-parametric method to measure a statistic and its associated standard error. Suppose that there is a set of data $X = (x_1, x_2, \cdots, x_n)$, where each x_i represents a random vector drawn from a population. The basic bootstrap is performed by creating some number of bootstrap samples from the sample X, calculating the statistic of interest for each bootstrap sample (creating bootstrap replications of the statistic), and then calculating the mean and the standard error of the bootstrap replications. The mean and standard error of the bootstrap replications will serve as estimates for the statistic itself and the standard error of the statistic respectively.

A bootstrap sample, say X^*, is created by randomly sampling n times with replacement from the sample X [9]. For example, if X is a sample consisting of five vectors, $(x_1, x_2, x_3, x_4, x_5)$, a bootstrap sample might look like the following: $X^* = (x_2, x_5, x_1, x_3, x_2)$. In order to bootstrap a statistic, B bootstrap samples are created from the sample X. The bootstrap samples would be indexed as $X^{*1}, X^{*2}, \cdots, X^{*B}$. For each bootstrapped sample, the statistic of interest, say $s(x)$ is calculated, resulting in B bootstrap replications of the statistic of interest,

$s(X^{*1}), s(X^{*2}), \cdots, s(X^{*B})$. The bootstrap estimate of the statistic, $s(\cdot)$ would then be given by the following equation [9]:

$$s(\cdot) = \frac{\sum_{b=1}^{B} s(X^{*b})}{B} \tag{5}$$

Put another way, the bootstrap estimate of the statistic is the average value of the bootstrap replications of the statistic. Similarly, the bootstrap estimate of the standard error is given by the standard deviation of the bootstrap replications [9]. The equation for the bootstrap estimate of standard error is as follows:

$$\widehat{se}_{boot} = \left\{ \frac{\sum_{b=1}^{B} [s(X^{*b}) - s(\cdot)]^2}{B-1} \right\}^{\frac{1}{2}} \tag{6}$$

Efron and Tibshirani have shown that if one takes the limit of these equations for the sample mean (i.e. where $s(\cdot)$ is the sample mean) as B tends towards infinity, the equations approach the conventional definitions for the mean and standard error of the sample [9]. The bootstrap extends to more general statistics as well. In fact, the mean is generally not bootstrapped because standard statistical formulas enable one to efficiently and quickly calculate the mean and standard error (or standard deviation) of a sample. One example where the bootstrap is effective and useful is in the calculation of the standard error of a sample median. The bootstrap samples and their associated bootstrap replications lend to a nonparametric method for calculating an estimate of this statistic. The term nonparametric is used to describe a statistical method for which knowledge about the underlying distribution of the statistic being estimated is unknown (or simply not used). The bootstrap as described in this section is sometimes referred to as the nonparametric bootstrap. For most statistics, the limiting value of the standard error is not known. However, the bootstrap still provides a good estimator for the standard error of these statistics [9].

2.3 Principal Components

Principal components are based on a given distribution's covariance matrix. If there is a random sample from a population X, consisting of random vectors, a covariance matrix, C, is a matrix whose entries are given by the following equation[11]:

$$C_{a,b} = \sigma_{x_a x_b} = E[(X_a - \mu_a)(X_b - \mu_b)] = \frac{1}{N}\sum_{i=1}^{N}(x_{ai} - \mu_a)(x_{bi} - \mu_b) \qquad (7)$$

where x_{ai} denotes the a^{th} dimension in the i^{th} sample vector, μ_a denotes the mean of the sample in the a^{th} dimension, and $E[...]$ is the standard notation for the expected value. The expected value presented in Equation (7), for a randomly chosen vector X, is the difference of X in dimension a to the mean of the population in dimension a times the difference of X in dimension b to the mean of the population in dimension b. In this case, there would be N sample vectors. Each entry of the covariance matrix should be read as the covariance between the a^{th} and b^{th} features (or dimensions) of the sample. In the case where a is equal to b (i.e. for the diagonal entries in the covariance matrix), the above equation becomes, $\frac{1}{N}\sum_{i=1}^{N}(x_{ai} - \mu_a)^2$, and each diagonal entry, as given by this equation, would be referred to as the variance in the a^{th} dimension.

The covariance matrix essentially provides a measure for how the different features of a potentially multi-dimensional sample are related to each other. A positive covariance $C_{a,b}$ indicates that when x_a increases, x_b also increases. In other words, they are directly related. On the other hand, a negative covariance indicates that when x_a increases, x_b decreases (i.e. they are inversely related). The variance terms $C_{a,a}$ in the covariance matrix are always positive and provide a measure of the spread of the distribution in the direction of a.

The covariance matrix is a real symmetric matrix. This means that $C_{a,b} =$

$C_{b,a}$ (as can easily be observed based on the formula used to compute the entries of the covariance matrix) and that the entries in the matrix are real numbers. Real symmetric matrices have one especially nice property, namely that they are always diagonalizable. This is given by the spectral theorem [12] as applied to real symmetric matrices. For a real symmetric matrix C, being diagonalizable means that there exists an orthogonal matrix U such that the following holds true: $D = U^T C U$, where U is a matrix which has the eigenvectors of C as its columns and D is a diagonal matrix (i.e. all entries not along the diagonal are zero) which has the eigenvalues of C along its diagonal. Note that U^T is a matrix which has the eigenvectors of C as its rows. One significant property of orthogonal matrices is that their inverse is their transpose, meaning that $U^T = U^{-1}$. This lends to the following property, $UU^T = U^T U = I$, where I is the identity matrix.

If one wanted to construct the matrix D such that the eigenvalues descended down the diagonal (i.e. for an NxN matrix, the $D_{1,1}$ entry had the largest eigenvalue and the $D_{N,N}$ entry had the smallest eigenvalue), one would have to construct the matrix U as follows:

$$U = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ v_1 & v_2 & \cdots & v_N \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix} \tag{8}$$

where v_1 is the eigenvector associated with λ_1, the largest eigenvalue, v_2 is the eigenvector associated with λ_2, the second largest eigenvalue, and so on with v_N being the eigenvector associated with λ_N, the smallest eigenvalue. If one were to diagonalize a covariance matrix C by using the matrix U constructed in this fashion, one would essentially be projecting the variance of the given population onto the eigenspace formed by the eigenvectors of the covariance matrix. Hence, one would discover how much variance existed along the coordinates specified by v_1, v_2 and so on. It should be noted that the amount of variance projected onto

v_1 is the greatest amount of variance that can be projected by the data [13], the amount projected onto v_2 is the second greatest amount of variance that can be projected by the data and is in a direction orthogonal to v_1 and so on. v_1 is called the first principal component, v_2 is called the second principal component and so on.

One can rotate a sample into the coordinate system specified by the principal components by multiplying the sample data X on the left by U^T (i.e. the matrix which has the eigenvectors of C as rows). In the case that one were to multiply all sample data by the matrix U^T, the covariance matrix would also be rotated according to the matrix U^T. Let us examine what this rotation would cause to occur to the covariance matrix.

The covariance equation can also be defined by the following equation: $C_x = E[(X - E(X))(X - E(X))^T]$, where X is a vector of random variables. The expected value presented here, for a randomly chosen vector X, is the difference of X from the mean of the population (i.e. $E(X)$ is estimated by the mean of the population) times the transpose of the same. Let us examine what happens to the covariance matrix C when the data is multiplied by an arbitrary matrix. We will use the following two equations while examining the transformation process:

$$(AB)^T = B^T A^T \tag{9}$$

$$E(AX) = AE(X) \tag{10}$$

where A, B, and C are matrices and X is a vector of random variables. Letting $y = Ax$, where A is a matrix of constants, we have the following:

$$C_y = C_{Ax}$$
$$= E[(AX - E(AX))(AX - E(AX))^T]$$

$= E[(AX - AE(X))(AX - AE(X))^T]$, using (10)

$= E[(A(X - E(X)))(A(X - E(X)))^T]$

$= AE[(X - E(X))(X - E(X))^T A^T]$, using (10) and (9)

$= AE[(X - E(X))(X - E(X))^T]A^T$, using (10)

$= AC_xA^T$

Note that the above equation works in the case where A is any matrix of constants. Now, let A be given by U^T, then the last equation lends to $C_y = C_{U^T x} = U^T C_x U$ since the transpose of a transpose of a matrix will give you back the original matrix. Now, $U^T C_x U$ is the exact equation to diagonalize the covariance matrix, since U^T is the matrix which has the eigenvectors of C as its rows. Hence, when the data is rotated into the coordinates specified by the principal components, the covariance matrix of the rotated data is diagonalized. Hence, the covariance between any two coordinates in the sample is zero, the correlation between any two coordinates is zero, and, if the sample is assumed to be drawn from a multivariate normal distribution where the variables are jointly normally distributed, then each coordinate is independent of the other coordinates [14, 15].

Principal components are a powerful tool in that they can allow one to express collected data using a set of uncorrelated coordinates. They also allow one to express data in such a way that the direction along which the greatest amount of variance occurs in the sample is given by the first coordinate, the direction along which the second greatest amount of variance (which is orthogonal to the variance provided by the first coordinate) is given by the second coordinate, and so on. In other words, it does a least squares linear curve fit of the data to find the first principal component (or first coordinate), then, excluding that coordinate, does a least squares linear curve fit for the remaining data, and so on. Principal component analysis (PCA) can also be used to reduce dimensionality in a way

which least impacts the data (for data compression) or to extract noise from a signal by excluding those eigenvectors associated with the smallest eigenvalues from the matrix U (i.e. by making U an NxM matrix which has the eigenvectors associated with the M largest eigenvalues in descending order for its columns - so $M \leq N$, the number of dimensions in the original matrix) [16].

CHAPTER 3

A Conventional SOM Reliability Measure

3.1 Basis of existing criteria

SOM reliability criteria, as proposed by Cottrell, are based on measures performed on the map itself, evaluating where pairs of training data fall on the map with relation to each other [8]. The act of performing measurements on the map itself in order to measure the stability or convergence of a map is a common practice [6, 7, 8]. Cottrell et al provided an elegant set of statistical tools to assess the reliability and organization of SOMs. These tools are based on the stability of neighborhood relations and the reliability of the quadratic quantization error of a SOM [8]. The method works by creating many maps which are trained using bootstrap samples of training data and analyzing the quantization error and neighborhood relationships in the resulting maps.

3.2 SOM Reliability Measures

3.2.1 Quantization Error

The quantization error of a given training vector may be defined for an arbitrary metric as follows [2]: $d(x, m_c) = \min_i \{d(x, m_j)\}$, where x is a training vector, m_j is a neuron in the SOM, and c is the best matching neuron in the SOM. Typically the Euclidean distance is used to compute the quantization error. The Euclidean distance is defined as follows: $\sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}$ where the subscript i indicates the ith coordinates in each of the vectors x and y. The quadratic quantization error of a training vector is described as the squared Euclidean distance from the training vector to its best matching neuron in the SOM (i.e. simply remove the square root in the Euclidean distance formula) [8]. If one were to sum the quadratic quantization error over all training data, one would calculate the

intra-class sum of squares [8].

3.2.2 Neighborhood Stability

A stability criterion proposed by Cottrell et al introduces the idea of measuring the significance of neighborhood stabilities in order to analyze whether or not inputs close in the sample space remain close when projected on the SOM. In addition, checks are performed in order to determine whether or not the neighborhood relationships are preserved from map to map [i.e. if many SOMs are created using bootstrap samples of the training data, and the training data is projected onto each of the SOMs, do the training vectors appear to be randomly mapped within a defined neighborhood or is their neighborhood relation significant][8]. In order to fully understand neighborhood stability, one needs to understand the NEIGH function, the random map, and the STAB function, which are discussed in the following sections.

The NEIGH Function

The neighborhood function is defined as follows [8]

$$NEIGH^{b}_{i,j}(r) = \begin{cases} 1 & \text{if } x_i \text{ and } x_j \text{ are neighbors within radius } r \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

This means that if training vectors x_i and x_j exist in bootstrap sample b, and, for the map trained using bootstrap sample b, their best matching neurons on the map are within radius r from each other, then they are neighbors. Within radius r means that the neurons under consideration are within r from each other in each dimension of the SOM. Typically, SOMs are two-dimensional, hence if there is a given neuron, its neighbors would be defined by a square with sides of length $2r+1$ which had the given neuron at its center. Consider the following example from

Cottrell et al [8].

