Chapter

Improving Feature Map Quality of SOM Based on Adjusting the Neighborhood Function

Le Anh Tu

Abstract

This chapter presents a study on improving the quality of the self-organizing map (SOM). We have synthesized the relevant research on assessing and improving the quality of SOM in recent years, and then proposed a solution to improve the quality of the feature map by adjusting parameters of the Gaussian neighborhood function. We have used quantization error and topographical error to evaluate the quality of the obtained feature map. The experiment was conducted on 12 published datasets and compared the obtained results with some other improving neighborhood function methods. The proposed method received the feature map with better quality than other solutions.

Keywords: quantization error, topographical error, self-organizing map, feature map, projection quality, learning quality

1. Introduction

SOM is a very useful neural network for visualization and data analysis. Among SOM's application areas, urban design is a potential area. Many of SOM's applications can be included in urban design such as: analysis of growth factors in urban design proposal [1], consider urban spatial structure [2], analysis of city systems [3], city data mining [4], predicting accessibility demand for healthcare infrastructure) [5], etc. However, for SOM's calculation results to be more accurate, improving the quality of feature map is a problem to solve.

SOM creates a map of the input data in the multi-dimensional space to the less dimensional space that is usually two-dimensional space called by the feature map of the data. To evaluate the quality of feature map, people mainly use two indicators: learning quality and projection quality [6–9]. The learning quality indicator is determined through measurement of quantization error (QE) [10, 11]. The projection quality indicator is determined through measurement of topographical error (TE) [12–14]. If the values of the QE and TE are small, feature map will be assessed with good quality.

Many studies have shown that the quality of feature map is affected greatly by the initial parameters of the network, including map size, numbers of training and neighborhood radius [11, 15–18]. Beside that, a feature map achieving with a set of fit parameters is not considered as the best quality map. Therefore, improving the feature map quality of SOM is concerned by many researchers.

To achieve good quality map for each dataset in traditional method is “trying error” with different parameters of the network. These parameters, creating a map with the smallest error measurement are suitable for the dataset [11]. According to Chattopadhyay et al. [19], with a specific dataset, the size of the map is selected by “trying error” until reaching value of QE, TE small enough. Polzlbauer [20] indicates the technical correlation between QE and TE, which TE often arises when QE reduces. In case of increasing the size of Kohonen layer, QE may reduce but TE increases (i.e., the large size of the Kohonen layer can distort the shape of the map), and vice versa when the size of Kohonen layer is too small, TE is not trust. The use of a small neighborhood radius leads to reduced QE. If the neighborhood radius is the smallest value, QE will reach a minimum value [21].

Besides the method of “trying error” to determine a suitable network configuration, the study on improving the algorithm of SOM to enhance the quality of feature map is also interested by researchers. Germen [22, 23] optimized QE by integrating “hit” parameter when updating the weight vector of the neurons, the term “hit” means the number of excitation to a neuron (or BMU counter). The “hit” parameter will determine adjusting weight vector of neuron, i.e., the neurons representing for many samples are adjusted less (to ensure not lose information) than neurons representing for less samples.

Neme [24, 25] proposed SOMSR model (SOM with selective refractoriness), which allows reducing TE. In this model, the neighborhood radius of the BMU did not reduce gradually in the learning process. In every training times, every neuron in the neighborhood radius of the BMU will decide itself whether being affected by the BMU or not in the next training.

Kamimura [26] has integrated the “firing” rate in the distance function to maximize information input. The “firing” rate identifies the important degree of each feature comparing to the remaining features. This method can reduce both QE and TE; however, with each dataset, it needs to “trying error” in several times to determine the appropriate value of “firing.”

Lopez-Rubio [27] describes the topographical error of the map as a state of “self-intersections.” If it detects a “self-intersections” state between neurons after each learning times, it will redo that learning times. This solution can reduce the TE, but increase QE.

Another approach is to adjust the scope and the learning rate of the neighborhood neurons. Kohonen [11] homogenised learning rate of all the neurons in the neighborhood radius to learning rate of the BMU by using the “bubbles” neighborhood function. He concluded that the bubbles function is less effective than the Gaussian function.

