Brief Review of `Self-Organizing Maps`

Manan Sharma

1 Introdution

Among numerous neural network architectures, particularly interesting architecture was introduced by Finish Professor Teuvo Kohonen in the 1980s. Self-organizing map (SOM), sometimes also called a Kohonen map use unsupervised, competitive learning to produce low dimensional, discretized representation of presented high dimensional data, while simultaneously preserving similarity relations between the presented data items. Such low dimensional representation is called a feature map, hence map in the name. Feature maps generated by the SOFM algorithm are characterized by the fact that weight vectors which are neighbors in the input space are mapped onto neighboring neurons. If the dimensionalities of the input space and the network differ, it is impossible to preserve all similarity relationships among weight vectors in the input space; only the most "important" similarity relationships are preserved and mapped onto neighborhood relationships on the network of neurons, while the less "important" similarity relationships are not retained in the mapping. If the input space and network are of the same dimensionality, the SOFM algorithm can preserve all the similarity relationships and generates a distorted, but topographic map, of the input space, where more "important" regions of the input space are represented with higher resolution.
The low-dimensional, ordered representation of data generated by the SOFM algorithm has proven useful for a variety of technical applications in the areas of pattern classification and function approximation as well as knowledge representation (Ritter and Kohonen 1989; Scholtes 1991), and the algorithm has been successfully applied as a model for the development of structural representations in biological neural systems, the so-called brain maps.
The formation of topologically ordered feature maps occurrs easily in a wide range of situations, where the dimensionalities of the input and image spaces are the same or different and where the interaction function, or neighborhood function, takes a variety of forms.

This brief review report attempts to introduce a reader to SOMs, covering in short basic tenets, underlying biological motivation, its architecture, math description, properties of feature maps and then finally conclusion. The results of SOM depend on some initialization and learning parameters. In this experiment , five neighboring functions (bubble, Gaussian, heuristic, Epanechikov and Cutgauss) and three learning rates (linear, inverse-of-time and power series) are investigated. In the training process, learning rates can be changed in two ways (epochs and iterations). The dependence of these ways on the results is researched, too. The quality of SOM is commonly estimated by quantization and topographic errors. The main goal of the research is to estimate dependences of learning parameters on the results obtained by SOM in the sense of quantization error. Experiments have been carried out with six synthetic data sets (Spiral, Double Moon, Eliptical, Gaussian Mixture, Triangular 3 class and Corner) and 3 sparse data sets (Zoo, Glass and Wine) .

2 Motivation

After millions of years of evolution, brain in animals and humans has evolved into the massive parallel stack of computing power capable of dealing with the tremendous varieties of situations it can encounter. The biological neural networks are natural intelligent information processors. Artificial neural networks (ANN) constitute computing paradigm motivated by the neural structure of biological systems. ANNs employ a computational approach based on a large collection of artificial neurons that are much simplified representation of biological neurons. Synapses that ensure communication among biological neurons are replaced with neuron input weights. Adjustment of connection weights is performed by some of numerous learning algorithms. ANNs have very simple principles, but their behavior can be very complex. They have a capability to learn, generalize, associate data and are fault tolerant.

3 Self-Organization Map (SOM)

This neural network, or called Kohonen is a popular neural network, which is based on unsupervised learning. The structure of SOM is a single feed forward ,it has a set of input (x1, x2,...., xn) elements and a set of output unit (w1,w2,...., wn) where each input element is connected to all output neurons. SOM network uses competition concept that tries to find the most similar distance between the input vector (Xi) and neuron with weight vector (Wi),the architecture as shown in figure (1).

The learning process includes the following steps.
1- Define and initialize number of parameters, which are network size learning iteration (t), radius of neighborhood (d0), learning rate, and total number of iterations (T). At this stage, the user experience plays a major role in defining these parameters to achieve the best results.
2- Initialize all node's weight W_{ij} with small random values in the same dimensionality as the training patterns.
3- Find the winner node or called the best matching unit BMU by computing Euclidean distance between them.

$$Dist = \sqrt[2]{\sum_{i=1}^{n}(X_i - W_i)^2}$$

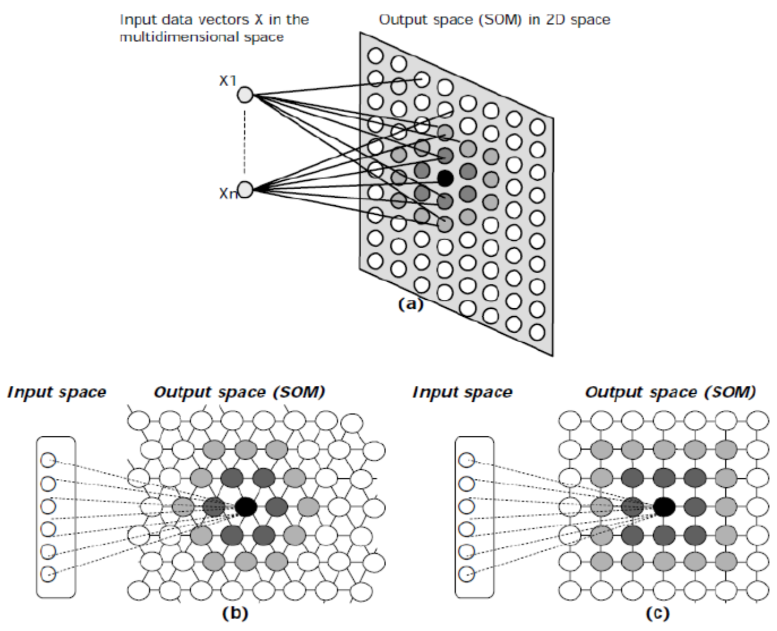

Input data vectors X in the multidimensional space
Output space (SOM) in 2D space
X1
Xn
(a)
Input space
Output space (SOM)
(b)
Input space
Output space (SOM)
(c)

Figure 1: SOM Network Structure output space (2-D) and input space (1-D)

Where X is the current input data vector and W is the weight of nodes.
4- Update all the unit weights depending on the distance in the output space between each unit and the BMU, the distance in the input space between the unit and the training pattern.
The following formulas are used to update weight vectors and radius of both the winner neuron j and its neighbors as:

$$\sigma(t) = \sigma 0 \exp(-t/\lambda)$$
$$W(t+1) = W(t) + \alpha(t,T) h_{ci}(t)(V(t) - W(t))$$

Where $\sigma 0$ represents the width of the grid at time t0 and the λ denotes a time constant, t is the current time-step. $\alpha(t.T)$ is a small variable called the learning rate, $\alpha(t.T)$ and $\sigma(t)$ both decreases with time and $h_{ci}(t)$ is a neighborhood function.
Both the learning rate and the neighborhood radius should converge to zero to reach stable solutions in SOM. Further, both parameters must be updating after presenting each individual data pattern to the network (iteration).

3.1 SOM Parameters

Before the SOM training process, many factors related to SOM structure that affect the result must be define.These include: the size, topology, and shape of the used map. In addition to, the training parameters (number of iterations, initial learning rate, learning function, initial neighborhood radius and neighborhood function).

3.1.1 Size and Dimension of the Map

The size of the problem at hand determines the SOM size and dimension, which makes this definition mainly an empirical process.SOM's size must also take into consideration the size of the dataset of training patterns and the number of units should be equal to the number of expected clusters. Therefore each cluster should be represented by a single unit.Each unit is a cluster centroid .The SOM output space dimensionality reflects the number of units used in each dimension (x, y, z, etc.). For instance in a two-dimensional SOM, using x equal to 10, y equal to 10 produces a network with 100 units.

3.1.2 Topology, shape and initialization

SOM uses two basic types of topology square and hexagonal. In square topology, each unit is connected to its four neighbors, while in hexagonal topology each unit is connected to its six neighbors. Figure (2) shows type of topology where the dark gray units represent neighbors of the black units. Unit vector must be initialized before the training phase , proper initialization allows SOM converges faster to a good solution.

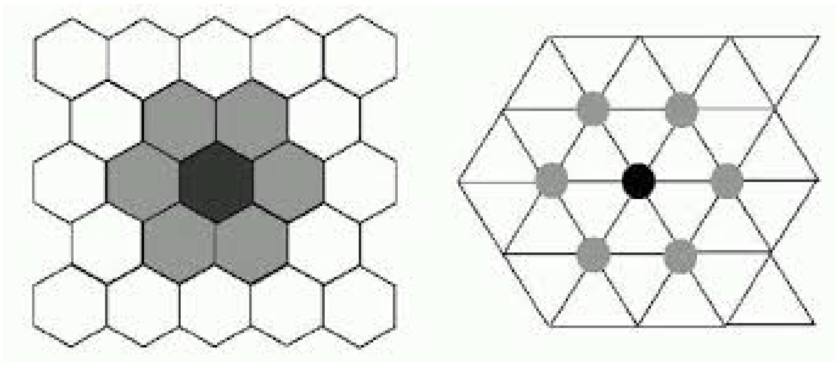


Figure 2: Hexagonal topology with six neighbors

3.1.3 Number of iterations

The number of iteration (usually called epochs) is important parameter that defines how long training phase in SOM. One epoch is achieved when all the training patterns were presented once to the network. There is no rule about selecting optimal number of iterations to use. However the training phase of the SOM should be large enough to fit with the dataset to reach a good solutions.