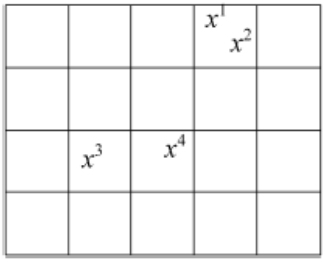

Figure 2: Projections onto a 4 x 5 SOM

Following are some of the values of the $NEIGH$ function as measured on the example above: $NEIGH_{1,2}(0) = 1$, $NEIGH_{1,4}(1) = 0$, $NEIGH_{3,4}(1) = 1$, $NEIGH_{1,3}(1) = 0$, and $NEIGH_{1,3}(2) = 1$.

The Random Map

The random map may be defined as a map onto which training data is mapped in an entirely random way (i.e. the best matches appear to be completely random). The probability of two training vectors being mapped as neighbors would, in most cases, be given by $p = \frac{\tau}{A}$, where τ is the number of neurons in the neighborhood anchored at one of the two training vectors and A is the number of neurons in the SOM. This probability represents the odds of one training vector being randomly mapped onto a neuron which lies in the neighborhood of the other. For a two dimensional SOM, and a neighborhood radius r, p would be given by $\frac{(2r+1)^2}{A}$.

The probability p is actually a little more complicated than the simple formula provided above due to edge effects. Neurons near the edge of a map may not have a full neighborhood of size r. Consider the case where $r = 2$. A neuron in the middle of the upper edge of a given SOM will not have any neighbors above it, hence the $r = 2$ neighborhood for this neuron will extend two neurons down and two neurons left and right (if possible). In other words, its $r = 2$ neighborhood, assuming that the SOM had two neurons left, right, and down, would consist of $(2*2+1)(2*2-1)$

neurons (i.e. 15 neurons as opposed to the 25 neurons which would be implied by the simple formula). In the following figure of a 7 by 9 SOM, those neurons which share the same sized neighborhood where $r = 2$ are highlighted using the same color. The grey neurons use the simple formula for τ (the numerator of p), namely $(2r+1)^2 = (2 \cdot 2 + 1)^2 = 25$ and these neurons account for 15 out of the 63 total neurons in the map. The other neurons have to take edge effects into account.

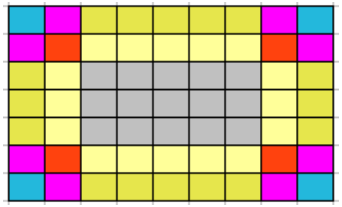

Figure 3: 7 x 9 SOM where the neurons which have the same sized $r = 2$ neighborhood are highlighted with the same color

One can create a general probability p for a neighborhood of size r by taking a weighted average of the percentage of neurons within radius r from a neuron over the whole map. For instance, the weighted average for the above figure would start out as follows: $p_{map} = \frac{15}{63} * \frac{25}{63} + \frac{16}{63} * \frac{20}{63} + \cdots$. One could interpret this as 15 out of 63 neurons (they grey neurons) have a p value of $\frac{25}{63}$, 16 out of 63 neurons (the light yellow neurons) have a p value of $\frac{20}{63}$, and so on. Notice that in p_{map} each item in the sum will have the map size in the denominator twice. A formula to generically compute p_{map} is as follows:

$p_{map} = Center + Sides + Diagonal$, such that

$Center = \frac{(2r+1)^2(x-2r)(y-2r)}{A^2}$

$Sides = \frac{2(x-2r)+2(y-2r)}{A^2} \sum_{i=1}^{r}(2r+1)(2r+1-i)$

$Diagonal = \frac{4}{A^2}\{\sum_{i=1}^{r}(2r+1-i)^2 + \sum_{i=1}^{r-1}(2r-i)\sum_{j=2r-i+1}^{2r} 2j\}$

where x is the number of columns, y is the number of rows, and A is the number of neurons in the SOM (i.e. $A = xy$). Note that each of the summations in the above formulas will only make a contribution to p_{map} in the case where the

upper bound of the summation is greater than or equal to the lower bound of the summation. In the case where the upper bound of the summation is less than the lower bound of the summation, the summation will be excluded in the calculation of p_{map} since the summation will not be valid in this case. The *Center* contribution to p_{map} in the example provided above would come from the grey neurons. The *Sides* contribution would come from the yellow and light green neurons. The *Diagonal* contribution would come from the remaining neurons with the red and blue neurons accounted for by the first summation of the *Diagonal* formula and the magenta neurons accounted for by the second summation (which includes an additional summation in its formula). For large SOMs, the edge effects become less significant. In these cases, the simple formula $p = \frac{(2r+1)^2}{A^2}$, serves an adequate approximation for p_{map} and is easier to compute. It is important to note that p_{map} represents the probability of two training vectors being randomly mapped onto neurons which lie within radius r of each other (i.e. the probability that they are mapped in the same r neighborhood in a completely unorganized or random map).

The STAB Function

The stability function is defined as follows [8, 17]

$$STAB_{i,j}(r) = \frac{\sum_{b=1}^{B} NEIGH_{i,j}^{b}(r)}{B} \tag{12}$$

Recall that $NEIGH_{i,j}^{b}(r)$ means that if training vectors x_i and x_j exist in bootstrap sample b, and, for the map trained using bootstrap sample b, their best matching neurons on the map are within radius r from each other, then they are neighbors. For each pair of training vectors, $STAB_{i,j}(r)$ is computed over the B bootstrap samples which include both x_i and x_j. Thus, B could be different for different pairs of training vectors. $STAB_{i,j}(r)$ essentially captures the fraction, or proportion, of the number of times that x_i and x_j were neighbors in the bootstrapped maps.

Measuring Non-randomness of Neighborhood Relations of SOMs

When analyzing the neighborhood relationships of the SOM, one must ask whether or not the relationships are significant. We know that for the random map, the probability of two training vectors being randomly mapped as neighbors is p_{map}. We can create a binomial random variable Y which has p_{map} as its probability of success [8]. For each pair of training vectors x_i and x_j, we can treat the B bootstrap samples that x_i and x_j appear in as the number of trials or samples from the binomial distribution. If we hypothesize that x_i and x_j are only ever randomly mapped as neighbors, then we would expect $STAB_{i,j}(r)$ to fall into the following interval (as given by the estimate of proportions [14]):

$$\hat{\theta} - z_{\frac{\alpha}{2}} \cdot \sqrt{\frac{\hat{\theta}(1-\hat{\theta})}{B}} < \theta < \hat{\theta} + z_{\frac{\alpha}{2}} \cdot \sqrt{\frac{\hat{\theta}(1-\hat{\theta})}{B}} \tag{13}$$

where $\hat{\theta}$, is given by p_{map}, the probability of a pair of training data randomly being neighbors, and B is the number of bootstrap samples. Note that $\hat{\theta}$ is the estimator of θ, the proportion of times x_i and x_j are randomly neighbors. The 95% confidence interval would be given by:

$$\hat{\theta} - 1.96 \cdot \sqrt{\frac{\hat{\theta}(1-\hat{\theta})}{B}} < \theta < \hat{\theta} + 1.96 \cdot \sqrt{\frac{\hat{\theta}(1-\hat{\theta})}{B}} \tag{14}$$

The confidence intervals above use the normal approximation for the binomial distribution and are only valid when B is large enough. Miller provides the following conditions as rules of thumb in order to determine when the normal approximation to the binomial distribution may be used [14]

$$B\theta > 5 \tag{15}$$

$$B(1-\theta) > 5 \tag{16}$$

Cottrell et al provide a different rule of thumb, which consists of simply using 10 instead of 5 on the right hand side of each of the inequalities given by (15) and (16) [8]. The 95% confidence interval above is a 95% confidence interval for the proportion of times that x_i and x_j would be mapped as neighbors in the random case. Thus, we can conclude with a test level of 5%, that if $STAB_{i,j}(r)$ lies outside of the 95% confidence interval, then the neighbor relationship between x_i and x_j is meaningful, or non-random.

CHAPTER 4

A New Approach to SOM Convergence

4.1 Overview

SOM convergence measures are usually based upon one of several common methods. On method is measuring neighborhood relationships on the SOM, or a family of SOMs created from the same training data. Another method is based on imposing energy functions on the neurons of the SOM and minimizing these energy functions. Still other methods are based on modifying the SOM algorithm itself so that statistical measures or other objective analysis techniques can be imposed. These methods tend to be more complicated than the SOM algorithm itself and/or are time intensive and computationally unwieldy.

One interesting observation is that these convergence criteria do not impose statistical tests to compare the training data to the neurons of the map which are meant to model the probability density of the training data. In this chapter, we propose a convergence measure which is based on such a comparison. The convergence measure proposed is fast and provides an indicator of how well the neurons in the SOM model the probability space spanned by the training data.

4.2 Normalizing and Rotating Along the Principal Components

In order to model the input data, we first remove any features which are constant throughout the entire training data set as these features provide no classification power. Then, we normalize the input data in a standard fashion over each dimension, or feature. Standard normalization essentially consists of subtracting out the mean and dividing by the standard deviation. Hence, the mean along each of the normalized dimensions is zero and both the standard deviation and variance are one (note that the variance is simply the standard deviation squared). Nor-

malizing in this fashion enables us to create a SOM using dimensionless features which are scaled similarly. In other words, we do not have to worry about the scaling of each of the features. The SOM algorithm is sensitive to the scaling of the training data.

Making the data in each feature standard normal allows each feature to be treated equally during the SOM training. If feature i is measured in thousands of inches with a standard deviations in the 10s of inches and feature j is measured in tens of inches with a standard deviation in inches, then the SOM will model feature i better than j. This makes sense since the SOM is trying to recreate the probability density of the training data used to create it. Since, in this case, there is a greater variance in dimension i, the updates performed by the SOM algorithm will have more impact on dimension i than dimension j. If features i and j are normalized, then the SOM will provide dimension i and j equal attention since their variances are equal. Subtracting out the mean is common when performing statistical analyses. Note that the impact of dividing by the standard deviation is that the covariance matrix of the new data will be equal to the correlation matrix of the original data (and the new data). Hence, correlation is preserved using this type of normalization scheme.

Suppose we call the new training data, which has been normalized in each dimension, Y. We now want to rotate this data into the dimensions specified by the principal components in accordance with the procedure in section 2.3. Call the transformed data, the training data which has been rotated into the dimensions specified by the principal components, X. By definition, the covariance matrix of the data set X will now be diagonalized, with the eigenvalues of the covariance matrix of data set Y along the diagonal in descending order. In addition, the sum of the diagonal entries (i.e. the trace of the matrix) will be equal to the number

of coordinates in the training data. This occurs because each of the diagonal elements in the covariance matrix of the data set prior to rotation (i.e. of data set Y) would be one. Hence, the trace of the covariance matrix of data set Y is equal to the number of coordinates in the training data. Note that the trace of a matrix is invariant under an orthogonal transformation [18]. Rotations are orthogonal transformations.

Let the covariance of the data set X be given by C. C has zeros everywhere except for the diagonal. Hence, the variables of X are uncorrelated. If we assume that the random variables in X are jointly normally distributed, then we also have that the variables in X are independent [14, 15]. In the case where the random variables in X are not jointly normally distributed, we still have that each of these variables are not correlated. In the case where the variables are jointly normally distributed, we can justifiably treat each of the variables independently. In the case where they are not, the fact that they are uncorrelated (i.e. that they do not have a linear relationship), allows us to focus our analysis on the diagonal terms of the covariance with the greatest attention, ignoring the off diagonal terms (since they are all zero). Also, because Yin and Allison proved that, in a fairly general setting, the SOM will converge on the probability density of the training data in the limit, a converged SOM will have a covariance matrix which is very similar to that of the training data[10]. Hence, the covariance matrix of the neurons of a converged SOM should have zeros for the off diagonal terms, supporting our focus on the diagonal terms. The features in X may be thought of as "PCA" features which are linear combinations of the original features (normalized features).