Aoki and Aoyagi [28] and Ota et al. [29] published an asymmetric neighborhood function. The essence of this function is extending the neighborhood radius towards one direction and shrinking the opposite one. Theoretically, this could “slide” down the topographical error out of the map. However, his experiment has been limited in the certain situations and not really convinced.

Lee and Verleysen [30] replaced the neighborhood function by “fisherman” rule. “Fisherman” rule updates the neurons in neighborhood radius following the recursive principle, which BMU is adjusted following input sample and the BMU-adjacent neurons (adjacent level 1) is governed by the BMU (unadjusted by input samples), moreover, each adjacent neuron in level 2 is adjusted by the previous adjacent neuron in level 1. The remaining neurons in the neighborhood radius are adjusted in the same rule. However, the way to determine the order of the adjacent neurons when they are organized in a rectangular or a hexagonal grid is not shown in his article. In addition, he concluded that the Gaussian function has better results than the rule of “fisherman”.

It can be recognized that achieving a feature map with good quality according to many criterion is a difficult problem. So far, there has not any solution, reducing simultaneously both QE and TE that is well-applied for every dataset.

In this chapter, we improved Gaussian neighborhood function by adding the adjusting parameter in order to simultaneously reduce the QE, TE of the map. The next contents of the chapter include: Section 2 presents an overview of SOM and assessment measures of the quality of feature map; Section 3 presents our studying on adjusting the parameter of the Gaussian neighborhood function; Section 4 indicates the empirical results and the conclusion of the proposed method.

2. Self-organizing map neural network

2.1 Structure and the algorithm

SOM includes input and output Kohonen layer. Kohonen layer is usually organized under the form of a two-dimensional matrix of neurons. Each unit i (neuron) in the Kohonen layer having a weight vector $w_i = [w_{i,1}, w_{i,2}, \dots, w_{i,n}]$, with n is the size of the input vector; $w_{i,j}$ is the weight vector of neuron i going with input j (**Figure 1**). SOM is trained by unsupervised algorithm. The process is repeated many times, at time t doing three steps:

- Step 1. Finding BMU: randomly select sample $x(t)$ from dataset (with t is training times), search for a neuron c of the Kohonen matrix containing the minimum *dist* distance (frequently use functions of Euclidean, Manhattan or vector dot product). Neuron c is called by *Best Matching Unit* (BMU).

$$dist = \|x(t) - w_c\| = \min_i \{\|x(t) - w_i\|\} \tag{1}$$

- Step 2. Calculating neighborhood radius of BMU: using the interpolation function (reduce gradually following the times of iterations)

$$N_c(t) = N_0 \exp\left[-\frac{t}{\lambda}\right] \tag{2}$$

where $N_c(t)$ is the neighborhood radius in the t training time; N_0 is initial neighborhood radius; $\lambda = \frac{K}{\log(N_0)}$ is time constant, with K is the total number of iterations.

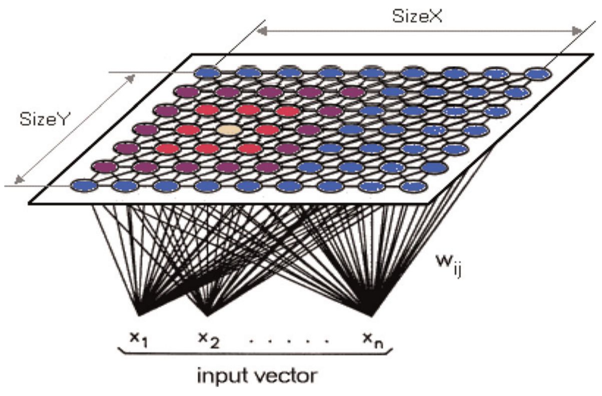

Figure 1.
Illustrates the structure of SOM.

- Step 3. Updates weight vector of neurons in the neighborhood radius of BMU towards being near to sample $x(t)$:

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

where $L(t)$ is the learning rate at the iteration t, (the learning rate is reduced simply along with time similar to neighborhood radius, with $0 < L(t) < 1$). $L(t)$ could be a linear function, exponential function ...; $h_{ci}(t)$ is a neighborhood function, showing the impact of distance on the learning process calculated by the formula (4)

$$h_{ci}(t) = \exp\left[-\frac{\|r_c - r_i\|^2}{2{N_c}^2(t)}\right] \tag{4}$$

where r_c and r_i are the positions of neuron c and neuron i in Kohonen matrix.