3.1.4 Topographic Maps

Neurobiological studies indicate that different sensory inputs (motor, visual, auditory, etc.) are mapped onto corresponding areas of the cerebral cortex in an orderly fashion. This form of map, known as a topographic map, has two important properties:
1. At each stage of representation, or processing, each piece of incoming information is kept in its proper context/neighbourhood.
2. Neurons dealing with closely related pieces of information are kept close together so that they can interact via short synaptic connections.
Our interest is in building artificial topographic maps that learn through self-organization in a neurobiologically inspired manner.
We shall follow the principle of topographic map formation : "The spatial location of an output neuron in a topographic map corresponds to a particular domain or feature drawn from the input space".

3.1.5 Learning rate and learning functions

Learning rate is the initial value given by the user which is usually between (0,1), and gradually decreases during the training phase. Several different functions can be used to control the decreasing of learning rate. Linear, power of series , inverse of time (INV) and heuristic are major learning rate functions as shown in figure (3).

$$Inverse\ of\ Time \qquad \alpha(t,T) = \alpha(0)(1-\frac{t}{T})$$

$$linear \qquad \alpha(t) = \alpha(0)(\frac{1}{t})$$

$$Power\ Series \qquad \alpha(t,T) = \alpha(0)\exp\frac{t}{T}$$

$$Heuristic \qquad \alpha(t,T) = max(\frac{T+1-t}{T}, 0.01)$$

3.1.6 Neighborhood radius and neighborhood functions

Neighborhood function (h) is a value between (0, 1), and it is a position function of two units (BMU and another unit) with a given radius r, it usually decreases with time. BMU has a high value and decreases as distance increases . Neighborhood radius (r) can have value between (0,1) (when only BMU updated) and maximum size of the network (when all units will be updated). There are several neighborhood functions, such as Bubble, Gaussian , Cutgauss and ep(or Epanechikov function) .Figure (4) shows the four neighborhood functions (Bubble, Gaussian , Cutgauss and ep (or Epanechikov function). respectively) in one dimension.

$$Bubble \qquad h_{ci}(t) = F(\sigma_i - d_{ci})$$

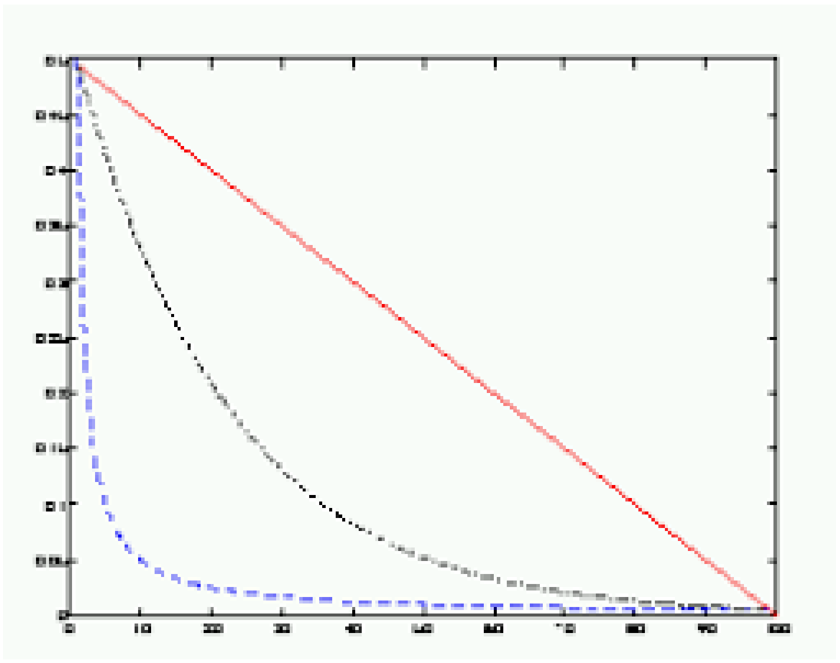


Figure 3: Learning Rate Functions

$$\begin{array}{ll} \textit{Gaussian} & h_{ci}(t) = \exp(\frac{-d_{ci}}{2\sigma_i^2}) \\ \textit{Cutgauss} & h_{ci}(t) = \exp(\frac{-d_{ci}}{2\sigma_i^2})F(\sigma_i - d_{ci}) \\ \textit{Epanechikov} & h_{ci}(t) = max(0, 1 - (\sigma_i - d_{ci})^2) \\ \textit{Heuristic} & h_{ci}(t) = \frac{1}{\alpha(t).\sigma_i + 1} \end{array}$$

Where σ_t is the neighborhood radius at time t, and $d_{ci} = |r_c - r_i|$ is the distance between units c and i in the output space and $\alpha(t)$ is learning rate and F is a step function such that,

$$F(x) = 0 \quad if \qquad x < 0$$

$$F(x) = 1 \quad if \qquad x \geq 0$$

Basically, Neighborhood function determines the rate of change of the neighborhood around the winner neuron. Neighborhood function influences the training result of SOM procedure. Therefore, it is important to choose the proper neighborhood function with the data set. Same as learning rate, there are many functions but bubble and Gaussian are widely used in SOM. Bubble function is a constant function while Gaussian function is decreasing function in the defined neighborhood of the winner neuron.

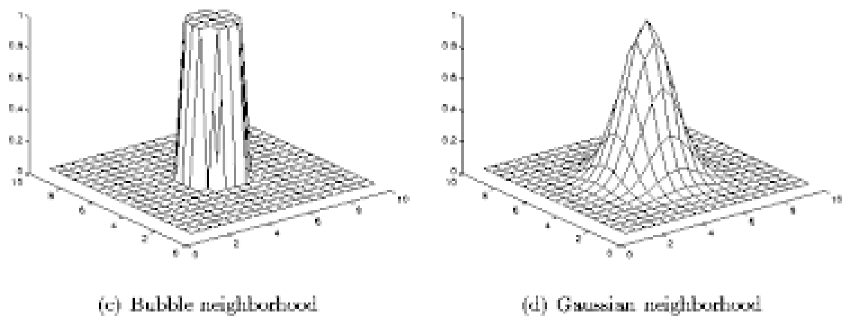


Figure 4: Bubble Function in 2-D

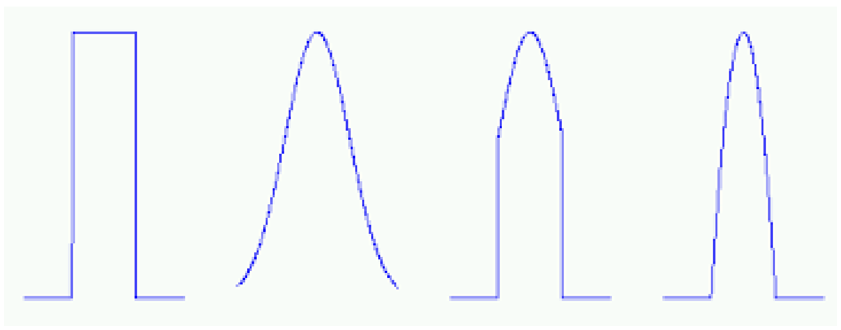


Figure 5: The three neighborhood functions: Bubble, Gaussian Cutgass and Epanechikov in 1-D

3.2 Overview of SOM Algorithm

Following steps represent the SOM algorithm:
Step 1: Initialize the neuron weights (iteration step n=0)
Step 2: Randomly sample a vector x(n) from the datset.
Step 3: Compute the distance for all input data and weight.
Step 4: Find the BMU.
Step 5: Update the neighborhood function using neighborhood equations and learning rate function by using learning rate equations , update the BMU weight and its neighbors by using weight update equation.
Step 6: Clustering- Find the best matching unit (BMU) .Test pattern x belongs to the cluster represented by the best matching unit c.
Step 7: Repeat steps 3-6 for all input data.

3.3 Important Properties of SOMs

Once the SOM algorithm has converged, the feature map displays important statistical characteristics of the input space. Given an input vector x, the feature map ϕ provides a winning neuron I(x) in the output space, and the weight vector $W_I(x)$ provides the