4.3 A Measure for SOM Convergence

The SOM convergence measure proposed here uses the data X as given in the previous section to train the SOM. In other words, we train the SOM using data

which has been normalized along each feature (i.e. dimension or coordinate) and then rotated into the directions given by the principal components. Once the SOM has been trained, we compare the training data to the SOM along each dimension (or feature). If a SOM has adequately modeled both the mean and the variance of the given feature, we will say that the SOM has converged on the feature. The proportion of the total variance included in that feature will then be added to the convergence measure. In order to better understand these concepts, we will clearly define the mean convergence of a feature, the variance convergence of a feature, and the population based convergence measure.

4.3.1 Mean Convergence

In the case where $\bar{x_1}$ and $\bar{x_2}$ are the values of the means from two random samples of size n_1 and n_2, and the known variances of these samples are σ_1^2 and σ_2^2 respectively, the following formula provides $(1-\alpha)*100\%$ confidence interval for the difference between the means [14]:

$$(\bar{x_1}-\bar{x_2}) - z_{\frac{\alpha}{2}} \cdot \sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}} < \mu_1 - \mu_2 < (\bar{x_1}-\bar{x_2}) + z_{\frac{\alpha}{2}} \cdot \sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}} \qquad (17)$$

To test for SOM convergence, $\bar{x_1}$ is the sample mean of given feature in the training data, $\bar{x_2}$ is the sample mean of the given feature in the neurons of the map, σ_1^2 is the sample variance of the feature in the training data, sample σ_2^2 is the variance of the feature in the map, n_1 is the cardinality of the training data set, and n_2 is the number of neurons in the map. Note that $\bar{x_1}-\bar{x_2}$ is an estimator for $\mu_1-\mu_2$, the true difference between the means of the two populations. We will say that the mean of a particular feature has converged if zero lies in the confidence interval denoted by equation (17). In other words, the confidence interval affords the possibility that the means are equal.

4.3.2 Variance Convergence

The following is the formula for the $(1-\alpha)*100\%$ confidence interval for the ratio of the variances from two random samples [14]:

$$\frac{s_1^2}{{s_2}^2} \cdot \frac{1}{f_{\frac{\alpha}{2},n_1-1,n_2-1}} < \frac{\sigma_1^2}{\sigma_2^2} < \frac{s_1^2}{{s_2}^2} \cdot f_{\frac{\alpha}{2},n_1-1,n_2-1} \tag{18}$$

where s_1^2 and s_2^2 are the values of the sample variances from two random samples of sizes n_1 and n_2 respectively, and where $f_{\frac{\alpha}{2},n_1-1,n_2-1}$ is an F distribution with n_1-1 and n_2-1 degrees of freedom. To test for SOM convergence, s_1^2 will be the sample variance of the feature in the training data, s_2^2 will be the sample variance of the feature in the neurons of the map, n_1 will be the number of training samples (i.e. the cardinality of the training data set) and n_2 will be the number of neurons in the SOM. Note that $\frac{s_1^2}{{s_2}^2}$ is an estimator for $\frac{\sigma_1^2}{\sigma_2^2}$, the ratio of the variances of the two populations. We will say that the variance of a particular feature has converged (or appears to be drawn from the same probability space) if one lies in the confidence interval denoted by equation (18). In other words, the confidence interval affords the possibility that the variances are equal.

4.3.3 The Population Based Convergence Measure

We will say that a SOM has converged on a feature, or that a feature has converged, if both the mean and variance converged for that feature in accordance with the above criterion. We can then form a measure for SOM convergence as follows:

$$convergence = \frac{\sum_{i=1}^{N} \rho_i}{N}, \text{ where } \rho_i = \begin{cases} \lambda_i & \text{if feature } i \text{ has converged} \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

where λ_i is the variance of the data in dimension i and N is the number of dimensions in the training data (which is also the total amount of variance contained in

the training data). Hence, the convergence measure proposed here is essentially the proportion of the total variance contained by those features which converged (i.e. those "PCA" features whose mean and variance were adequately modeled by the neurons in the SOM). Recall that λ_i will be the ith largest eigenvalue of the normalized training data's covariance matrix (since the data was normalized and then rotated into the directions given by the principal components). This approach views the SOM as a sample builder. This convergence measure provides an idea of how well the SOM has replicated the structure of the probability density of the training data.

CHAPTER 5

Results

5.1 Experiment Design

For the experimentation performed in support of this thesis, the proposed convergence measure was compared to the neighborhood stability measure given by Cottrell et al at different levels of training (i.e. for maps trained for different numbers of iterations) [8]. For each data set analyzed and for each number of training iterations considered, 200 bootstrap samples were created from the data set being analyzed. These 200 samples were used to train 200 randomly initialized SOMs for the given number of iterations. The 200 randomly initialized maps then had the neighborhood relationships proposed by Cottrell calculated for them and the average population based convergence measure calculated for them. The 200 maps were created primarily to support Cottrell's neighborhood stability criterion. Since the average population based convergence measure was calculated from the 200 maps created using 200 different bootstrap samples, the plots of the population based convergence measure, called fConv in the plots, will indicate "boot = TRUE" signifying that the values used to create the plots were averages over the bootstrapped maps. The population based convergence measure does not rely on the creation of a bootstrap sample of maps. The population based convergence can be imposed upon a single map in order to see how accurately it reflects the probability density of the training data used to create it. 95% confidence intervals were used for all statistical tests.

Cottrell's STAB indicator was calculated for each pair of training data vectors. If the value of STAB indicated that the neighborhood relationship between the pair was non-random in accordance with equation (13) [i.e. the value of STAB was outside the 95% confidence interval given by the random neighborhood (the

confidence interval given by equation (14))], then the pair was counted as a non-random pair. Each pair was checked in order to see whether or not the pair was non-random. Then the proportion of non-random pairs was calculated (simply the total number of non-random pairs divided by the total number of pairs). Only pairs which were included in at least 50 of the 200 bootstrap samples were analyzed (this included nearly all of the pairs) in order to ensure that the randomness measurements were statistically significant. When the proportion of non-random neighbors stabilizes, the indication is that the SOM is organized and is non-random. Note that the proportion of non-random neighbors will rarely, if ever, be one (i.e. 100% of the pairs of training data have non-random neighborhood relationships) because some pairs of training data will always be "border-lined", meaning that they may or may not be neighbors in a seemingly random fashion. For example, consider data from different clusters. Depending on how the clusters are oriented and where they are located on the map, the pairs of training data such that each member of the pair maps onto a different cluster in the SOM may or may not be neighbors in a seemingly random fashion, especially in the case where the SOM is randomly initialized before training.

For the experimentation, 2-dimensional rectangular SOMs were created with lengths chosen subject to the following constraints:

$$xy \geq M \tag{20}$$

$$\frac{x}{y} = \sqrt{\frac{\sigma_1^2}{\sigma_2^2}} \tag{21}$$

where x is the length of the SOM, y is the height of the SOM, M is the target number of neurons, σ_1^2 is the variance along the first principal component, and σ_2^2 is the variance along the second principal component. Using these constraints, we can compute the sizes of x and y as follows:

$$y = \left\lceil \sqrt{\frac{M}{\sqrt{\frac{\sigma_1^2}{\sigma_2^2}}}} \right\rceil \tag{22}$$

$$x = \left\lceil y\sqrt{\frac{\sigma_1^2}{\sigma_2^2}} \right\rceil \tag{23}$$

Scaling the dimensions of the SOM in this way allows us to create a SOM with at least M neurons which is scaled to optimally capture the variance in those dimensions with the greatest variance (i.e. the first and second principal components). Experimentation has shown $M = 2N$, where N is the cardinality of the training data set, to be a good starting point, especially for smaller data sets. For larger data sets, smaller maps are acceptable. We will refer to the usage of $M = 2N$ in the equations (22) and (23) as the $M = 2N$ sizing method. Creating a SOM using the $M = 2N$ sizing method allows us to build a sample larger than the one represented by the initial training data. A size of $2N$ will also result in a larger SOM which is more capable of modeling the variance of the training data, which is key to modeling a given probability distribution.

It should be noted the Cottrell et al proposed a measure for choosing the size of (or the number of neurons in) a SOM. This method was used in conjunction with equations (22) and (23) in order to ensure consistency with Cottrell's stability criterion. Cottrell's method was based on calculating the intra class sum of squares, SSIntra, for each of the bootstrapped maps created (200 in this case)[8]. Then, the coefficient of variation would be calculated for these 200 values. The coefficient of variation is computed as:

$$CV(SSIntra) = 100\frac{\sigma_{SSIntra}}{\mu_{SSIntra}} \tag{24}$$

where $\sigma_{SSIntra}$ is the standard deviation of SSIntra over the 200 bootstrapped maps and $\mu_{SSIntra}$ is the mean of SSIntra over the 200 bootstrapped maps. Recall that

SSIntra is the sum of the quadratic quantization error over all training data (see Section 3.2.1). The value of CV(SSIntra) would be calculated for 200 maps of different sizes (note that 200 maps would be created for each size under consideration). Then, CV(SSIntra) would be plotted. The last size given before the largest increase in CV(SSIntra) would be used as the appropriate size of the map. The claim was that low values of CV(SSIntra) lead to values of SSIntra which were close among the 200 maps, hence, there appeared to be stability in the quantization error at this size and perhaps, at the next size up, centroids were switching from one cluster to another in the input sample space potentially indicating some sort of instability in the placement of the neurons of the SOM [8]. This appears to be speculation without much proof. Cottrell et al admitted that this method was very empirical in nature but argued that other empirical measures exist in the realm of statistics, such as choosing the number of principal components to keep when performing a principal component analysis. It should be noted that the coefficient of variation only provides a reasonable measure when all measurements of a sample are positive and is intended to be a percentage of the standard deviation with respect to the mean. However, since the intra class sum of squares is always positive, we are guaranteed to only be dealing with positive measurements.

During the experimentation, maps were created using both Cottrell's method and using the $M = 2N$ method described above. In all cases, the results were similar. However, the $M = 2N$ maps tended to have a higher population based convergence measure as the additional neurons allowed the SOM more freedom to model the variance of the training data sample.

CV(SSIntra) was analyzed for SOMs with different numbers of neurons but which were scaled according to equations (22) and (23) with different values substituted for M (recall that M represents the number of neurons desired in the map).

Each map was trained for the same number of iterations. Hence, not surprisingly, CV(SSIntra) sometimes trends upwards over time (especially in the case where the training data is drawn from a more complicated probability space). This is not unexpected. Larger maps need to be trained longer simply because they have more neurons to train (think of teaching one student as opposed to 200 students, clearly teaching 200 students will require more time). The larger maps, if trained longer, would actually have consistently lower quantization errors as they have more freedom to capture the variance inherent to the training data. Hence, the upwards trend may be somewhat misleading and arbitrary. However, in order to stay consistent with the method proposed by Cottrell et al, this method is used regardless.

Note that in the following sections, each plot of the population based convergence measure against the number of iterations also contains additional horizontal lines. These lines represent the variance contained in the first several principal components. The lowest of these lines represents the variance contained in the first principal component, the second lowest line represents the amount of variance contained in the first two components, and so on. "neighRad" appears in some plots in the following sections and different values for neighRad can be observed depending on the data set under analysis. neighRad represents the neighborhood radius size used in the calculation of cottNeigh, the proportion of all neighbors which have a non-random neighbor relationship. It should be noted that during the experimentation for each data set, all neighborhood radii from zero up through the numbers observed in the following plots were analyzed. The neighborhood radii in the plots appearing in the following sections were chosen because they most reflect the underlying clustering of the SOM (given the SOM size used and the a priori knowledge of the number of clusters which are in the data). They were also

chosen in order to ensure that Cottrell's versions of the rules of thumb in Equations (15) and (16) were followed. Also, all results were qualitatively similar in that cottNeigh leveled out after the same, relatively small numbers of iterations. Hence, the additional results were excluded.

Three data sets were analyzed for the experimentation supporting this thesis. The results of the analyses are reported in order of the complexity of the data used. The iris data set is the least complicated data set, consisting of four dimensional data with low variance. The wine data is the second most complicated data set, consisting of thirteen dimensional data with differing variances. The ionosphere data is the most complicated data set, consisting of thirty-three non-constant dimensions with differing variances. Each data set will be described in more detail in the section containing its results.