2.2 The quality of feature map

Quantization error and topographical error are main measurements to assess the quality of SOM. Quantization error is the average difference of the input samples compared to its corresponding winning neurons (BMU). It assesses the accuracy of the represented data, therefore, it is better when the value is smaller [11].

$$QE = \frac{1}{T}\sum_{t=1}^{T}\|x(t) - w_c(t)\| \tag{5}$$

where $x(t)$ is the input sample at the training t; $w_c(t)$ is the BMU's weight vector of sample $x(t)$; T is total of training times.

Topographical error assesses the topology preservation [13, 14]. It indicates the number of the data samples having the first best matching unit (BMU_1) and the second best matching unit (BMU_2) being not adjacent. Therefore, the smaller value is better.

$$TE = \frac{1}{T}\sum_{t=1}^{T} d(x(t)) \tag{6}$$

where $x(t)$ is the input sample at training times t; $d(x(t)) = 1$ if BMU_1 and BMU_2 of $x(t)$ not adjacent, vice versa, $d(x(t)) = 0$; T is total of training times.

3. Adding adjust parameter for Gaussian neighborhood function

Formula 3 shows the learning ability of SOM depends on two components: learning rate $L(t)$ and neighborhood function $h_{ci}(t)$.

Because the learning rate decreases simply over time, it should define the general learning rate of SOM over the training time. Therefore, the quality of feature map will be influenced mainly by neighborhood function $h_{ci}(t)$. The adjustment of the neighborhood function will affect directly to the learning process and the quality of the feature map of SOM.

Neighborhood function $h_{ci}(t)$ defines the influence level of input sample on neurons in the neighborhood radius $N_c(t)$ of BMU (**Figure 2**).

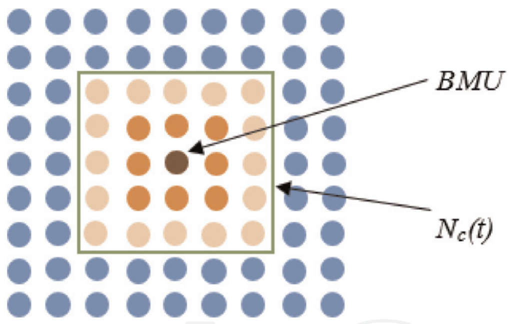

Figure 2.
Illustrates the influencing of input sample on the neurons in the neighborhood radius at training times t.

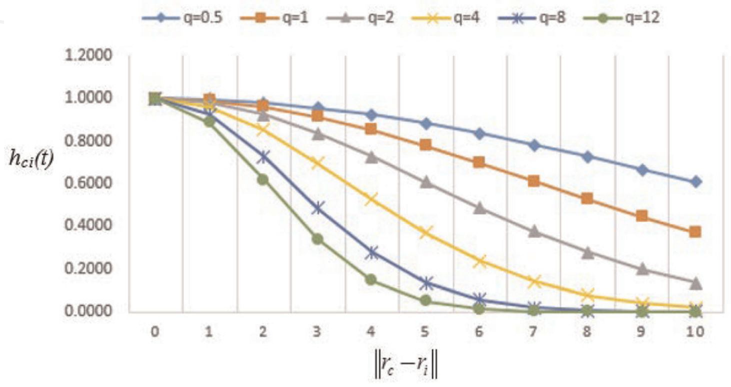

Figure 3.
Illustrates function $h_{ci}(t)$ after changing the value of q.

The formula (4) is rewritten in the following general form:

$$h_{ci}(t) = \exp\left[-q\frac{\|r_c - r_i\|^p}{N_c^p(t)}\right] \tag{7}$$

where q and p are two adjustable parameters, with $q \geq 0$ và $p \geq 0$.