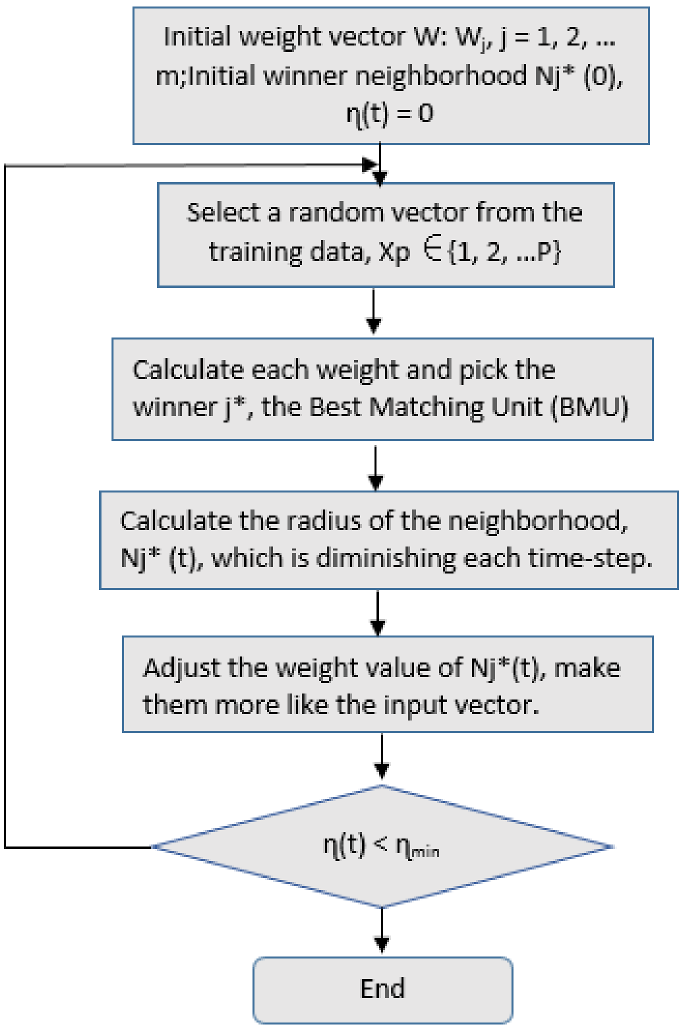
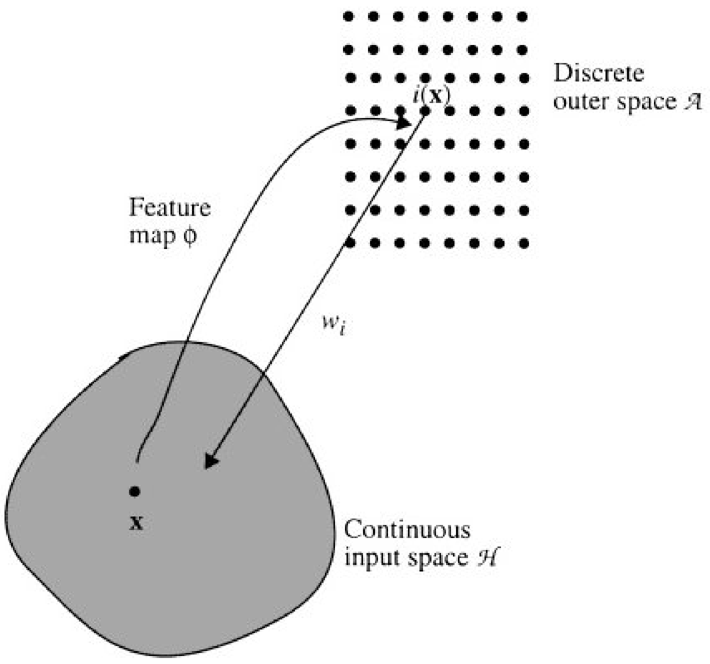

Initial weight vector W: W_j, j = 1, 2, … m;Initial winner neighborhood Nj* (0), η(t) = 0
Select a random vector from the training data, Xp ∈{1, 2, …P}
Calculate each weight and pick the winner j*, the Best Matching Unit (BMU)
Calculate the radius of the neighborhood, Nj* (t), which is diminishing each time-step.
Adjust the weight value of Nj*(t), make them more like the input vector.
η(t) < $η_{min}$
End

Figure 6: Flowchart of model

coordinates of the image of that neuron in the input space. Thus, as depicted in given figure, the SOM algorithm embodies two ingredients that define the algorithm:
• A projection from the continuous input data space x onto the discrete output neural space a . In this way, an input vector is mapped onto a "winning neuron" in the lattice structure in accordance with the similarity matching step as discussed in the algorithm.
• A pointer from the output space back to the input space. In effect, the pointer defined by the weight vector of the winning neuron identifies a particular point in the input data space as the "image" of the winning neuron; this operation is iteratively carried out in accordance with the updating step .



In other words, there is communication, back and forth, between the output space where the lattice of neurons resides and the input space where the examples are generated. The SOM algorithm has some important properties that are discussed next.

3.3.1 Approximation of the Input Space

he feature map represented by the set of weight vectors W_i in the output space, provides a good approximation to the input space.

We can state the aim of the SOM as storing a large set of input vectors x by finding a smaller set of prototypes W_i so as to provide a good approximation to the original input space. In effect, the goodness of the approximation is given by the total squared distance

$$D = \sum_x ||(X - W_{I(x)})||^2$$

which we wish to minimize. If we work through gradient descent style mathematics we do end up with the SOM weight update algorithm, which confirms that it is generating a good approximation to the input space.

3.3.2 Topological Ordering

The feature map ϕ computed by the SOM algorithm is topologically ordered in the sense that the spatial location of a neuron in the output lattice/grid corresponds to a particular domain or feature of the input patterns.

The topological ordering property is a direct consequence of the weight update equation that forces the weight vector $W_{I(x)}$ of the winning neuron I(x) to move toward the input vector x. The crucial factor is that the weight updates also move the weight vectors W_j of the closest neighbouring neurons j along with the winning neuron I(x). Together these weight changes cause the whole output space to become appropriately ordered.

Each output node can be represented in the input space at coordinates given by their weights. Then if the neighbouring nodes in output space have their corresponding points in input space connected together, the resulting image of the output grid reveals directly the topological ordering at each stage of the network training.

3.3.3 Density Matching

The feature map ϕ reflects variations in the statistics of the input distribution: regions in the input space from which the sample training vectors x are drawn with high probability of occurrence are mapped onto larger domains of the output space, and therefore with better resolution than regions of input space from which training

We need to relate the input vector probability distribution p(x) to the magnification factor m(x) of the feature map. Generally, for two dimensional feature maps the relation cannot be expressed as a simple function, but in one dimension we can show that

$$m(x) \propto p^{\frac{2}{3}}(x)$$

So the SOM algorithm doesn't match the input density exactly, because of the power of 2/3 rather than 1. Computer simulations indicate similar approximate density matching in general, always with the low input density regions slightly over-represented.

3.3.4 Feature Selection

Given data from an input space with a non-linear distribution, the self organizing map is able to select a set of best features for approximating the underlying distribution.

This property is a combination of Approximation of the Input Space, Topological Ordering, Density Matching. Principal Component Analysis (PCA) is able to compute the input dimensions which carry the most variance in the training data. It does this by computing the eigenvector associated with the largest eigenvalue of the correlation matrix. PCA is fine if the data really does form a line or plane in input space, but if the data forms a curved line or surface (e.g. a semi-circle), linear PCA is no good, but a SOM will overcome the approximation problem by virtue of its topological ordering property. The SOM provides a discrete approximation of finding so-called principal curves or principal surfaces, and may therefore be viewed as a non-linear generalization of PCA.

4 Experimental Results

4.1 Data Set Analyzed

Six different synthetic data sets are used in the experimental investigation. Corner data set which comprises of two features and four classes .Two-dimensional vectors $X_1, X_2, X_3.....X_{1000}$ where $X_i = (X_{i1}, x_{i2})$, i = 1 ,..., 1000 . Two features are measured with 4 different classes.
Similarly , Double Moon Data set is used which comprises of two features and two different classes. In the Similar way rest of the data sets are analyzed . Triangular 3 class dataset which comprises of 3 different classes. Some kind of Sparse dataset are also investigated. Three data sets are used in the experimental investigation. The glass data set was collected by a scientist, which wanted to help criminalists to recognize glass slivers found. Nine-dimensional vector $X_1, X_2, ..., X_{214}$ are formed, where $X_i = (X_{i1}, x_{i2}....x_{i9})$ where , i = 1 ,..., 214 . Nine features are measured: x_1 is a refractive index,x_2 is sodium,x_3 is magnesium,$x_4$is aluminum, x_5 is silicon,x_6 is potassium,$x_7$is calcium,$x_8$is barium andx_9 is iron.
The wine data set consists of the results of chemical analysis of wine grapes grown in the same region in Italy but derived from three different cultivators. The analysis determined the quantities of 13 constituents found in each of the three types of wines. Thirteen-dimensional vectors $X_1, X_2, ..., X_{178}$ are formed, where $X_i = (X_{i1}, x_{i2}....x_{i13})$, i = 1 ,..., 178 . Thirteen features are measured: x_1 is alcohol,x_2 is malic acid,x_3 is ash, x_4 is alcalinity of ash, x_5 is magnesium,x_6 is total phenol, $x_7$is flavanoid, x_8 is non flavanoid phenol,x_9 is proanthocyanin, x_{10} is color intensity,x_{11} is hue, x_{12} is OD280/OD315 of diluted wines,x_{13} is proline.
The third data set is zoological (zoo). The data set consists of 16 boolean values. $X_1, X_2, ..., X_{92}$ are formed, where Sixteen-dimensional vectors $X_i = (X_{i1}, x_{i2}....x_{i16})$, i = 1 ,..., 92 . Sixteen features are measured: x_1 is hair, x_2 is feathers,x_3 is eggs, x_4 is milk,x_5 is airborne,$x_6$is aquatic,x_7 is a predator,x_8 is toothed,x_9 is a backbone,x_{10} is breathes,x_{11} is venomous, x_{12} is a fin,x_{13} is legs, x_{14} is a tail,x_{15} is domestic, andx_{16} is catsize. Images for different datasets which are used in this experiment are shown in Appendix.

4.1.1 Dependence of Results on the Learning Parameters and Neighborhood functions

The SOM with three neighboring functions like Bubble , Gaussian , Cutgauss, Epanechikov , Heuristic and different learning rates like Inverse of Time, Linear, Power Series, Heuristic functions are used to check how much different parameters affect the SOM training results.
All initial parameters of SOM are shown in Table I.

Table 1: THE INITIAL PARAMETERS SETTING OF SOM

Parameters	**Values**
Initial learning rate	0.5
Initial Weight Vector	Random
Initial Lattice Radius	3
Number of Epochs	1000
SOM Dimesnion	7×7

4.1.2 Quantization Error

Quantization error (QE) is a technique to measure the quality of SOM. It is computed from the average distance of input vectors, x to the weight vector on the winner node (W_i^c) of the BMU. A SOM with lower average error is more accurate than a SOM with higher average error. Quantization error is calculated by

$$QE = \frac{\sum_{i=1}^{n}(||X_i - W_i^c||)}{N}$$

where N is the number of input vectors used to train the map.

