SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
VOLUME 4, ISSUE 6
ISSN 2054-7390

On Modified Self Organizing Feature Maps

[1]Jasser Jasser, [2]Tony Bazzi and [3]Mohamed Zohdy
[1]Department of Computer Science and Informatics, Oakland University, Rochester, MI 48309, USA;
[2,3]Department of Electrical and Systems Engineering, Oakland University, Rochester, MI 48309, USA;
jjasser@oakland.edu; tbazzi@oakland.edu; zohdyma@oakland.edu

ABSTRACT

This paper presents optimization of self-organizing feature maps by adjusting tunable parameters and in the iterative process by utilizing linear algebra concepts. A gradient rule is applied on the weights matrix singular value decomposition values during convergence phases to optimize the algorithm while achieving sufficient statistical accuracy. Tunable parameters such as the learning rate, and neighborhood radius are adjusted to support the learning algorithm. The algorithm presented herein is tested on the self-organization of the standard Iris dataset, in the literature.

Keywords: Self-Organizing Feature Map, Neural Networks, Unsupervised Learning, Singular Value Decomposition

1 Introduction

1.1 Self-Organizing Feature Map

The self-organizing feature map (SOFM) is a neural network based algorithm developed by Kohonen that iteratively projects high dimensional input vectors into 1D, 2D, or 3D topological map, to visualize dissimilarities between data by grouping them into different clusters [1]. Thus, making it easier to obtain an insight into the topographic relationship between the data items. SOFM have been widely utilized in the industry and academia to establish correlations between data items on a visual output lattice, and many research groups previously modified it in several directions, and found useful applications. Industry applications include monitoring and diagnosing engine health [2], malware removal [3], Global Positioning Systems [4], and many others in the fields of robotics, telecommunications, acoustic and musical, process control, machine vision, classification of mathematical curves, image analysis and signal processing [1, 5]. There are two main stages for the SOFM, the competitive, and the adaptive stages. In the competitive stage, the nodes compete to become the best matching unit (BMU) of a randomly selected input sequence. The node that is most like the input vector, having the least Euclidean distance is chosen as the BMU, and hence considered a winning node[1].

$$\beta = argmin\left(\sqrt{\sum_{i=0}^{i=n}\left(V_i(t) - W_i(t)\right)^2}\right) \quad (1)$$

β: *Best Matching Unit*

DOI: 10.14738/tmlai.46.2533
Publication Date: 30th December, 2016
URL: http://dx.doi.org/10.14738/tmlai.46.2533

$V_i(t)$: $ith\ Input\ Vector$

$W_i(t)$: $ith\ Weight\ Vector$

n: Number of features

The neighborhood around BMU has a radius that is calculated through the neighborhood function. Because of the unique decaying feature, the neighborhood area exponentially shrinks over a time constant to the size of few nodes [6].

$$\sigma = \sigma_0 e^{-\frac{t}{\lambda^2}} \tag{2}$$

σ: $Neighborhood\ Function$

σ_0: $Map\ Radius$

λ: $Time\ Constant$

t: $Iteration\ Number$

The nodes within the neighborhood range are updated to become closer in space to the input vector. These nodes are identified by calculating the Euclidean distance between them and the BMU as in (4). If the node's distance is within the neighborhood radius, then the update occurs on it as follows.

$$W_i(t+1) = W_i(t) + \theta * \alpha * (V_i - W_i) \tag{3}$$

$W_i(t)$: $ith\ Weight\ Vector$

θ: $Best\ Matching\ Unit\ Influence$

α: $Learning\ Rate$

V_i: $Input\ Vector$

$$\theta = e^{\left(\frac{\left(\sqrt{\sum_{i=0}^{i=n}(N_i-\beta_i)^2}\right)^2}{2\sigma^2}\right)} \tag{4}$$

θ: $Best\ Matching\ Unit\ Influence$

N: $Node\ within\ Neighborhood\ Radius$

β: $Best\ Matching\ Unit$

σ: $Neighborhood\ Function$

n: $Number\ of\ Features$

By the end of each iteration, a new learning rate is calculated in a decaying matter follows.