5.2 Iris Experiment Results

For the following results, the Fisher/Anderson iris data set [19] was used. This data set consists of four dimensional measurements which have a small variance. The measurements are for the sepal length, the sepal width, the petal length, and the petal width of three classes of irises. There are 150 training instances in this data set. This data set is very easy to model due to its low dimensionality and its small variance. The plot of CV(SSIntra) in this section will show how CV(SSIntra) increases with the number of neurons in the SOM. However, this may be misleading since the larger maps require more training in order to converge on a good solution.

5.2.1 CV(SSIntra)

For the iris data set, CV(SSIntra) was calculated for maps of increasing size which were trained for 5,000 iterations. The results are as follows:

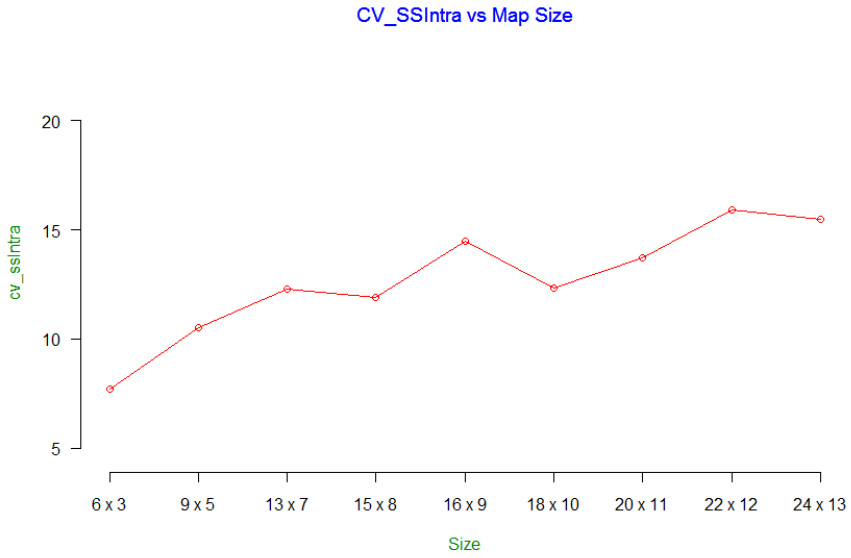

Figure 4: CV(SSIntra) plotted as a function of map size

The last major increase (and possibly the only major increase) appearing here occurs in the jump from the 15×8 map to the 16×9 map. Hence, Cottrell's method for choosing the size of the SOM would lead one to choose the 15×8 SOM.

5.2.2 Results Using 15×8 SOMs

Using 15×8 SOMs, both the population based converge measure and proportion of non-random neighbors as given by Cottrell's neighborhood stability method were calculated. They were calculated for maps trained using different numbers of iterations. For each number of iterations included in the following maps, 200 maps were created so that these statistics could be performed. In the following plots, fConv represents the population based convergence measure, cottNeigh represents the proportion of all neighbors which have a non-random neighbor relationship, and SSMean represents the average quadratic quantization error.

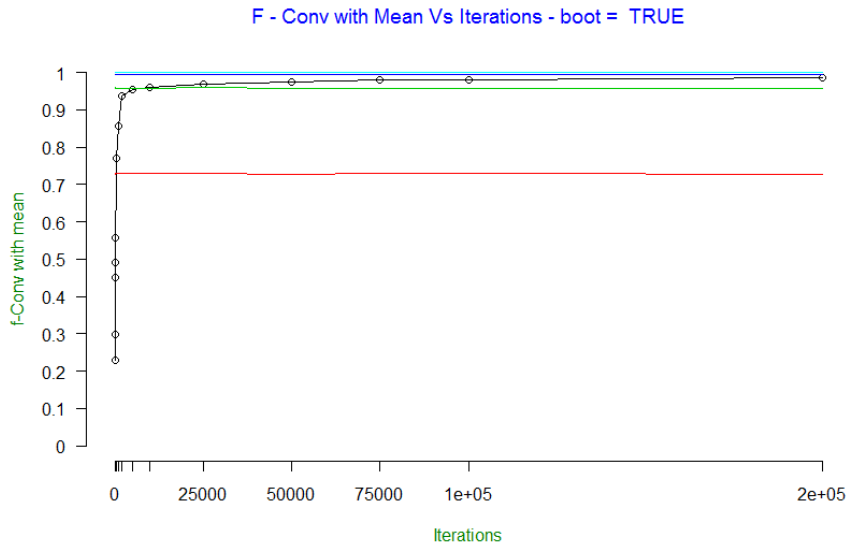

Figure 5: Population Based Convergence Measure Plotted Against Iterations - Data averaged over the 200 bootstrapped maps (that is the meaning of boot = TRUE)

It should be observed that fConv levels out after around 10,000 iterations. In addition, the variance in the first two principal components is captured at 10,000 iterations (i.e. this is where the black line crosses the green line).

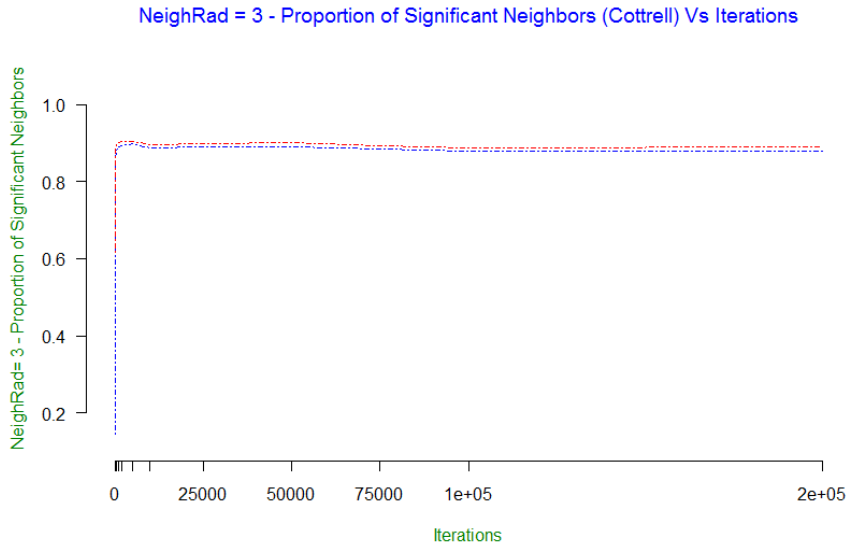

Figure 6: Proportion of Non-Random Neighbors Plotted Against Iterations. A neighborhood radius of three is used. The red line does not take edge effects into account while the blue line does.

It should be observed that cottNeigh also levels out after around 10,000 iterations. Recall that the variance in the first two principal components is captured at 10,000 iterations (as indicated in the plot of the population based convergence measure). Here, cottNeigh and fConv do not appear to be very different.

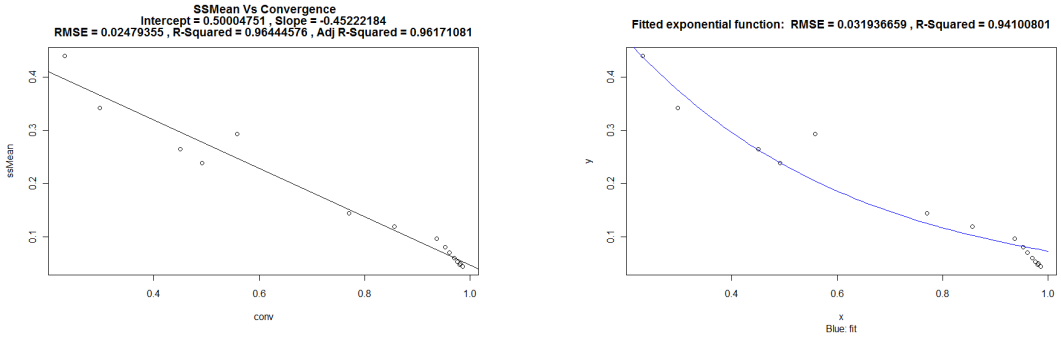

Figure 7: SSMean Plotted Against fConv, the Population Based Convergence Measure. On the left, ssMean is plotted as a linear function of fConv while on the right, ssMean is plotted as an exponential decay function of fConv.

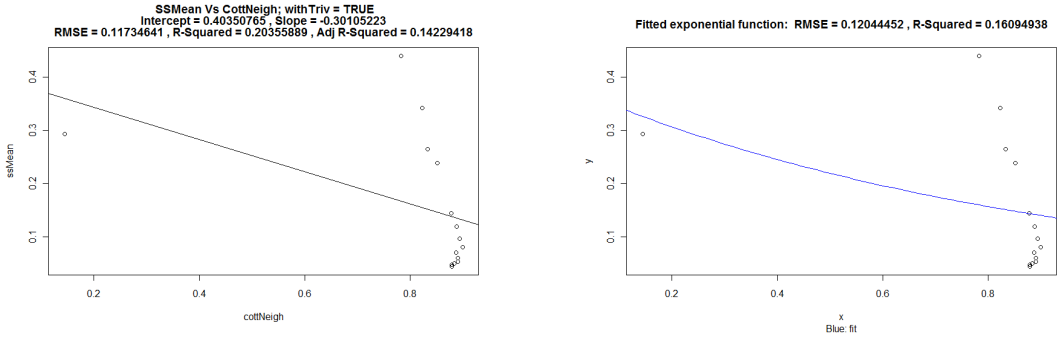

Figure 8: SSMean Plotted Against the proportion of non-random neighbors, cottNeigh. On the left, ssMean is plotted as a linear function of cottNeigh while on the right, ssMean is plotted as an exponential decay function of cottNeigh.

From Figures 7 and 8, it is clear that fConv tracks the SSMean better than cottNeigh regardless of whether or not a linear or exponential decay fit is used.

Note that the closer the R-squared value is to one, the better the model. One interesting observation is that in Figure 8, we see that when cottNeigh indicates stability, the SSMean is not settled at all and takes on a wide range of values. However, when fConv indicates high levels of convergence (see Figure 7), the SSMean is stable and takes on a more compact range of values. Also, as fConv increases, SSMean decreases. With cottNeigh, there is no real correlation in the limiting case.

A second round of experimentation was performed on the iris data set using 24×13 SOMs. This dimensionality is given by the $M = 2N$ method proposed earlier. However, the results were not qualitatively different from those reported using the 15×8 SOMs and hence were excluded.

5.3 Wine Experiment Results

For the following results, the wine data set from the UCI machine learning repository [20] was used. This data set consists of thirteen dimensional measurements which have differing and unique variances. The measurements consist of different chemical properties of three different wines. There are 178 training instances in this data set. This data set is moderately difficult to model because it contains thirteen dimensions with differing variances. Hence, the plot of CV(SSIntra) in this section will have an increase with the number of neurons in the SOM. This will occur for this data set primarily because the data set is moderately difficult to model. Hence, the larger maps need more training iterations in order to converge on a good solution.

5.3.1 CV(SSIntra)

For the wine data set, CV(SSIntra) was calculated for maps of increasing size which were trained for 5,000 iterations. The results are as follows:

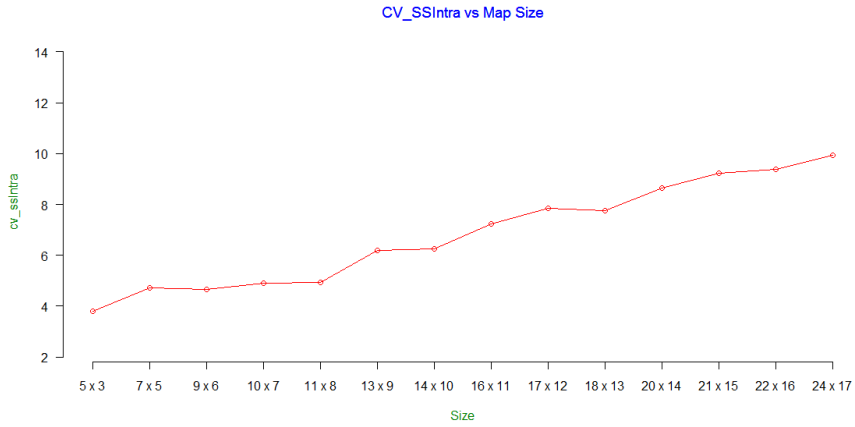

Figure 9: CV(SSIntra) plotted as a function of map size

The last major increase appearing here occurs in the jump from the 14×10 map to the 16×11 map. Hence, Cottrell's method for choosing the size of the SOM would lead one to choose the 14×10 SOM. Note that this method is quite subjective. While this was the author's interpretation of the CV(SSIntra), another individual may disagree.