5 Results

We can examine the quality of SOM by calculating Quantization Error.
In case of Corner data set Cutgauss neighborhood function gives the finest result . Data set was tested on three different learning rate functions and five different neighborhood functions.
The results from different experiment cases of five neighborhood functions and three learning rates to classify Corner data set are shown in Table 2. From the coming results we can see that Cutgauss neighborhood function is working better in comparison to different neighborhood functions in case of Corner data set.
Similarly , Eliptical Data set works well with Gaussian Neighborhood function . The results for different neighborhood functions are depicted in Table 3 .
In case of Triangular 3 class data set, Gaussian function with Inverse of time learning

rate function gives the good result. Errors are depicted in Table 4.
Gaussian Mixture data set works well with Bubble function. Results are depicted in Table 5. In case of Spiral dataset Cutgauss neighborhood function with power series as a learning rate gives the good result. Errors for different functions are depicted in Table 5. Double Moon data set works well with Gaussian function , results are depicted in Table 6.
In case of Sparse data set like wine data set gives best result with gaussian function , results are depicted in Table 7. Similarily Glass and Zoo Dataset gives good result with Gaussian neighborhood function when non linear learning rate functions are used. The reults are depicted in Table 8 and Table 9.

Table 2: Averages of the quantization errors for the corner data set. $X_i = (x_{i1}, x_{i2})$

Learning Rate	**Bubble**	**Gaussian**	**Heuristic**	**Epanechikov**	**Cutgauss**
Linear	0.0442	0.03913	0.0523	0.0762	0.0392
Inverse of Time	0.0423	0.0344	0.0483	0.0684	0.0316
Power Series	0.0401	0.0382	0.0446	0.0741	0.0381

Table 3: Averages of the quantization errors for the Eliptical data set. $X_i = (x_{i1}, x_{i2})$

Learning Rate	**Bubble**	**Gaussian**	**Heuristic**	**Epanechikov**	**Cutgauss**
Linear	0.0741	0.0422	0.1024	0.123	0.0983
Inverse of Time	0.0692	0.0344	0.183	0.1184	0.1021
Power Series	0.0721	0.0342	0.1006	0.1241	0.8297

Table 4: Averages of the Quantization errors for the Triangular 3 class data set . $X_i = (x_{i1}, x_{i2})$

Learning Rate	**Bubble**	**Gaussian**	**Heuristic**	**Epanechikov**	**Cutgauss**
Linear	0.0984	0.0422	0.1024	0.1231	0.01983
Inverse of Time	0.0994	0.0282	0.0920	0.0987	0.0199
Power Series	0.0712	0.0264	0.1106	0.0998	0.0202

Table 5: Averages of the Quantization errors for the Gaussian Mixture data set . $X_i = (x_{i1}, x_{i2})$

Learning Rate	**Bubble**	**Gaussian**	**Heuristic**	**Epanechikov**	**Cutgauss**
Linear	0.0401	0.0422	0.1024	0.1987	0.1983
Inverse of Time	0.03892	0.0501	0.1320	0.1384	0.0821
Power Series	0.02921	0.04822	0.1206	0.123	0.09492

Table 6: Averages of the Quantization errors for the Spiral data set . $X_i = (x_{i1}, x_{i2})$

Learning Rate	**Bubble**	**Gaussian**	**Heuristic**	**Epanechikov**	**Cutgauss**
Linear	0.1801	0.0932	0.1698	0.2987	0.04876
Inverse of Time	0.1567	0.09831	0.1320	0.1583	0.04321
Power Series	0.1492	0.0892	0.1906	0.2321	0.03876

Table 7: Averages of the Quantization errors for the Double Moon data set . $X_i = (x_{i1}, x_{i2})$

Learning Rate	**Bubble**	**Gaussian**	**Heuristic**	**Epanechikov**	**Cutgauss**
Linear	0.1832	0.0489	0.1982	0.2987	0.05876
Inverse of Time	0.1426	0.0431	0.1650	0.1583	0.04321
Power Series	0.1432	0.0413	0.1806	0.2321	0.04876

Table 8: Averages of the Quantization errors for the wine data set . $X_i = (x_{i1}, x_{i2}..x_{i13})$

Learning Rate	**Bubble**	**Gaussian**	**Heuristic**	**Epanechikov**	**Cutgauss**
Linear	0.1359	0.149	0.2169	0.2012	0.152
Inverse of Time	0.1736	0.128	0.1946	0.1583	0.1342
Power Series	0.1894	0.152	0.1928	0.1921	0.1632

For each observation there is a marker placed on the position of the winning neuron on the map. Each type of marker represents a class of the data set. The average distance map of the weights is used as background (see the color bar on the right to associate the value).
The results of the glass data set are presented in Fig. 16, 17 and 18 .In figures, only the numbers of epochs are presented. The number of epochs multiplied by the number of data items m corresponds to the number of iterations. For the glass data set (m=214), 10 epochs equal to 2140 iterations, 20 epochs equal 4280, etc. The worst results in the

Table 9: Averages of the Quantization errors for the Glass data set . $X_i = (x_{i1}, x_{i2}..., x_{i9})$

Learning Rate	**Bubble**	**Gaussian**	**Heuristic**	**Epanechikov**	**Cutgauss**
Linear	0.1805	0.2275	0.2394	0.2631	0.01837
Inverse of Time	0.1457	0.1563	0.1912	0.2013	0.1662
Power Series	0.1980	0.1688	0.1830	0.1982	0.1837

Table 10: Averages of the Quantization errors for the Zoo data set . $X_i = (x_{i1}, x_{i2}, ..., x_{i16})$

Learning Rate	**Bubble**	**Gaussian**	**Heuristic**	**Epanechikov**	**Cutgauss**
Linear	0.1787	0.1886	0.2111	0.231	0.1932
Inverse of Time	0.181	0.114	0.138	0.1902	0.1232
Power Series	0.212	0.126	0.144	0.1837	0.1532

sense of quantization error are obtained, if learning rate (4) is used. In this case, the way, when the number of epochs is used in changes of the learning rate, gives better results in the sense of quantization error comparing with the results, obtained when the number of iterations is used in changes of the learning rate. When we use learning rate (5), Gaussian function (3) gives the best results, independent of the ways of changes of learning rates (by epochs or iterations). Learning rates (6) and (8) give the best results with the Gaussian function, but heuristic neighborhood function (7) with epochs also gives good results.
Learning Rate(4)—Linear
Learning Rate(5)—Inverse of Time
Learning Rate(6)—Power Series

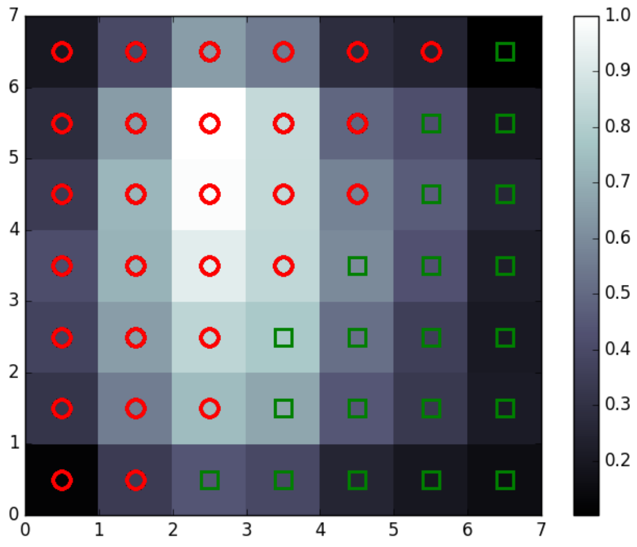


Figure 7: Double Moon dataset in 7×7 SOM : Gaussian Neighborhood function is used