$$\alpha(t) = \alpha_0 e^{\left(-\frac{t}{T}\right)} \tag{5}$$

α: $Learning\ Rate$

α_0: $Initial\ Learning\ Rate$

t: $Current\ Iteration\ Number$

T: $Total\ Length\ of\ Training$

However, the theoretical research frontier has yet to determine a better learning mechanisms for neural network based algorithms to mitigate large convergence time of the learning phase, and parallelization in-capabilities due to the iterative nature of the algorithm and intensive computing resources needed [7-9]. In this paper, a new method providing users the flexibility to optimize the convergence number of iterations is discussed for clustering of input data. It also achieves a level were the SOFM is capable of semi-supervising itself. However, the end-user still must adjust the initial neighborhood radius, and the learning rate parameters, to sacrifice or maintain the accuracy of the SOFM in favor of further reducing the numb

1.2 Application of Singular Value Decomposition on Weight Matrix

Consider a non-normal matrix ***A***, it is not unitarily or orthogonally like a diagonal matrix. Nevertheless, this matrix can be simplified using orthogonal transformation, to some extent, by a famous decomposition called the Singular Value Decomposition (SVD) [10]. SVD is considered the most widely used method for real or complex matrix factorization with a lot of applications, among them, enhancement of digital images [11], image, and audio watermarking [12, 13], data mining[14], and many other. In this paper, we use SVD to determine Whether the weight matrix (w) is enduring heavy changes during the loop cycle of the algorithm. Reaching a point where there are no more updates occurring on the matrix triggers an event to break the loop of the algorithm.

To calculate SVD values, let's consider a real $m \times n$ matrix ***A***, with $m \geq n$, then:

$$A = U\Sigma V^T \tag{6}$$

Where *U* is a matrix that consists of *n* orthonormalized eigenvectors that are allied with the largest eigenvalues of AA^T, while the normalized eigenvectors of A^TA are found in the *V* matrix, and the SVD values are found in the Σ matrix [15]. In this paper, we consider the normalized summation of the Σ matrix to monitor the rate of change of the weight matrix. Notice that another metrics can also be used but in our case, the summation provided the best accuracy results.

2 Linear Algebra-based SOFM Algorithm

In this section, we discuss the concept of Linear Algebra-based Self Organizing Feature Map (LA-SOFM), and the different steps that constitute the proposed algorithm. Statistical methods are integrated with the SVD vector calculation where its recursive effects on the convergence speed of the algorithm along with its accuracy are analyzed, and presented in the following sections:

2.1 Initialization and Matrix Size

Initialization of SOFM involves randomizing the weights matrix but this does not mean, however, that random initialization would be the best. In our approach, the inputs are further normalized to achieve faster execution. A normally distributed randomization in the range between (0, 1) is considered for the weights matrix.

2.2 Learning Rate and Neighborhood Functions

The learning rate is calculated using the following stochastic method. Notice that the summation of the SVD values (ε) replaces the total requested number of iterations in the conventional SOFM algorithm. Since we leave our algorithm to decide the optimum number of iteration, it is necessary to update the learning rate (α) in gradient descent fashion like the original algorithm, and for that, the SVD values are

chosen to replace the total number of iteration since they converge to zero in the fashion we require, ensuring our learning rate is monotonically decreases with the regression step. Notice that we take the SVD values calculated in the previous iteration, and for that, we initialize ε to 1 to calculate the first iterations (t=1).

$$\alpha(t) = \alpha_0 e^{\left(-\frac{t}{\varepsilon(t-1)}\right)} \tag{7}$$

α: $Learning\ Rate$

α_0: $Initial\ Learning\ Rate$

t: $Current\ Iteration\ Number$

ε: $Summation\ of\ SVD\ Values$

The neighborhood function adaptation around the BMU is applied per plasticity control kernel taking on Gaussian or similar form.

$$\lambda(t) = \frac{\varepsilon(t-1)}{\log(\sigma_0)} \tag{8}$$

λ: $Time\ Constant$

ε: $Summation\ of\ SVD\ Values$

σ_0: $Map\ Radius$

t: $Current\ Iteration\ Number$

$$\sigma(t) = \sigma_0 e^{\left(-\frac{t}{\lambda(t)}\right)} \tag{9}$$

σ: $Neighborhood\ Function$

ε: $Summation\ of\ SVD\ Values$

σ_0: $Map\ Radius$

t: $Current\ Iteration\ Number$

2.3 Weight Matrix Update and SVD value Calculations

Updating the weights is like the conventional SOFM. However, the SVD values summation are calculated every iteration after the weights are updated, and then the rate of change of the SVD values summation. obtaining a rate of change of zero means we arrived at a convergence for the lattice. To make sure of convergence, we verify its consistency for 10 consecutive cycles before breaking the loop, or the iterative process will continue its work normally.