5.3.2 Results Using 14×10 SOMs

Using 14×10 SOMs, both the population based converge measure and proportion of non-random neighbors as given by Cottrell's neighborhood stability method were calculated. They were calculated for maps trained using different numbers of iterations. For each number of iterations included in the following maps, 200 maps were created so that these statistics could be performed. In the following plots, fConv represents the population based convergence measure, cottNeigh represents the proportion of all neighbors which have a non-random neighbor relationship, and SSMean represents the average quadratic quantization error.

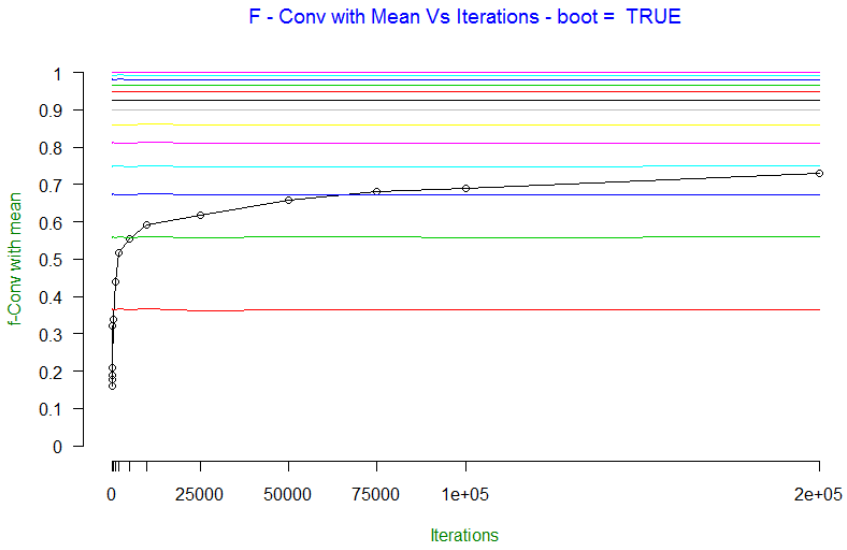

Figure 10: Population Based Convergence Measure Plotted Against Iterations - Data averaged over the 200 bootstrapped maps (that is the meaning of boot = TRUE)

It should be observed that fConv levels out after around 75,000 iterations. In addition, the variance in the first two principal components is captured at 10,000 iterations.

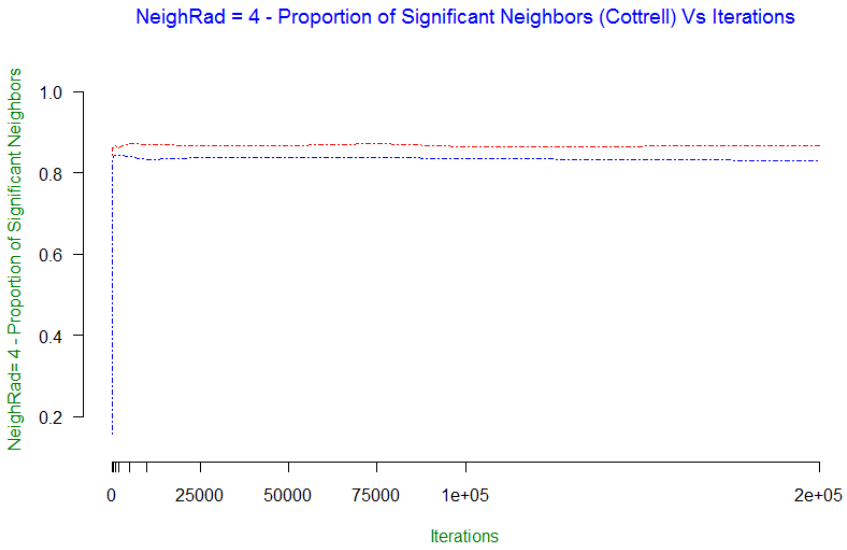

Figure 11: Proportion of Non-Random Neighbors Plotted Against Iterations. A neighborhood radius of four is used. The red line does not take edge effects into account while the blue line does.

It should be observed that cottNeigh levels out after only 5,000-10,000 iterations. Here, cottNeigh and fConv *do* appear to be different.

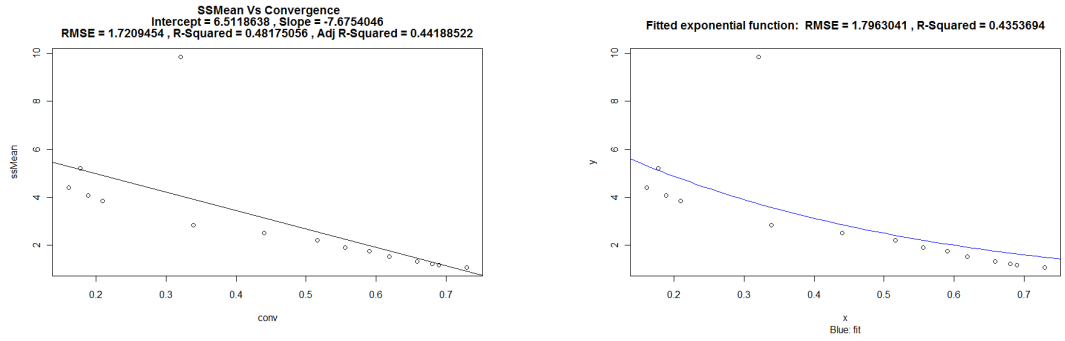

Figure 12: SSMean Plotted Against fConv, the Population Based Convergence Measure. On the left, ssMean is plotted as a linear function of fConv while on the right, ssMean is plotted as an exponential decay function of fConv.

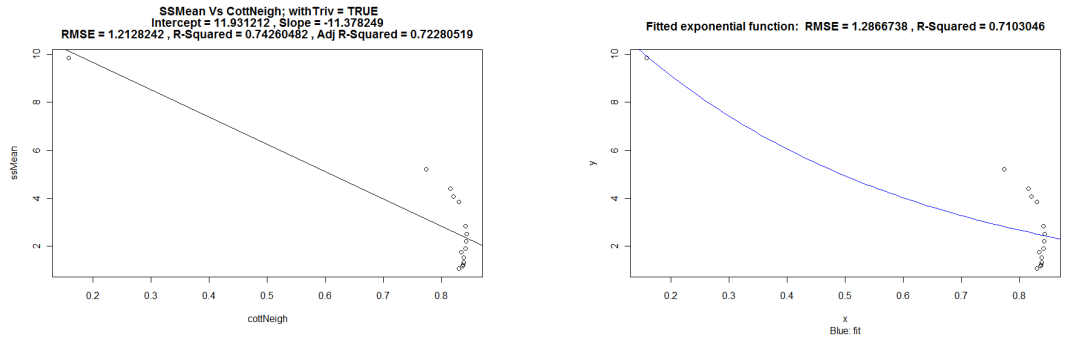

Figure 13: SSMean Plotted Against the proportion of non-random neighbors, cottNeigh. On the left, ssMean is plotted as a linear function of cottNeigh while on the right, ssMean is plotted as an exponential decay function of cottNeigh.

From these plots it is clear that in the limiting case (i.e. when the measures fConv and cottNeigh indicate that the map is converged), fConv tracks the SSMean better than cottNeigh regardless of whether or not a linear or exponential decay

fit is used. The R-squared value in the above plots is misleading because there is one major outlier in all plots. This outlier is from the first data point which represents a completely random SOM (i.e. a randomly initialized SOM trained for zero iterations). If this outlier is removed, the updated plots would more fully show how well fConv tracks the SSMean and how poorly cottNeigh does the same. The plots with the outlier removed are shown.

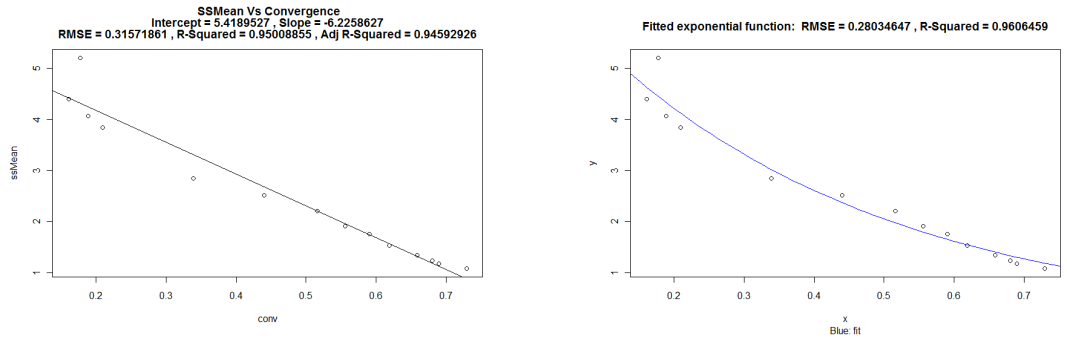

Figure 14: SSMean Plotted Against fConv, the Population Based Convergence Measure with the one outlying point removed. On the left, ssMean is plotted as a linear function of fConv while on the right, ssMean is plotted as an exponential decay function of fConv.

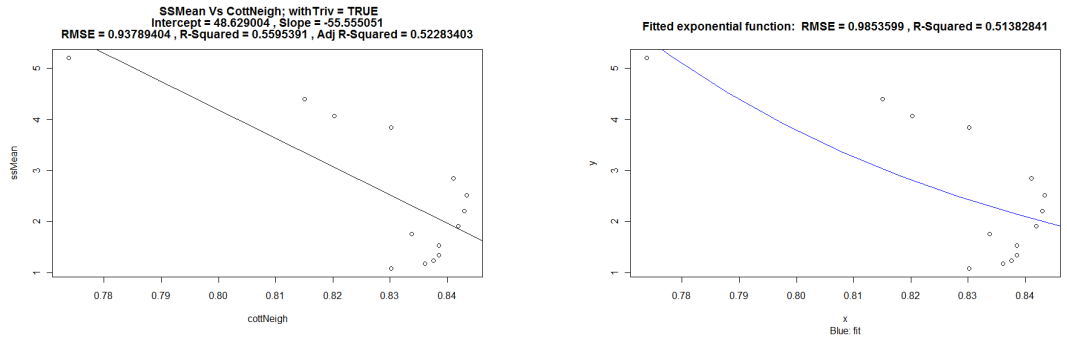

Figure 15: SSMean Plotted Against the proportion of non-random neighbors, cottNeigh with the one outlier removed. On the left, ssMean is plotted as a linear function of cottNeigh while on the right, ssMean is plotted as an exponential decay function of cottNeigh.

From Figures 14 and 15 it is clear that fConv tracks the SSMean better than cottNeigh. Again in Figure 15, we see that when cottNeigh indicates stability,

the SSMean is not settled at all and takes on a wide range of values. However, when fConv indicates high levels of convergence (see Figure 7), the SSMean is stable and takes on a more compact range of values. Also, as fConv increases, SSMean decreases. With cottNeigh, there is no real correlation in the limiting case. It should also be noted that fConv is a more conservative convergence measure, only indicating convergence after cottNeigh has indicated stability in the map (i.e. fConv implies cottNeigh but the converse is not true). Consider the following plot of both fConv and cottNeigh plotted as a function of iterations.

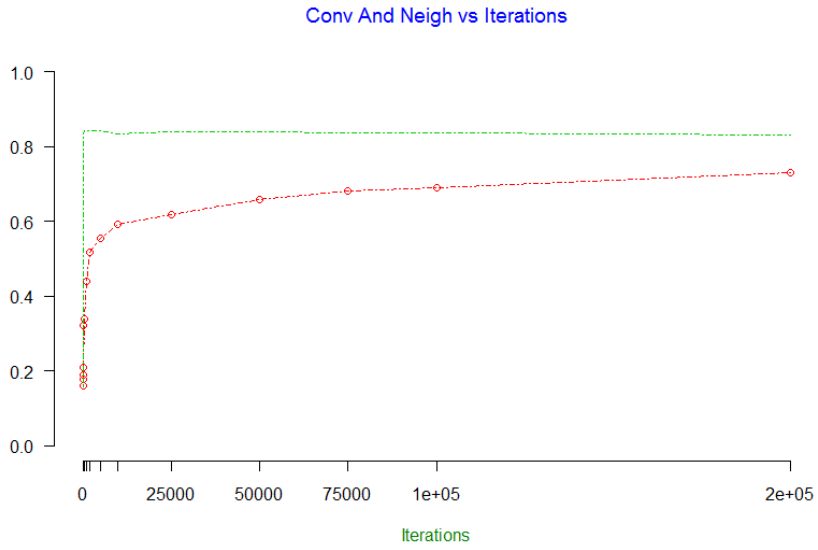

Figure 16: cottNeigh (green) and fConv (red) plotted as functions of iterations. It is clear that fConv levels out after cottNeigh and hence provides a more conservative measure of convergence.