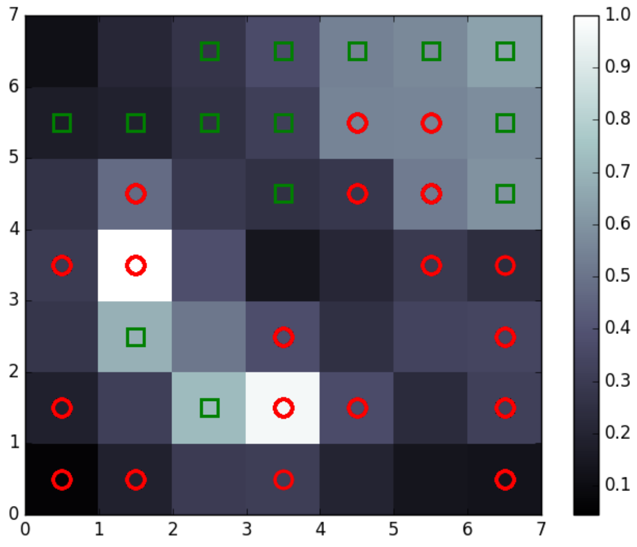


Figure 8: Double Moon dataset in 7×7 SOM : Bubble Neighborhood function is used

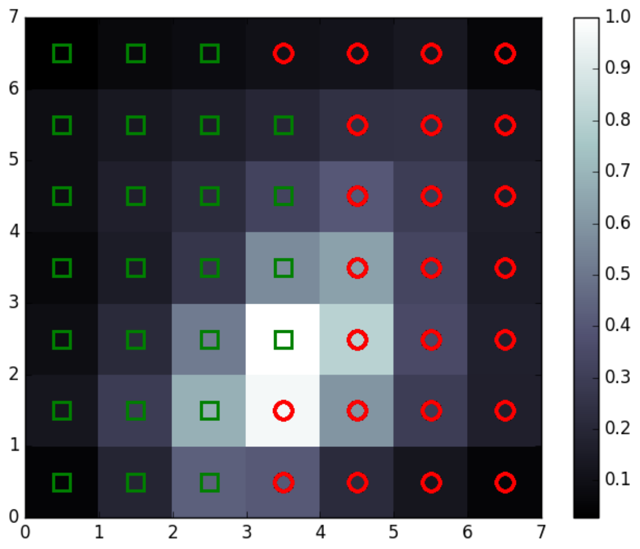


Figure 9: Spiral dataset in 7 × 7 SOM :Cutgauss Neighborhood function is used

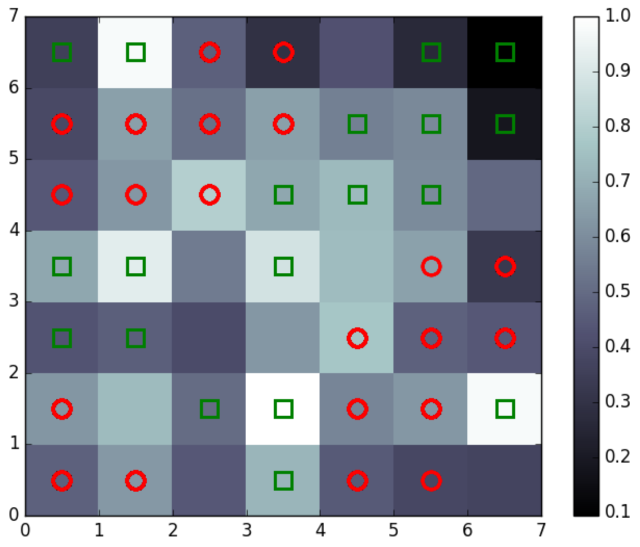


Figure 10: Spiral dataset in 7 × 7 SOM : Gaussian Neighborhood function is used

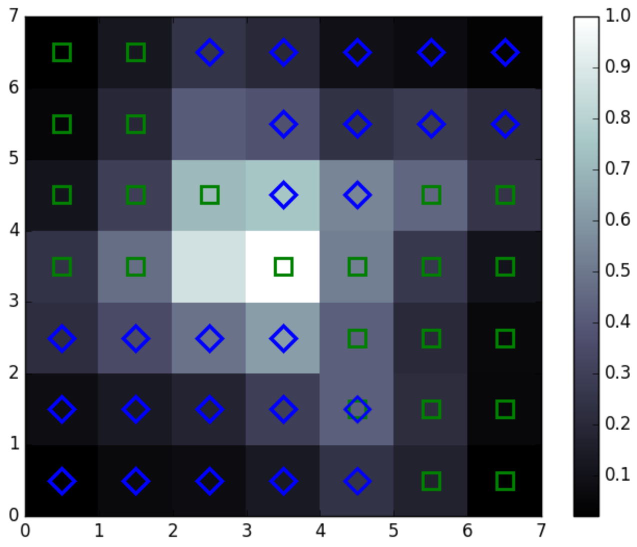


Figure 11: Gaussian Mixture dataset in 7 × 7 SOM :Bubble Neighborhood function is used

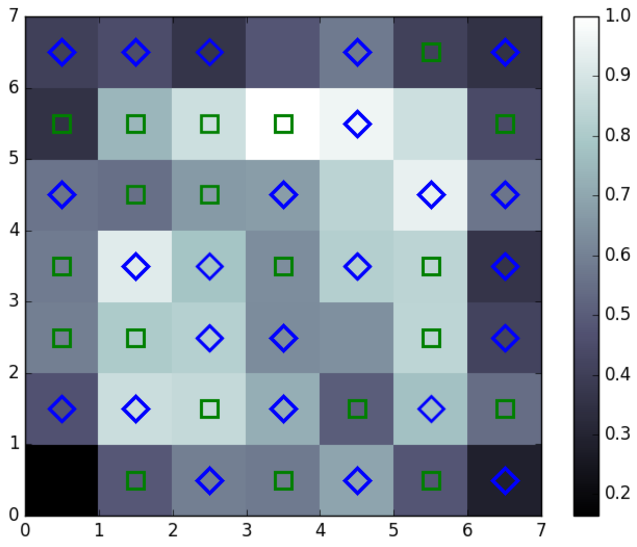


Figure 12: Gaussian Mixture dataset in 7 × 7 SOM : Gaussian Neighborhood function is used

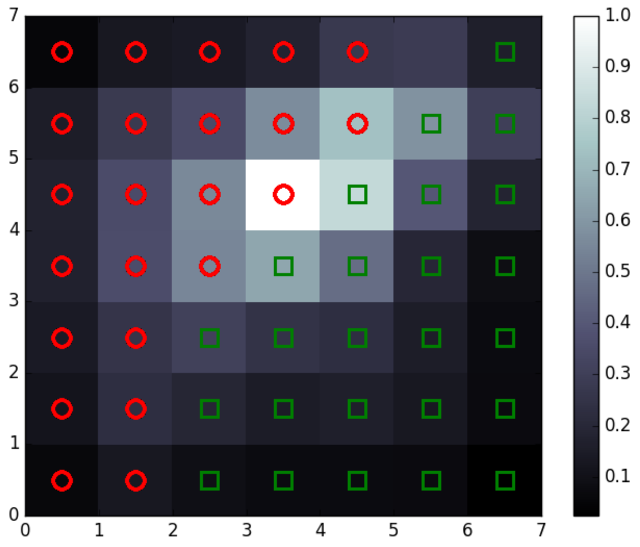


Figure 13: Elliptical dataset in 7 × 7 SOM :Gaussian Neighborhood function is used

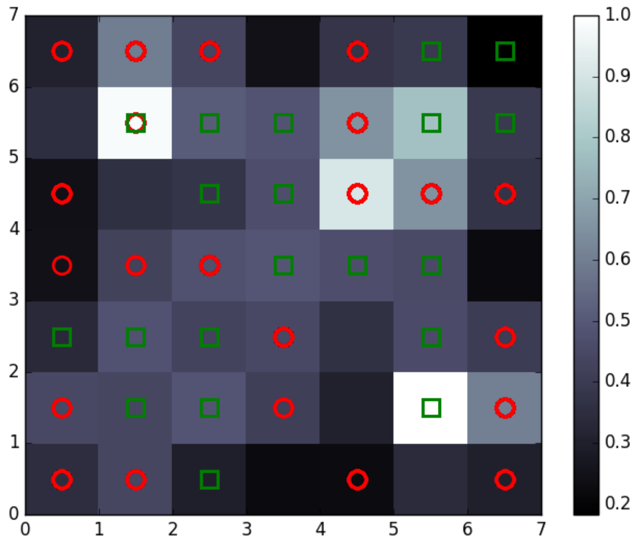


Figure 14: Elliptical dataset in 7 × 7 SOM : Heuristic Neighborhood function is used

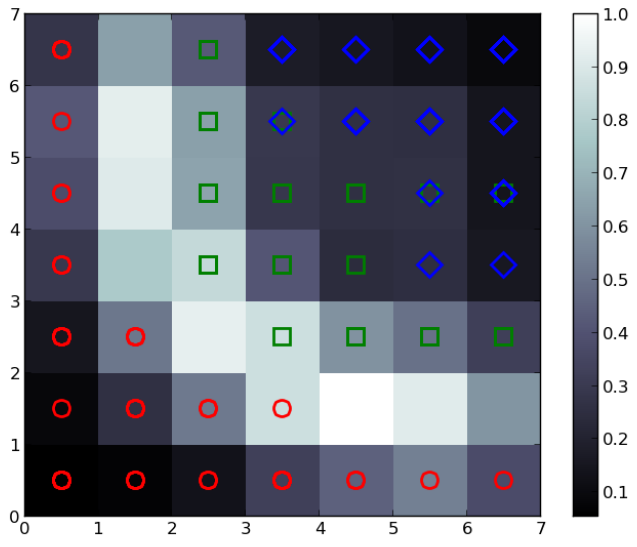


Figure 15: Triangular 3 class dataset in 7×7 SOM : Gaussian Neighborhood function is used

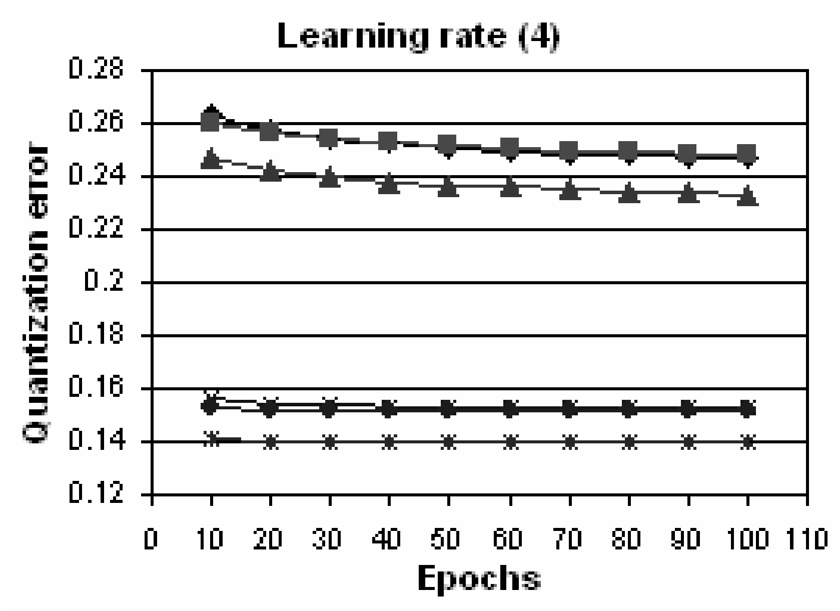


Figure 16: The averages of Quantization errors, obtained using the Glass data set when linear learning rate function is used.