3 Experiments and Analysis

3.1 Iris Flower Dataset

The iris flower dataset is a multi-class classification dataset that represents three types of iris flowers, Iris Setosa, Iris Versicolor, and Iris Virginica, each with different sepal length, and width, and petal length, and width dimensions [16]. The dataset contains 150 inputs, 50 for each iris type.

3.2 Training and Testing

The iris dataset is randomly split into training, and testing subsets every time a session is conducted. Furthermore, data is normalized to speed up the clustering process, and the LA-SOFM are tuned to achieve the best results possible. Clustering has been done using both the conventional method, and ours as shown in Table1, where 80% of the original dataset is training data, and 20% is testing. While in Table2, the training, and testing datasets are split in half (50% - 50%).

Table 1. Testing Results with 80% Training and 20% Testing.

Test #	LA-SOFM			Conventional SOM		
	# iterations	Time	Accuracy	# iterations	Time	Accuracy
1	121	3.5706	0.9666	1000	19.3728	0.9000
2	138	3.9672	0.9333	1000	19.6997	0.9666
3	148	4.1391	1.0000	1000	22.7906	1.0000
4	133	4.0139	0.9666	1000	21.9220	1.0000
5	122	3.5513	0.9666	1000	20.5624	1.0000
6	140	4.0854	0.9666	1000	19.9320	0.9333
7	146	4.0480	0.9666	1000	23.0422	0.9333
8	135	3.9431	1.0000	1000	21.0772	0.9333
9	126	3.6985	0.9666	1000	20.1508	0.9333
10	149	4.2281	1.0000	1000	20.5058	0.9333
Average	135.8	3.9245	0.9733	1000	20.9055	0.9533

In this test, 80% of the randomly selected data were training data and 20% were testing. LA-SOFM proved better by providing better accuracy with way less execution time.

Table 2. Testing Results with 50% Training and 50% Testing.

Test #	LA-SOFM			Conventional SOM		
	# iterations	Time	Accuracy	# iterations	Time	Accuracy
1	121	3.8937	0.8933	1000	20.4131	0.8533
2	137	4.2285	0.9066	1000	21.2074	0.8800
3	120	3.7879	0.9600	1000	20.6351	0.9066
4	97	3.2838	0.9600	1000	20.3298	0.9600
5	134	4.1403	0.9333	1000	20.5927	0.8933
6	128	3.9855	0.9200	1000	20.4367	0.9200
7	108	3.7454	0.9333	1000	20.3451	0.9600
8	129	4.0464	0.9733	1000	19.8269	0.9200
9	142	4.3352	0.9733	1000	20.3973	0.9600
10	130	4.1004	0.9733	1000	20.4778	0.9733
Average	124.6	3.9547	0.9426	1000	20.4661	0.9226

In this test, LA-SOFM maintained better accuracy in less execution time even in 50%-50% data split.

From Table1, and Table2, LA-SOFM is shown to produce a better accuracy than the conventional SOFM, while having a way shorter execution time since it stops the iteration of the algorithm once it reaches a statistical efficiency. In our method, the algorithm keeps iterating until there are no more updates happening on the weight matrix. Usually, we achieve that when the rate of change in the SVD values summation is zero, but in these tests, we break the loop whenever we achieve a rate of change that is less than $\mathbf{1}e-\mathbf{4}$ for *10* consecutive iterations since it produced accurate results. In Figure1, it shows how the SVD values played a role in converging the learning rate, and neighborhood function. It also shows the rate of change of the value summation, where is displays a high rate of change at the first epochs, but then the rate of change started to go down until it reached the threshold we set.

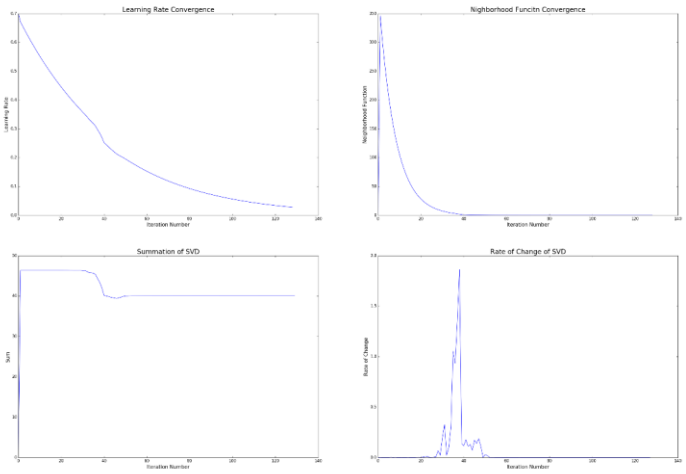

Figure 1. LA-SOFM Functions.