It shows that the value of $h_{ci}(t)$ depending on the distance from the position of the being assessed neuron (r_i) (neuron i) to the position of BMU (r_c) and parameters q, p, specifically:

- If $\|r_c - r_i\| = 0$ (BMU is neuron being assessed), $h_{ci}(t) = 1$.
- If $\|r_c - r_i\| = N_c(t)$ (the being assessed neuron in the farthest position in neighborhood radius $N_c(t)$), the value of the neighborhood function depends on parameter q, with:

$$h_{ci}(t) = \exp[-q] \tag{8}$$

The formula (8) shows the minimum value of function $h_{ci}(t)$ depends on parameter q.

Figure 3 illustrates the neighborhood function $h_{ci}(t)$ in case of the neighborhood radius $N_c(t) = 10$, where p = 2 and q = 0.5, 1, 2, 4, 8, 12.

3.1 Parameter *q*

In principle, the bigger adjusting level of neurons's weight vector in the current learning times, the higher their difference with other input patterns in other learning times is. This is the cause of increasing the quantization error. Therefore, to reduce the QE, we must reduce the level and scope of the influencing of input sample, i.e., the increase of the value of q will reduce QE.

However, if q is too large, the learning ability of the map is restricted, i.e., the topography of map changes less, and partly depends on the initialization of the neural's weight vector. On the other hand, neighborhood radius $N_c(t)$ can be shrunk, due to $h_{ci}(t) \approx 0$ with neurons in remote positions of neighborhood radius (i.e., neurons in remote positions in the neighborhood radius are not adjusted or adjusted negligibly by input sample). Therefore, to ensure that all the neurons in the neighborhood radius $N_c(t)$ are adjusted by the input sample, the parameter q is not allowed to be too large. For example, the case of $q = 8$ and 12, function $h_{ci}(t) \approx 0$ when the value of $\|r_c - r_i\|$ reaching to $N_c(t)$.

In case of $q \approx 0$, Gaussian function has the same result as bubble function, i.e., $h_{ci}(t) \approx 1$ with all neurons in the neighborhood radius $N_c(t)$. As a result, if the neighborhood radius $N_c(t)$ is bigger, the feature map will be more likely to change locally following input sample $x(t)$. This reduces the remember ability the previous learning times of the network.

Therefore, TE may depends on initializing the weight vector of neurons if q is too large or depends on the order of the input samples if q is too small. It is notable that the initial weight vector of neurons and the order of the input sample are selected randomly. Therefore, the topographic learning ability of network is best when parameter q is not too small or too large.

3.2 Parameter p

When the parameter q is fixed, if the parameter p increases, the value of function $h_{ci}(t)$ of the neurons that near the BMU will increase gradually to 1, i.e., the number of neighbors around the BMU that are adjusted similar with BMU will extend. This is the cause of QE increasingly. If the parameter p is too large, the feature map tends to change locally according to the input sample from the closest training times (similar to the case that parameter q is too small). However, TE may vary slightly because TE is conducted by BMU_1 and BMU_2.

Figure 4 illustrates original neighborhood function $h_{ci}(t)$ (with $q = 0.5$ and $p = 2$) and adjusted neighborhood function $h_{ci}(t)$ (with $q = 4$ and $p = 1, 2, 3, 4, 5, 6$) in case of $N_c(t) = 10$.

In case of $p = 1$, the graph $h_{ci}(t)$ is similar to the case of $q = 8$, 12 in **Figure 3**, i.e., the smallest QE compared to the case of $p > 1$, but TE is unreliable due to depend on initializing the weight vector of neurons.