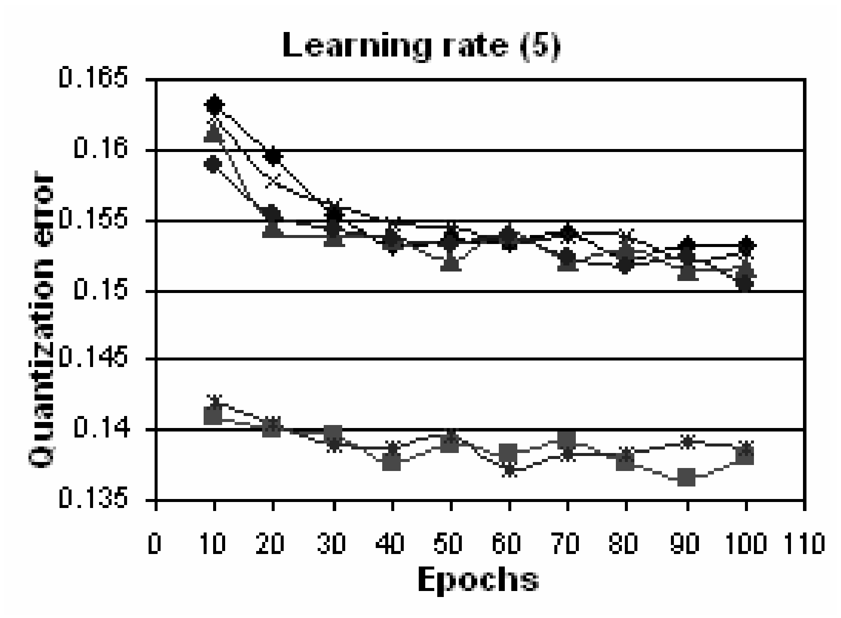


Figure 17: The averages of quantization errors, obtained using the Glass data set when Inverse of Time learning rate function is used.

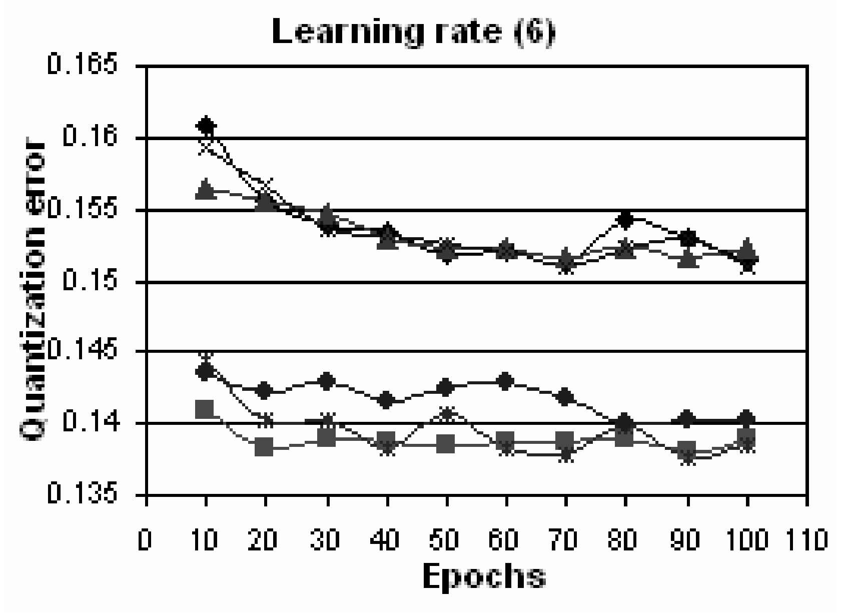


Figure 18: The averages of Quantization errors, obtained using the Glass data set when Power Series learning rate function is used.

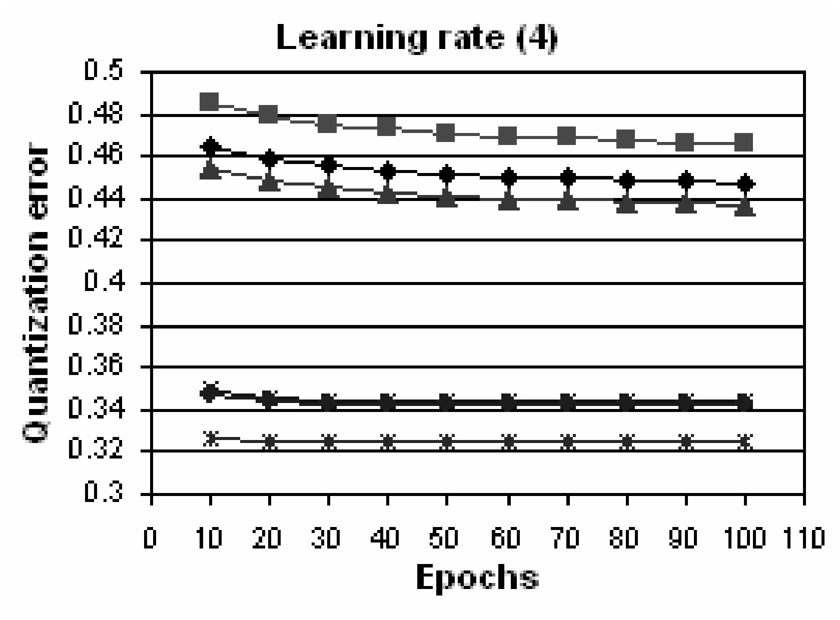


Figure 19: The averages of quantization errors, obtained using the Wine data set when Linear Time learning rate function is used.

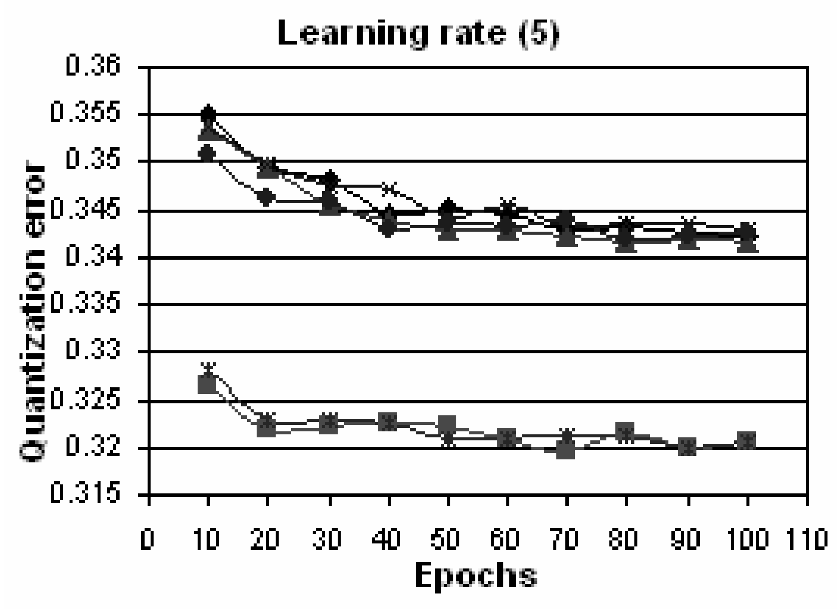


Figure 20: The averages of quantization errors, obtained using the Wine data set when Inverse Time learning rate function is used.

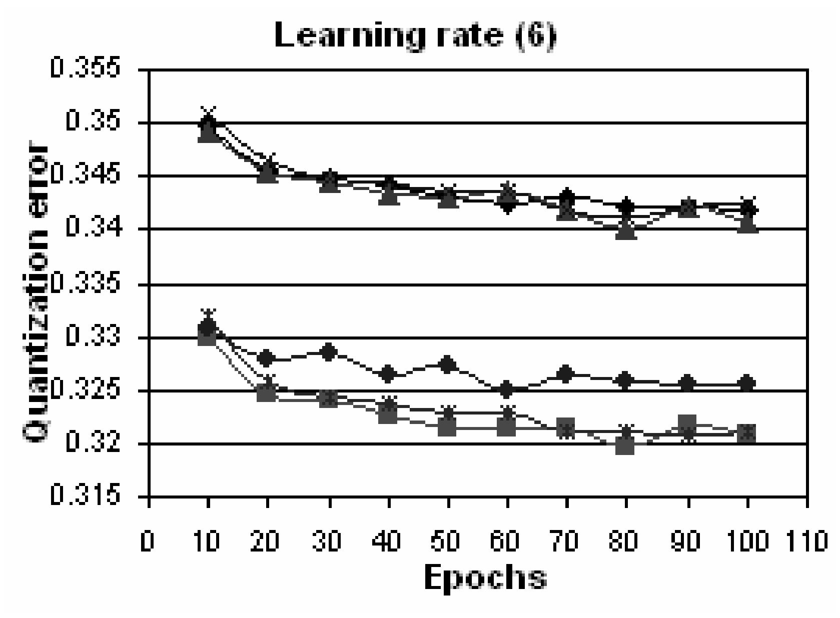


Figure 21: The averages of quantization errors, obtained using the Wine data set when power series learning rate function is used.