A second round of experimentation was performed on the wine data set using 24×17 SOMs. This dimensionality is given by the $M = 2N$ method proposed earlier. However, the results were not qualitatively different from those reported using the 14×10 maps and hence were excluded. The main difference was that fConv achieved a higher level of convergence in the limiting case (i.e. it reached over 85% convergence after $200,000$ iterations when the 24×17 maps were used

as opposed to the 70% convergence obtained via the usage of the 14×10 SOMs). This makes sense since larger maps give the model more freedom to capture the variance contained in the training data.

5.4 Ionosphere Experiment Results

For the following results, the ionosphere data set from the UCI machine learning repository [21] was used. This data set consists of thirty-three non-constant measurements which have differing variances (there are actually 34 dimensions but one of them is constant and hence was excluded in the experimentation). The measurements are constructed from radar returns from the ionosphere. The returns are classified in a binary fashion based on whether or not the returns indicated that there was some structure in the ionosphere. There are 351 training instances in this data set. This data set is very difficult to model because of its high dimensionality and the noise in the measurements. Hence, the plot of CV(SSIntra) in this section will have an increase with the number of neurons in the SOM. This will occur for this data set primarily because the data set is difficult to model. Hence, the larger maps need more training iterations in order to converge on a good solution.

5.4.1 CV(SSIntra)

For the ionosphere data set, CV(SSIntra) was calculated for maps of increasing size which were trained for 10,000 iterations. The results are as follows:

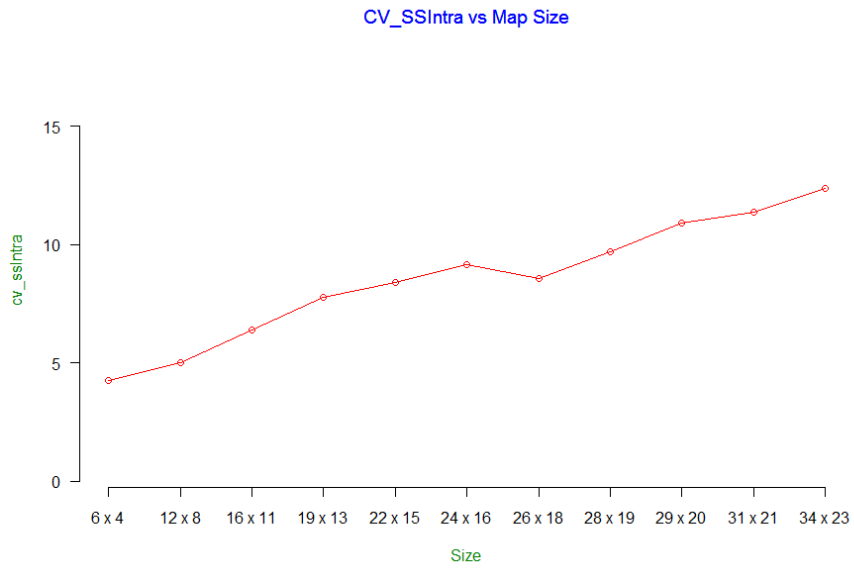

Figure 17: CV(SSIntra) plotted as a function of map size

The last major increase appearing here occurs in the jump from the 26×18 map to the 28×19 map. Hence, Cottrell's method for choosing the size of the SOM would lead one to choose the 26×18 SOM. One could argue that there is another reasonably major increase between the 28×19 map and the 29×20 map; however, since the choice of map size is empirical, and somewhat arbitrary using the CV(SSIntra) method, the 26×18 map is chosen by the author in this case.

5.4.2 Results Using 26×18 SOMs

Using 26×18 SOMs, both the population based convergence measure and the proportion of non-random neighbors as given by Cottrell's neighborhood stability method were calculated. They were calculated for maps trained using different numbers of iterations. For each number of iterations included in the following maps, 200 maps were created so that these statistics could be performed. In the following plots (and as in the previous sections), fConv represents the population based convergence measure, cottNeigh represents the proportion of all neighbors which have a non-random neighbor relationship, and SSMean represents the aver-

age quadratic quantization error.

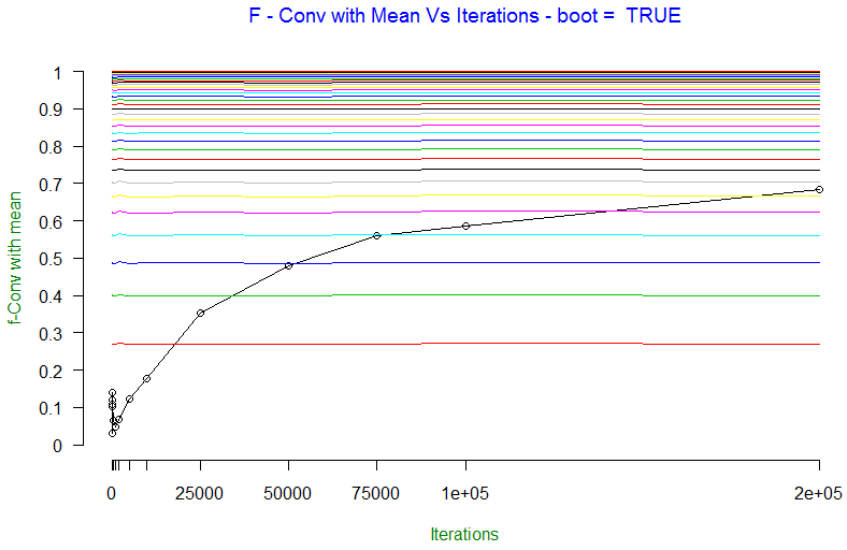

Figure 18: Population Based Convergence Measure Plotted Against Iterations - Data averaged over the 200 bootstrapped maps (that is the meaning of boot = TRUE)

It should be observed that fConv levels out after around 75,000 iterations (it continues upwards but the derivative decreases to a small value around this point). The variance in the first two principal components is captured at 50,000 iterations.

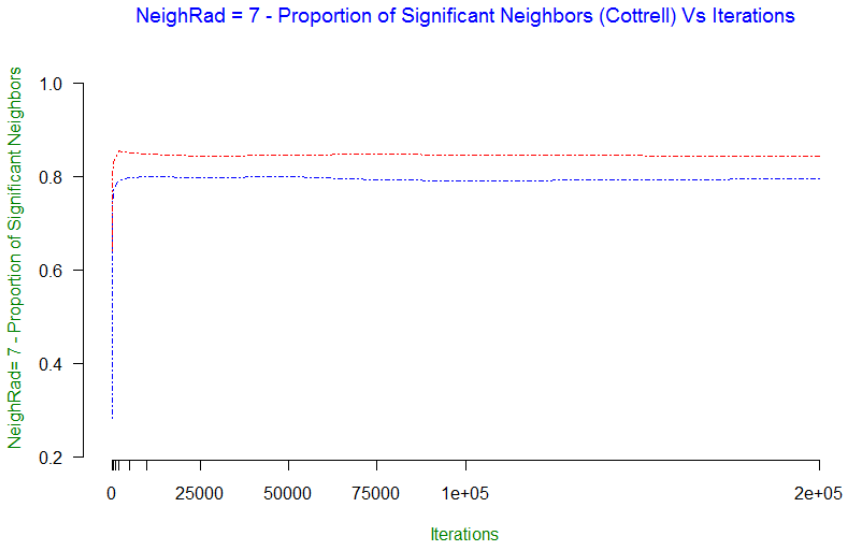

Figure 19: Proportion of Non-Random Neighbors Plotted Against Iterations. A neighborhood radius of seven is used. The red line does not take edge effects into account while the blue line does.

It should be observed that cottNeigh levels out after only 5,000-10,000 iterations. Here, cottNeigh and fConv appear to be *very* different. cottNeigh indicates stability after only 5,000-10,000 iterations while the variance along the first two principal components has not even been captured until around 50,000 iterations. Capturing the variance in at least these two components is important since the SOM itself is a 2-dimensional visual representation.

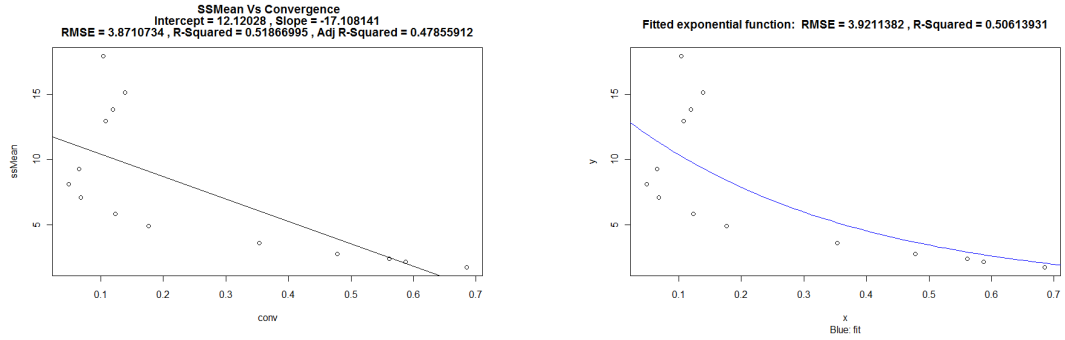

Figure 20: SSMean Plotted Against fConv, the Population Based Convergence Measure. On the left, ssMean is plotted as a linear function of fConv while on the right, ssMean is plotted as an exponential decay function of fConv. The maps trained for zero iterations were excluded.

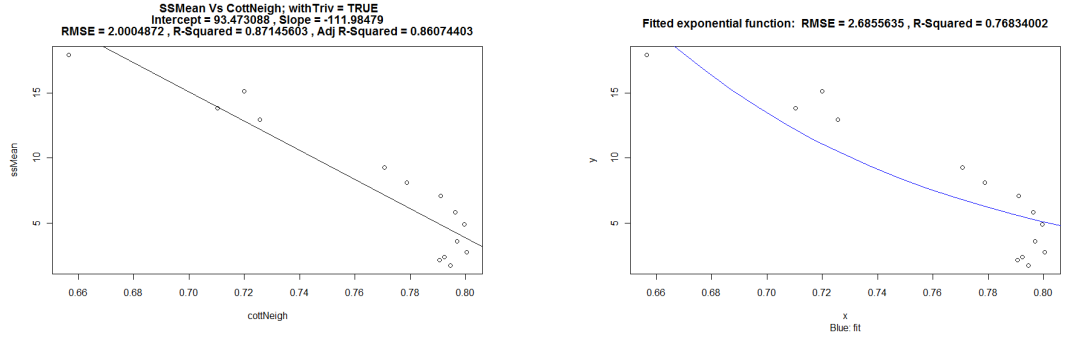

Figure 21: SSMean Plotted Against the proportion of non-random neighbors, cottNeigh. On the left, ssMean is plotted as a linear function of cottNeigh while on the right, ssMean is plotted as an exponential decay function of cottNeigh. The maps trained for zero iterations were excluded.

From these plots it appears that in the limiting case (i.e. when the measures fConv and cottNeigh indicate that the map is converged), fConv tracks the SSMean better than cottNeigh regardless of whether or not a linear or exponential decay fit is used. The R-squared value in the above plots is misleading because there are several major outliers in all plots where the ssMean is plotted against either fConv or cottNeigh. These outliers are from the first few data points where SOMs which were not trained enough to even remotely model the very complex ionosphere data set were used. If these outliers are removed, the updated plots would more fully show how well fConv tracks the SSMean in the limiting case and how poorly cottNeigh does the same. The plots with the outliers removed are shown below.