This figure shows what goes behind the scene of the LA-SOFM algorithm, and how the SVD integrate the SOFM functions consistently together.

4 Conclusion

Since the introduction of the modern computer circa 1950s, and the SVD values have been a constant factor in the research frontier since much of scientific computing depends critically in one way or another on numerical linear algebra algorithm. This paper reviewed the use of SVD values to determine, and control the flow of the iterative process of the LA-SOFM.

5 Future Work

Having the whole iterative process of the LA-SOFM to be decided based on the eigenvalue is the first step in this research. Still, the relationship between the eigenvalue, learning rate, the neighborhood function, and the generalization of the weight matrix is a good area to go deeper in. Reaching a level where the relationship between the mentioned parameters are acknowledged, then a full automatic LA-SOFM where the user does not need to worry about the parameters can be achieved.

REFERENCES

[1] T. Kohonen and T. Kohonen, '"Essentials of the self-organizing map," *Neural Networks*, vol. 37, 0, pp. 65; 65.

[2] Bryant, '"Self-organizing maps applied to engine health diagnostics," *Self (New York)*, vol. 3, no. 02.

[3] R. Yang, V. Kang, S. Albouq and M. Zohdy, '"Application of Hybrid Machine Learning to Detect and Remove Malware," *Transactions on Machine Learning and Artificial Intelligence*, vol. 3, no. 4, pp. 16.

[4] A. Nsour and M. Zohdy, '"Self Organized Learning Applied to Global Positioning System (GPS) Data," *Proceedings of the 6th WSEAS International Conference on Signal, Speech, and Image Processing, Lisbon, Portugal.*

[5] L. Puengue and Zohdy Mohamed, '"Modified Self Organizing Feature Maps for Classification of Mathematical Curves." *International Journal of Computer and Information Technology*, vol. 2, no. 5.

[6] Takaaki Aoki and Toshio Aoyagi, '"Self-Organizing Maps with Asymmetric Neighborhood Function," *Neural Computation*, vol. 19, no. 9, Sep 1, pp. 2515-2535.

[7] T. Kohonen, '"The self-organizing map," *Proc IEEE*, vol. 78, no. 9, 0, pp. 1480; 1480.

[8] J.L. Giraudel, '"A comparison of self-organizing map algorithm and some conventional statistical methods for ecological community ordination," *Ecol.Model.*, vol. 146, no. 1-3, pp. 339; 339.

[9] V.A. Patole, V.K. Pachghare and P. Kulkarni, '"Self Organizing Maps to build intrusion detection systems," *Journal of Computer Applications*, vol. 1, no. 7.

[10] S. Banerjee and A. Roy, '"Linear algebra and matrix analysis for statistics,", 2014.

[11] H. Demirel, C. Ozcinar and G. Anbarjafari, '"Satellite image contrast enhancement using discrete wavelet transform and singular value decomposition," *IEEE Geoscience and remote sensing letters*, vol. 7, no. 2, pp. 333-337.

[12] C. Lai and C. Tsai, '"Digital image watermarking using discrete wavelet transform and singular value decomposition," *IEEE Transactions on instrumentation and measurement*, vol. 59, no. 11, pp. 3060-3063.

[13] V. Bhat, I. Sengupta and A. Das, '"An adaptive audio watermarking based on the singular value decomposition in the wavelet domain," *Digital Signal Processing*, vol. 20, no. 6, pp. 1547-1558.

[14] R.L. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar and R. Namburu, '"Data mining for scientific and engineering applications,", vol. 2, 2013.

[15] G.H. Golub and C. Reinsch, '"Singular value decomposition and least squares solutions," *Numerische mathematik*, vol. 14, no. 5, pp. 403-420.

[16] J.C. Bezdek, J.M. Keller, R. Krishnapuram, L.I. Kuncheva and N.R. Pal, '"Will the real iris data please stand up?" *IEEE Trans.Fuzzy Syst.*, vol. 7, no. 3, pp. 368-369.