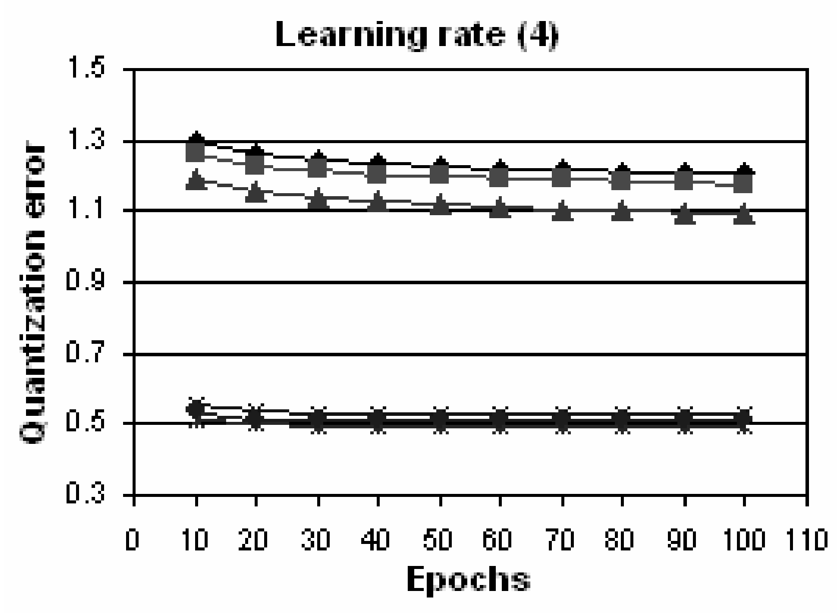


Figure 22: The averages of quantization errors, obtained using the Zoo data set when linear learning rate function is used.

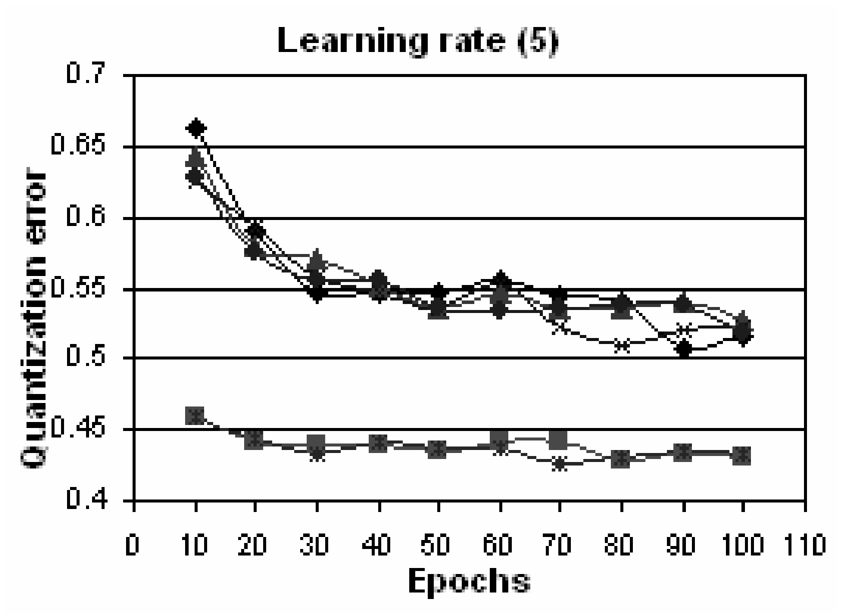


Figure 23: The averages of quantization errors, obtained using the zoo data set when Inverse Time learning rate function is used.

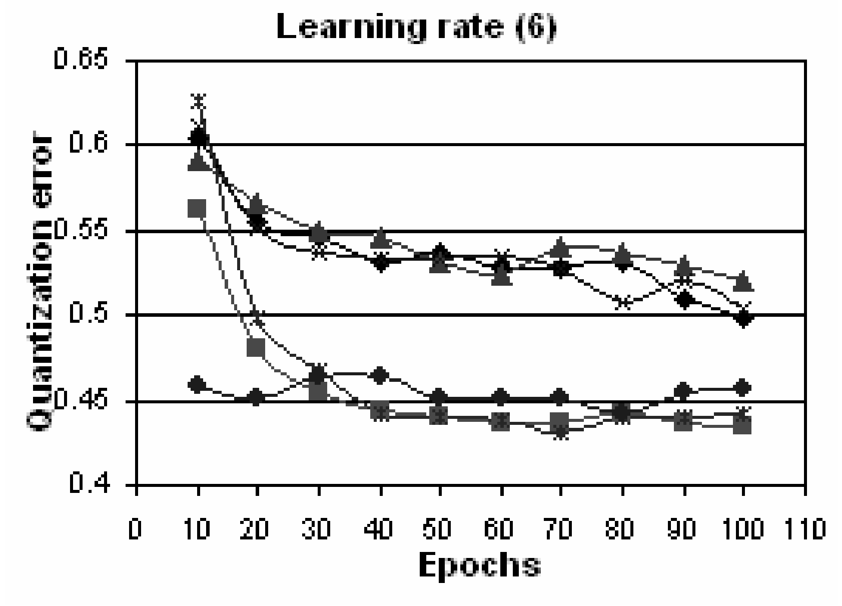
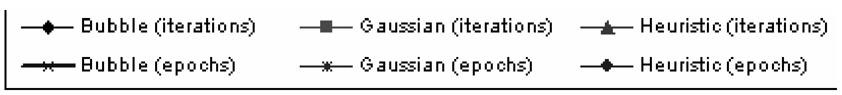


Figure 24: The averages of quantization errors, obtained using the Zoo data set when power series learning rate function is used.



6 Conclusion

The experimental results have showed that the smallest quantization error is obtained, if neighboring Gaussian function , Cutgauss funcion, Bubble function and nonlinear learning rates are used. The changes of learning rates function according to iterations do not influence the results obtained. The Heuristic and Epanechikov neighboring function yields the worst result for all the data sets analyzed. In order to get good results, the bubble function, gaussian and cutgauss can be helpful, too, but only if the learning rates are changed. The smallest quantization errors are achieved with inverse-of-time, power series and Linear learning rates. Learning rate (linear) gives large quantization errors, if learning rates are changed according to iterations. The research has shown that the neighboring function, the learning rate influence the SOM results in the sense of quantization error. So it is important to choose the proper training parameters for different data sets. It is not necessary that Gaussian function will always work there might be some another neighboring functions which can work well then gaussian function like in this experiment . In case of Spiral data set Cutgauss functions is giving good results in comparison to another neighboring functions. In case of gaussian mixture data set Bubble function is giving good result.
Neighborhood function and the number of neurons determine the granularity of the resulting mapping. The larger the area where neighborhood function has high values, the more rigid is the map. The larger the map, the more flexible it can become. This interplay determines the accuracy and generalization capability of the SOM. Generally, the neighborhood function is designed to have a global maximum at the "winning" neuron and it decreases as it gets further away from it. This makes it so that the neuron which are close to the "winning neuron" gets scaled towards the sample input the most while neurons far away get scaled the least , which creates grouping of similar neurons in the final map.

6.1 Advantages

Probably the best thing about SOMs that they are very easy to understand. It's very simple .Another great thing is that they work very well. As I have shown you they cluster data well and then are easily evaluate for their own quality so you can actually calculated how good a map is and how strong the similarities between objects are.

6.2 Disadvantages

One major problem with SOMs is getting the right data. Unfortunately you need a value for each dimension of each member of samples in order to generate a map. Sometimes this simply is not possible and often it is very difficult to acquire all of this data so this is a limiting feature to the use of SOMs often referred to as missing data.

Another problem is that every SOM is different and finds different similarites among the sample vectors. SOMs organize sample data so that in the final product, the samples are usually surrounded by simliar samples, however similar samples are not always near each other. So a lot of maps need to be constructed in order to get one final good map.

The final major problem with SOMs is that they are very computationally expensive which is a major drawback since as the dimensions of the data increases, dimension reduction visualization techniques become more important, but unfortunately then time to compute them also increases

7 References

1. Kurasova, O., Molytė, A.: Integration of the Self-organizing Map and Neural Gas with Multidimensional Scaling. Information Technology and Control 40(1), 12–20 (2011).

2. Kurasova, O., Molytė, A.: Quality of Quantization and Visualization of Vectors Obtained by Neural Gas and Self-organizing Map. Informatica 22(1), 115–134 (2011)

3. Dzemyda, G., Kurasova, O.: Heuristic Approach for Minimizing the Projection Error in the Integrated Mapping. European Journal of Operational Research 171(3), 859–878 (2006)

4. Bernatavičienė, J., Dzemyda, G., Kurasova, O., Marcinkevičius, V.: Optimal Decisions in Combining the SOM with Nonlinear Projection Methods. European Journal of Operational Research 173(3), 729–745 (2006).

5. Tan, H.S., George, S.E.: Investigating Learning Parameters in a Standard 2-D SOM Model to Select Good Maps and Avoid Poor Ones. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 425–437. Springer, Heidelberg (2004).

6. Kohonen, T.: Self-organizing Maps, 3rd edn. Springer Series in Information Sciences. Springer, Berlin (2001).

7. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository, University of California, School of Information and Computer, Irvine, CA (2007), http://www.ics.uci.edu/ mlearn/ML-Repository.html.