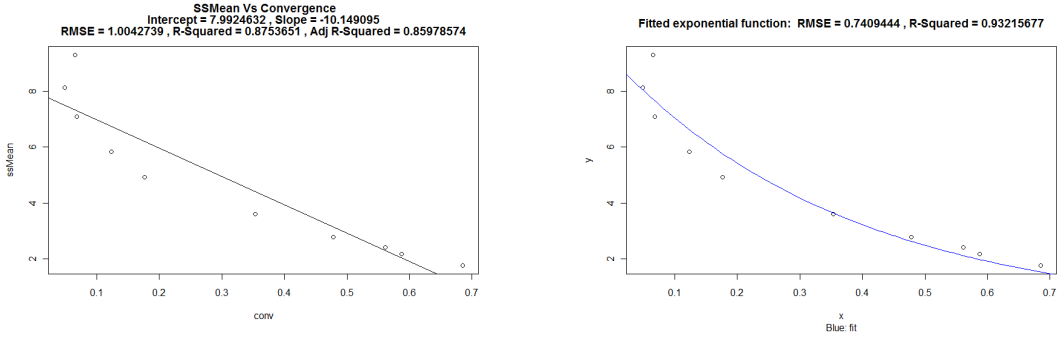

Figure 22: SSMean Plotted Against fConv, the Population Based Convergence Measure with the outliers removed. Only maps trained for 500 or more iterations are considered. On the left, ssMean is plotted as a linear function of fConv while on the right, ssMean is plotted as an exponential decay function of fConv.

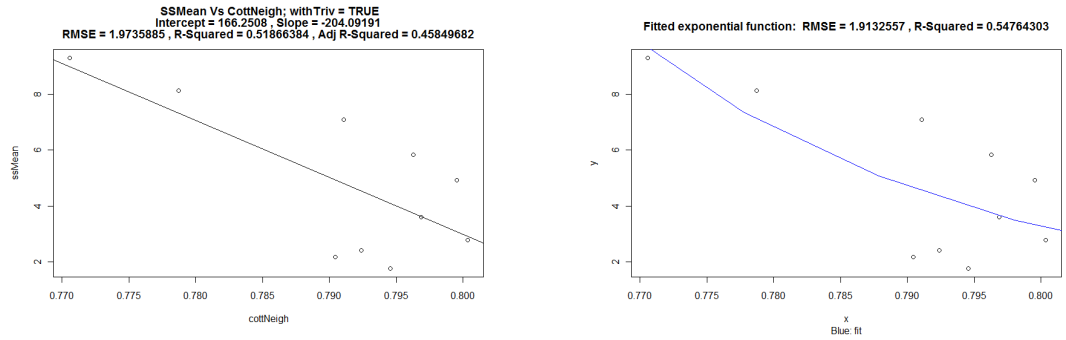

Figure 23: SSMean Plotted Against the proportion of non-random neighbors, cottNeigh with the outlier removed. Only maps trained for 500 or more iterations are considered. On the left, ssMean is plotted as a linear function of cottNeigh while on the right, ssMean is plotted as an exponential decay function of cottNeigh.

From Figures 22 and 23 it is clear that fConv tracks the SSMean more efficiently than cottNeigh in the limiting case. Again in Figure 23, we see that when cottNeigh indicates stability, the SSMean is not settled at all and takes on a wide range of values. However, when fConv indicates high levels of convergence (see Figure 7), the SSMean is stable and takes on a more compact range of values. Also, as fConv increases, SSMean decreases. With cottNeigh, there is no real correlation to SSMean in the limiting case. Also, fConv is a more conservative convergence measure, only indicating convergence after cottNeigh has indicated stability. Consider the following plot of both fConv and cottNeigh plotted as a function of iterations.

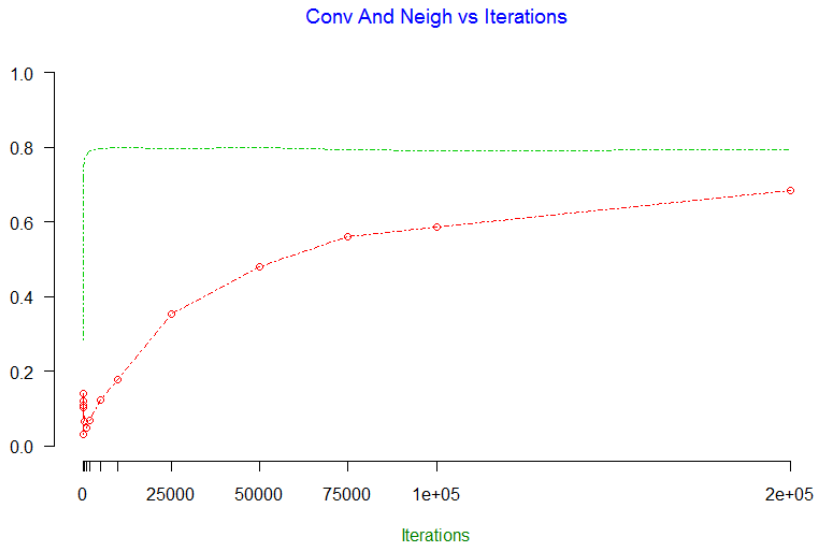

Figure 24: cottNeigh (green) and fConv (red) plotted as functions of iterations. It is clear that fConv levels out after cottNeigh and hence provides a more conservative measure of convergence.

In Figure 24 we can see that cottNeigh indicated convergence after only a few thousand iterations. However, fConv did not indicate complete convergence even after 200,000 iterations (the curve still had a fairly steep slope around 200,000 iterations and fConv was well short of one, which would indicate full convergence). If one were seeking a higher convergence level, several approaches could be used, including: training the map longer, increasing the training rate, or increasing the size of the map (thereby giving the map greater freedom to model the variance of the training data). Also, a lower convergence rate could indicate noise in the data. Removing some of the noise and re-training the data may result in a higher convergence score. One approach to removing noise involves expressing the training data in terms of its principal components and only keeping those components which have a significant variance [16]. Using only those components which have a variance greater than or equal to one is a common practice. The author was able to get an fConv value of one for the ionosphere data (i.e. 100% converged) by training

longer and using only the first few principal components of the training data to train the SOM.

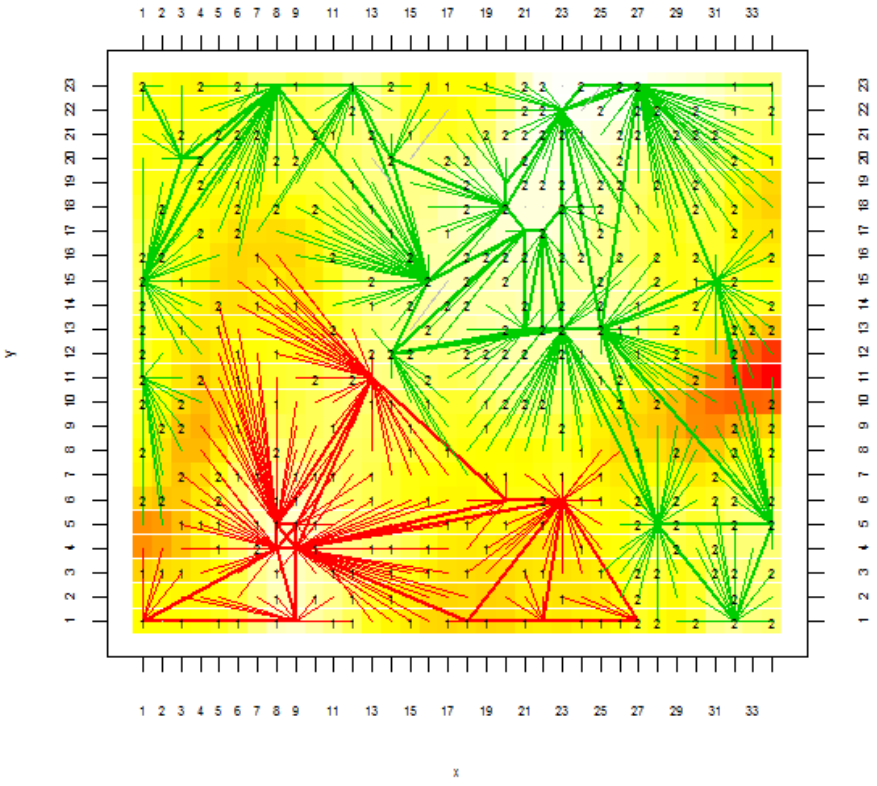

Figure 25: SOM achieving fConv = 1; trained for 375,000 iterations; noise reduction performed (first few principal components used to train the SOM)

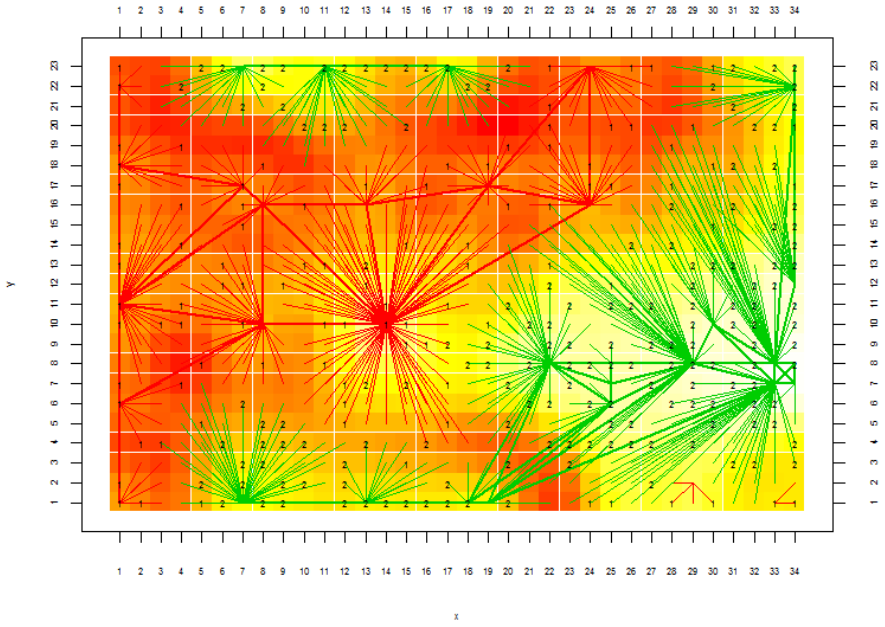

Figure 26: SOM achieving fConv = 0; trained for 15,100 iterations; noise reduction not performed; note that cottNeigh indicated stability after only 5,000-10,000 iterations even when no noise reduction was utilized on the data - this map should be stable according to cottNeigh

Note that when fConv indicates convergence, the clusters are contiguous whereas when cottNeigh indicates convergence, they are not. cottNeigh appears to indicate stability too early. The cluster representations in Figures 25 and 26 were created using a slightly modified version of the connected components approach as given in [22].

A second round of experimentation was performed on the ionosphere data set using 34×23 SOMs. This dimensionality is given by the $M = 2N$ method proposed earlier. However, the results were not qualitatively different from those reported using the 26×18 maps and hence were excluded.

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions

From the results presented in the previous chapter, we can conclude that fConv, the population based convergence measure proposed in this thesis, provides a reliable statistical measure of the "goodness" of a SOM. It also shows that the SOM is capable of modeling the probability densities of even very complex, noisy data sets. Since fConv is based on a simple two sample test, it is fast and easy to compute. fConv is more conservative as a measure of convergence than the approach using neighborhood stability which was proposed by Cottrell et al. In addition, fConv is a much less expensive method for determining how well a map models the training data since it can be computed on a map by map basis. It has no need for a bootstrapped collection of maps nor does it entail checking each pair of training data in order to determine if their neighborhood relationship is significant.