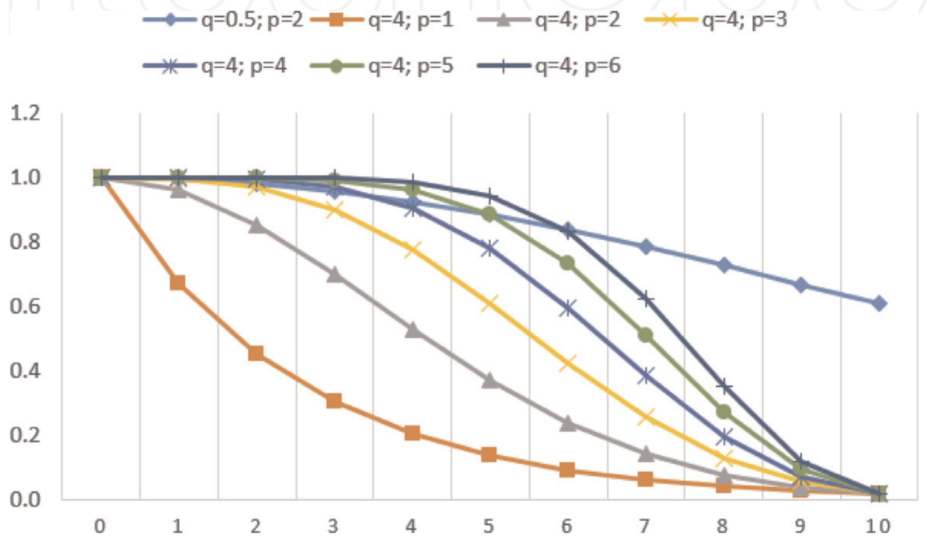

Figure 4.
Illustrates function $h_{ci}(t)$ after changing the value of p.

Therefore, the adjustment of parameter p has no significant impact on improving the quality of the feature map of SOM, but the parameter q has positive significance in improving the quality of the feature map of SOM. The bigger the parameter q is, the smaller QE is. However, q reaches the most appropriate value when TE is the smallest. Therefore, we recommend the neighborhood function $h'_{ci}(t)$ with an adjustable parameter as follows:

$$h'_{ci}(t) = \exp\left[-q\frac{\|r_c - r_i\|^2}{{N_c}^2(t)}\right] \tag{9}$$

with the parameter q can be adjusted depending on each the dataset to achieve better quality of feature map.

4. Experiments

We have conducted experiments for 12 published datasets, including: XOR (data samples are distributed within the XOR operation), Aggregation, Flame, Pathbased,

q	0.5	1	2	4	8	12
XOR	0.1890	0.1585	0.1299	0.1129	0.0902	0.0810
	0.0318	**0.0223**	0.0273	0.0427	0.0705	0.0925
Aggregation	5.9702	5.0643	4.0276	2.9340	2.2819	1.8472
	0.0549	0.0362	0.0294	**0.0245**	0.0424	0.0678
Flame	2.1839	1.9512	1.5194	1.1822	0.9129	0.8206
	0.0700	0.0567	0.0407	**0.0393**	0.0479	0.0833
Pathbased	4.5859	4.0427	3.2618	**2.4779**	1.9392	1.7401
	0.0561	0.0433	0.0373	**0.0315**	0.0434	0.0794
Spiral	4.7595	4.1719	3.4675	2.9239	2.2975	2.0085
	0.0543	0.0404	**0.0284**	0.0364	0.0413	0.0564
Jain	5.2745	4.4829	3.5726	2.3559	1.6236	1.5234
	0.0513	0.0395	0.0313	**0.0269**	0.0443	0.0637
Compound	4.4205	3.7595	3.1508	2.5672	1.8323	1.7744
	0.0624	**0.0299**	0.0349	0.0400	0.0630	0.0690
R15	2.2226	2.0212	1.8005	1.4606	1.0730	0.9562
	0.0722	0.0631	0.0368	**0.0274**	0.0613	0.1162
D31	4.7676	4.1204	3.3943	2.4569	2.0055	1.6793
	0.0479	0.0352	0.0284	**0.0207**	0.0332	0.0394
Iris	0.7709	0.6430	0.5353	0.4403	0.3773	0.3494
	0.0739	**0.0548**	0.0689	0.0940	0.1196	0.1566
Vowel	2.7459	2.5736	2.3755	2.2005	1.9150	1.7468
	0.0537	0.0436	**0.0412**	0.0448	0.0494	0.0497
Zoo	1.5841	1.4421	1.2468	1.0912	0.9790	0.9156
	0.0343	0.0254	0.0169	**0.0104**	0.0162	0.0208

Table 1.
Experiment results when fixed parameter p = 2, change parameter q.

Spiral, Jain, Compound, R15, D31, Iris, Vowel and Zoo. The parameters were used in the experiment as follows: network size: 10×10; initial neighborhood radius: 10; initial learning rate: 1; number of training times: 20,000.