8 Appendix

8.1 Datasets

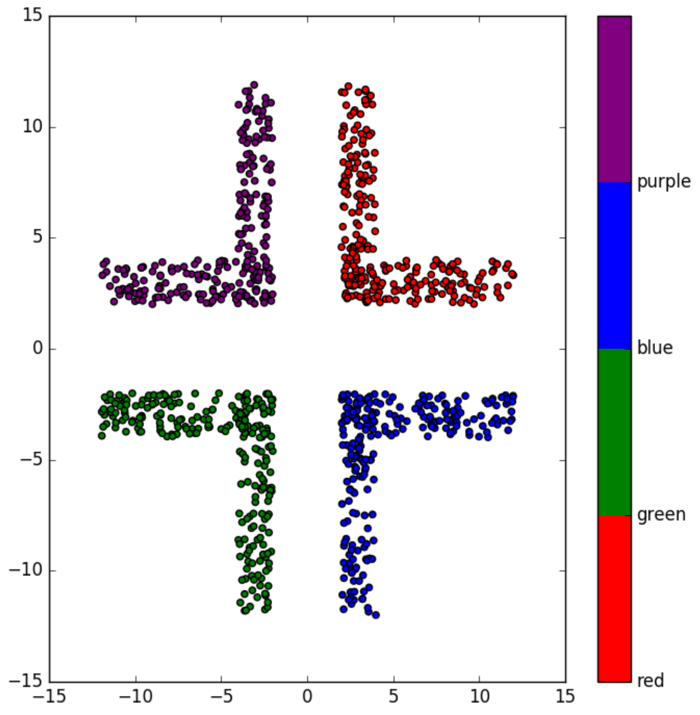


Figure 25: Corner Dataset

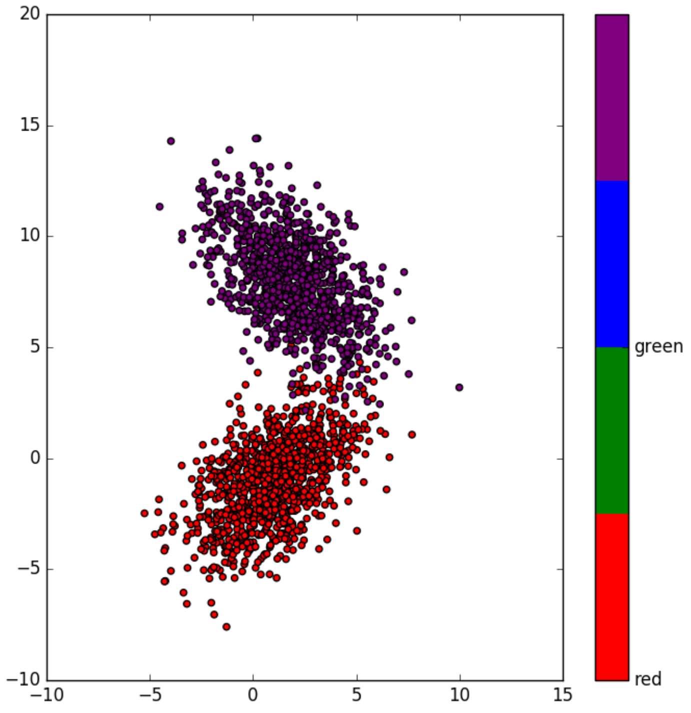


Figure 26: Eliptical Dataset

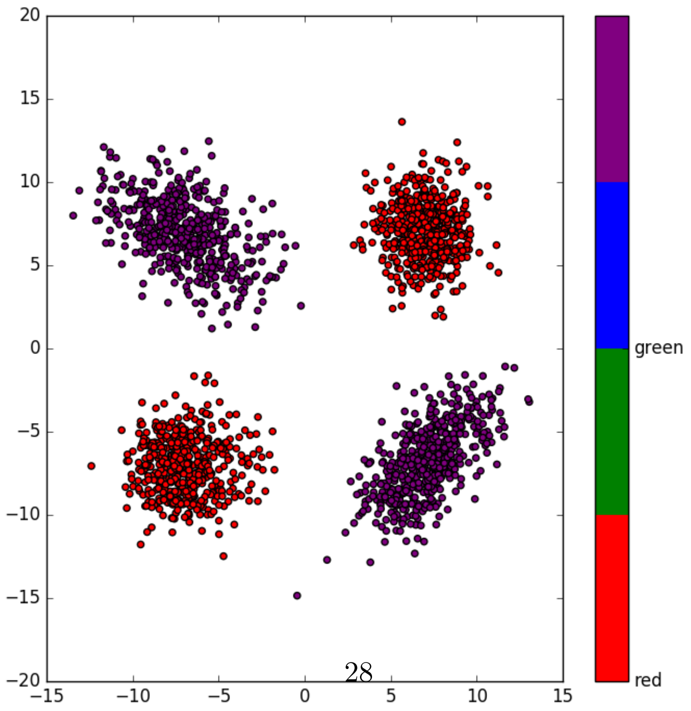


Figure 27: Gaussian Mixture Dataset

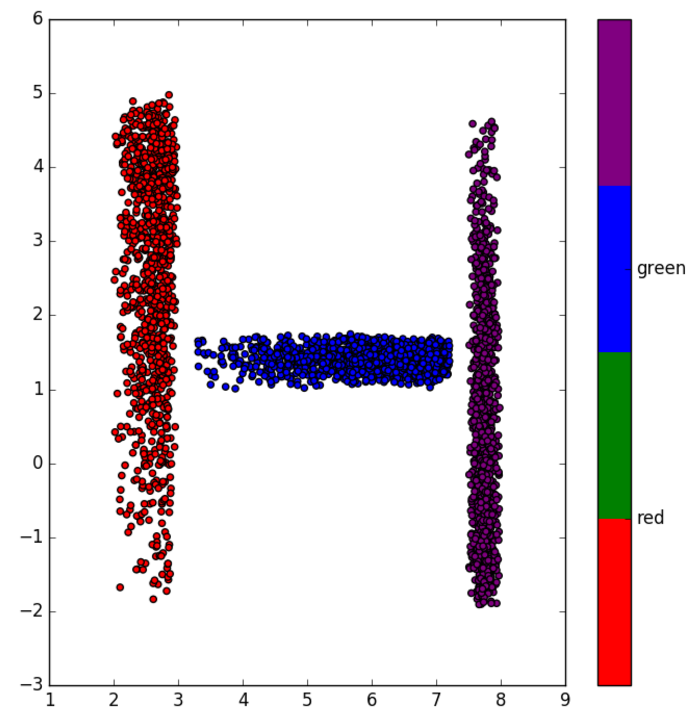


Figure 28: Triangular 3 class Dataset

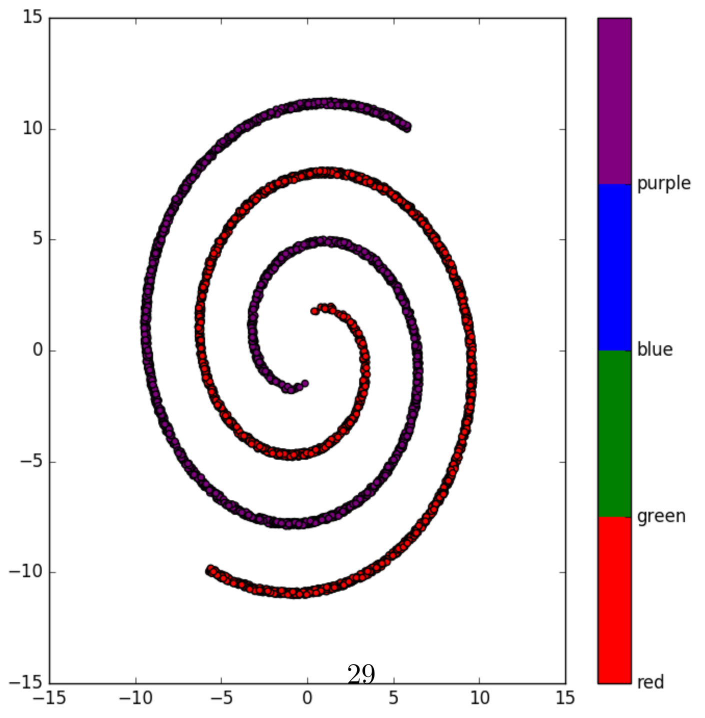


Figure 29: Spiral Dataset

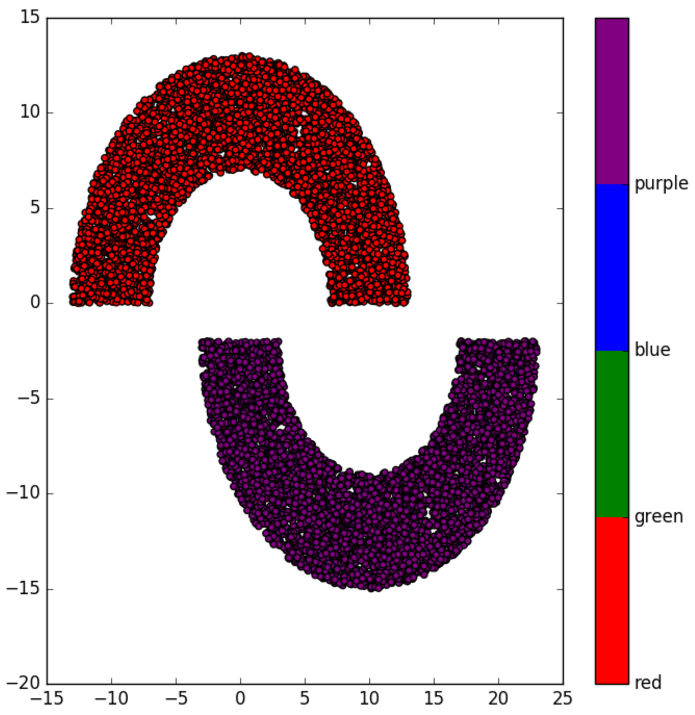


Figure 30: Double Moon Dataset