Cottrell's statistical analysis of neighborhood relationships may be useful in the case where one is trying to determine whether or not a given pair of training vectors are neighbors or not, and in either case, whether that relationship is significant. However, it does not appear to be useful as a measure for the overall convergence of a SOM. It is very expensive and is much more optimistic than the fConv method proposed in this thesis. In addition, fConv tracks the ssMean, a traditional measure of convergence, far better than cottNeigh, especially in the limiting case (which is the primary case that we are concerned with in a study pertaining to convergence).

fConv is a fast, effective method for determining how well a SOM has modeled the probability density of the training data. Its simplicity seems to increase its

appeal as a convergence measure since the SOM algorithm itself is quite simple. This also makes fConv preferable to other convergence methods which impose energy functions or modifications to the SOM which are much more complicated than the SOM algorithm itself. The results in this paper demonstrate that fConv is also correlated to the ssMean and tracks it well especially in the limiting cases. Hence, once an fConv value of one, or close to one is achieved, one would not expect to be able to decrease ssMean significantly with further training or an increase in the size of the map. In fact, attempting to decrease the ssMean in such a case would likely lead to an overfitted model. Hence, fConv may be used to determine when a given ssMean is "good enough". If fConv is much lower than one, then the map can either be trained longer, its size can be increased, or the learning rate can be increased in order to allow the map greater freedom to capture the variance in the training data. An fConv value much lower than one may also indicate that there is a substantial amount of noise in the data. In this case, a noise reduction technique can be applied to the training data before it is used to train the SOM.

6.2 Suggestions For Future Work

fConv provides a simple, elegant method for computing a measure indicative of how well a given SOM has modeled the probability density of the training data. However, it relies on both normalizing the data and rotating it into the direction of the principal components in order to increase its validity (since principal components are uncorrelated the PCA features can be treated separately and the cross terms in the covariance matrix can more justifiably be ignored). The normalization is strongly encouraged since the SOM is sensitive to scale. However, checking the variance convergence of each feature independently method does not account for the cross terms of the covariance matrix (which the SOM may not have modeled properly). While Yin and Allison's paper seems to indicate that this is not likely, it

may happen in practice, since SOMs are not trained for an infinite number of iterations and hence dead neurons will exist (Yin and Allison's paper required infinite training and no dead neurons). A future study could produce a test which tested the cross terms of the covariance matrix also and incorporated them into the convergence score. Cross terms may not be zero simply because a rotation occurred in a pair of dimensions which had similar variance (these do happen during SOM training, usually in cases where the two dimensions have a similar variance). The updated convergence measure would also penalize the convergence score less for rotations of this nature. If the whole covariance matrix is checked, one would also not have to rotate the training data into the directions of the principal components which would save some time up front.

Despite not accounting for cross terms of the covariance matrix, the fConv convergence measure proposed in this paper is demonstrably effective. Because training data expressed in terms of principal components is used to train the SOM, the SOM can model the data well. Most of the variance is usually captured in the first few principal components while the last few principal components usually just contain the noise in the data. The SOM will model the first two principal components with the greatest ease since it is a 2-dimensional grid, especially in the case where the $M = 2N$ sizing rule is used to determine the length and width of the map. Since the variance along the first component is usually much larger than the variance along the second principal component, rotations in these dimensions are rare in SOMs. Hence, fConv will at least capture these even in the case where cross terms are not examined. Generally the first few principal components of a high dimensional data set will contain most of the variance and the neurons of the SOM will not rotate along them unless the variance along some of the components are close in value. However, if the whole covariance matrix were

checked, one would not have to worry about the case where a rotation occurred. The convergence algorithm would penalize the convergence measure because some difference in the cross terms of the covariance matrix would be discovered.

LIST OF REFERENCES

[1] C. Hung, Y.-L. Chi, and T.-Y. Chen, "An attentive self-organizing neural model for text mining," *Expert Systems with Applications*, vol. 36, no. 3, Part 2, pp. 7064 – 7071, 2009.

[2] T. Kohonen, *Self-organizing maps*, ser. Springer series in information sciences. Springer, 2001.

[3] C. Bishop, M. Svensn, and C. Williams, "Gtm: A principled alternative to the self-organizing map," *Artificial Neural NetworksICANN 96*, pp. 165–170.

[4] J. J. Verbeek, N. Vlassis, and B. Krse, "The generative self-organizing map: a probabilistic generalization of kohonen's som," Tech. Rep., 2002.

[5] H. Kobayashi, B. Mark, and W. Turin, *Probability, Random Processes, and Statistical Analysis: Applications to Communications, Signal Processing, Queueing Theory and Mathematical Finance*. Cambridge University Press, 2011.

[6] T. Heskes, "Energy functions for self-organizing maps," *Kohonen maps*, p. 303316.

[7] E. Erwin, K. Obermayer, and K. Schulten, "Self-organizing maps: ordering, convergence properties and energy functions," *Biological cybernetics*, vol. 67, no. 1, pp. 47–55, 1992.

[8] E. De Bodt, M. Cottrell, and M. Verleysen, "Statistical tools to assess the reliability of self-organizing maps," *Neural Networks*, vol. 15, no. 8-9, p. 967978, 2002.

[9] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. New York: Chapman & Hall, 1993.

[10] H. Yin and N. M. Allinson, "On the distribution and convergence of feature space in self-organizing maps," *Neural computation*, vol. 7, no. 6, pp. 1178–1187, 1995.

[11] "Covariance matrix - wikipedia, the free encyclopedia." Jan. 2012. [Online]. Available: http://en.wikipedia.org/wiki/Covariance_matrix

[12] J. B. Fraleigh and R. A. Beauregard, *Linear Algebra*. Massachusetts: Addison-Wesley Publishing Company, 1995.

[13] "Principal component analysis - wikipedia, the free encyclopedia." Jan. 2012. [Online]. Available: http://en.wikipedia.org/wiki/Principal_component_analysis

[14] I. Miller and M. Miller, *John E. Freund's Mathematical Statistics with Applications (7th Edition)*, 7th ed. Prentice Hall, 2003.

[15] "Normally distributed and uncorrelated does not imply independent - wikipedia, the free encyclopedia." Feb. 2012. [Online]. Available: http://en.wikipedia.org/wiki/Normally_distributed_and_uncorrelated_does_not_imply_independent

[16] L. Smith, "A tutorial on principal components analysis," *Cornell University, USA*, vol. 51, p. 52, 2002.

[17] M. Cottrell, E. De Bodt, and M. Verleysen, "A statistical tool to assess the reliability of self-organizing maps," *Advances in self-organising maps*, p. 714, 2001.

[18] D. Ballard, *An Introduction to Natural Computation*. Massachusetts: MIT Press, 1999.

[19] "UCI machine learning repository: Iris data set." Feb. 2012. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Iris

[20] "UCI machine learning repository: Wine data set." Feb. 2012. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Wine

[21] "UCI machine learning repository: Ionosphere data set." Feb. 2012. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Ionosphere

[22] L. Hamel and C. Brown, "Improved interpretability of the unified distance matrix with connected components."

BIBLIOGRAPHY

"Self-organizing map - wikipedia, the free encyclopedia." Dec. 2011. [Online]. Available: http://en.wikipedia.org/wiki/Self-organizing_map

"Coefficient of variation - wikipedia, the free encyclopedia." Feb. 2012. [Online]. Available: http://en.wikipedia.org/wiki/Coefficient_of_variation

"Covariance matrix - wikipedia, the free encyclopedia." Jan. 2012. [Online]. Available: http://en.wikipedia.org/wiki/Covariance_matrix

"Generative topographic map - wikipedia, the free encyclopedia." Feb. 2012. [Online]. Available: http://en.wikipedia.org/wiki/Generative_topographic_map

"LinAlgReviewAndSpectral.pdf." Feb. 2012. [Online]. Available: http://web.williams.edu/go/math/sjmiller/public_html/OSUClasses/683L/LinAlgReviewAndSpectral.pdf

"MVA_Section3.pdf." Feb. 2012. [Online]. Available: http://www.maths.manchester.ac.uk/~mkt/MT3732%20(MVA)/Notes/MVA_Section3.pdf

"Normally distributed and uncorrelated does not imply independent - wikipedia, the free encyclopedia." Feb. 2012. [Online]. Available: http://en.wikipedia.org/wiki/Normally_distributed_and_uncorrelated_does_not_imply_independent

"Principal component analysis - wikipedia, the free encyclopedia." Jan. 2012. [Online]. Available: http://en.wikipedia.org/wiki/Principal_component_analysis

"Principal components and factor analysis." Jan. 2012. [Online]. Available: http://www.statsoft.com/textbook/principal-components-factor-analysis/

"Statistics part 3: Covariance and correlation." Jan. 2012. [Online]. Available: http://investing.calsci.com/statistics3.html

"UCI machine learning repository: Ionosphere data set." Feb. 2012. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Ionosphere

"UCI machine learning repository: Iris data set." Feb. 2012. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Iris

"UCI machine learning repository: Wine data set." Feb. 2012. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Wine

Ballard, D., *An Introduction to Natural Computation.* Massachusetts: MIT Press, 1999.

Bilodeau, M. and Brenner, D., *Theory of multivariate statistics*, ser. Springer texts in statistics. Springer, 1999.

Bishop, C., Svensn, M., and Williams, C., "Gtm: A principled alternative to the self-organizing map," *Artificial Neural NetworksICANN 96*, pp. 165–170.

Bishop, C., Svensn, M., and Williams, C., "Developments of the generative topographic mapping," *Neurocomputing*, vol. 21, no. 1-3, p. 203224, 1998.

Cottrell, M., De Bodt, E., and Verleysen, M., "A statistical tool to assess the reliability of self-organizing maps," *Advances in self-organising maps*, p. 714, 2001.

De Bodt, E., Cottrell, M., and Verleysen, M., "Statistical tools to assess the reliability of self-organizing maps," *Neural Networks*, vol. 15, no. 8-9, p. 967978, 2002.

Efron, B. and Tibshirani, R. J., *An Introduction to the Bootstrap.* New York: Chapman & Hall, 1993.

Erwin, E., Obermayer, K., and Schulten, K., "Self-organizing maps: ordering, convergence properties and energy functions," *Biological cybernetics*, vol. 67, no. 1, pp. 47–55, 1992.

Fraleigh, J. B. and Beauregard, R. A., *Linear Algebra.* Massachusetts: Addison-Wesley Publishing Company, 1995.

Hamel, L. and Brown, C., "Improved interpretability of the unified distance matrix with connected components."

Heskes, T., "Energy functions for self-organizing maps," *Kohonen maps*, p. 303316.

Hung, C., Chi, Y.-L., and Chen, T.-Y., "An attentive self-organizing neural model for text mining," *Expert Systems with Applications*, vol. 36, no. 3, Part 2, pp. 7064 – 7071, 2009.

Kinouchi, M., Takada, N., Kudo, Y., and Ikemura, T., "Quick learning for batch-learning self-organizing map," *GENOME INFORMATICS SERIES*, p. 266267, 2002.

Kobayashi, H., Mark, B., and Turin, W., *Probability, Random Processes, and Statistical Analysis: Applications to Communications, Signal Processing, Queueing Theory and Mathematical Finance.* Cambridge University Press, 2011.

Kohonen, T., *Self-organizing maps*, ser. Springer series in information sciences. Springer, 2001.

Laaksonen, J. T., Markus Koskela, J., and Oja, E., "Class distributions on som surfaces for feature extraction and object retrieval," *Neural Networks*, vol. 17, no. 8-9, pp. 1121–1133, 2004.

Lampinen, J. and Kostiainen, T., "Generative probability density model in the self-organizing map," *STUDIES IN FUZZINESS AND SOFT COMPUTING*, vol. 78, pp. 75–94, 2002.

Lin, S. and Si, J., "Weight-value convergence of the som algorithm for discrete input," *Neural computation*, vol. 10, no. 4, pp. 807–814, 1998.

Miller, I. and Miller, M., *John E. Freund's Mathematical Statistics with Applications (7th Edition)*, 7th ed. Prentice Hall, 2003.

Smith, L., "A tutorial on principal components analysis," *Cornell University, USA*, vol. 51, p. 52, 2002.

Ultsch, A., "Self-organizing neural networks for visualisation and classification," in *Information and classification: concepts, methods, and applications: proceedings of the 16th Annual Conference of the" Gesellschaft für Klassifikation eV," University of Dortmund, April 1-3, 1992*, vol. 16. Springer Verlag, 1993, p. 307.

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

Verbeek, J. J., Vlassis, N., and Krse, B., "The generative self-organizing map: a probabilistic generalization of kohonen's som," Tech. Rep., 2002.

Yin, H. and Allinson, N. M., "On the distribution and convergence of feature space in self-organizing maps," *Neural computation*, vol. 7, no. 6, pp. 1178–1187, 1995.