The experiments were conducted in two cases: case 1—fixed parameter p, changed parameter q; case 2—fixed parameter q, changed the parameter p.

Note: The results in **Tables 1** and **2** are the average value of 10 experiment times. The result of each dataset presented in two rows: the first row shows *QE* and the second row displays *TE*.

Case 1: Parameter p is fixed, parameter q changed.

Table 1 shows the experimental results with parameter $p = 2$ and change the value of parameter $q = 0.5, 2, 4, 8, 12$.

We can see that *QE* is in a reverse ratio to q, when q is bigger, *QE* is smaller, while *TE* reaches the minimum value with $q = 1, 2, 4$. This is especially true with the proposed analysis in Section 3.

The values in bold are the best results, in which: *TE* is the smallest, *QE* is also smaller than the case of using the original neighborhood function ($q = 0.5$) (column 2, **Table 1**).

p	1	2	3	4	5	6
XOR ($q = 1$)	0.1754	0.1587	0.1546	0.1518	0.1525	0.1513
	0.0534	0.0203	0.0225	0.0244	0.0238	0.0255
Aggregation ($q = 4$)	2.7895	3.0003	3.2722	3.6436	3.6100	3.8718
	0.0850	0.0300	0.0277	0.0273	0.0316	0.0282
Flame ($q = 4$)	1.1858	1.2105	1.2306	1.3158	1.4010	1.4209
	0.1438	0.0405	0.0284	0.0304	0.0331	0.0330
Pathbased ($q = 4$)	2.5458	2.4759	2.7586	2.8462	2.9400	2.9928
	0.1300	0.0313	0.0363	0.0351	0.0349	0.0304
Spiral ($q = 2$)	3.5976	3.4319	3.4334	3.4603	3.4926	3.5797
	0.0690	0.0290	0.0265	0.0290	0.0261	0.0264
Jain ($q = 4$)	2.3664	2.3519	2.7136	2.9018	3.1494	3.3035
	0.0896	0.0263	0.0270	0.0306	0.0402	0.0403
Compound ($q = 1$)	4.2063	3.7575	3.6224	3.4969	3.5082	3.4913
	0.0666	0.0291	0.0337	0.0340	0.0373	0.0398
R15 ($q = 4$)	1.3161	1.4406	1.5544	1.6498	1.6972	1.7376
	0.1055	0.0294	0.0367	0.0390	0.0454	0.0548
D31 ($q = 4$)	2.3832	2.4769	2.8137	2.9886	3.0686	3.1960
	0.0803	0.0199	0.0227	0.0238	0.0259	0.0284
Iris ($q = 1$)	0.7140	0.6382	0.6166	0.6002	0.5880	0.5849
	0.0665	0.0518	0.0555	0.0560	0.0572	0.0598
Vowel ($q = 2$)	2.3938	2.3715	2.4186	2.4310	2.4529	2.4627
	0.0635	0.0410	0.0416	0.0414	0.0429	0.0455
Zoo ($q = 4$)	1.1817	1.0912	1.1780	1.1954	1.2015	1.2131
	0.0366	0.0104	0.0182	0.0188	0.0176	0.0180

Table 2.
Experiment results when change parameter p, fixed parameter q.

Case 2: Parameter q is fixed, parameter p changes.

Table 2 shows the experimental results when fixes parameter q of each dataset corresponding to the best value of TE in **Table 1** and respectively change the value of $p = 1, 2, 3, 4, 5, 6$.

When $p = 1$: both QE and TE increase high.

When $p \geq 2$: TE tends to be stable or increase slightly when p rises. This shows that the parameter p is negligibly significance in improving the topographical quality when identifying suitable parameter q; QE tends to increase with the majority of datasets while increasing p (excepting for the dataset XOR, Compound and Iris, QE tends to decrease, but TE tends to increase). This suggests that, $p = 2$ is the best value.

From **Figures 5** to **16** are charts comparing the values of QE, TE when changing the parameters q and p, in which: figures on the left (a) are the results when fixing $p = 2$ and changing q; figures on the right (b) are the results when fixing q and changing p. Parameter q is selected by the corresponding value to achieve the smallest value of TE in figure (a).

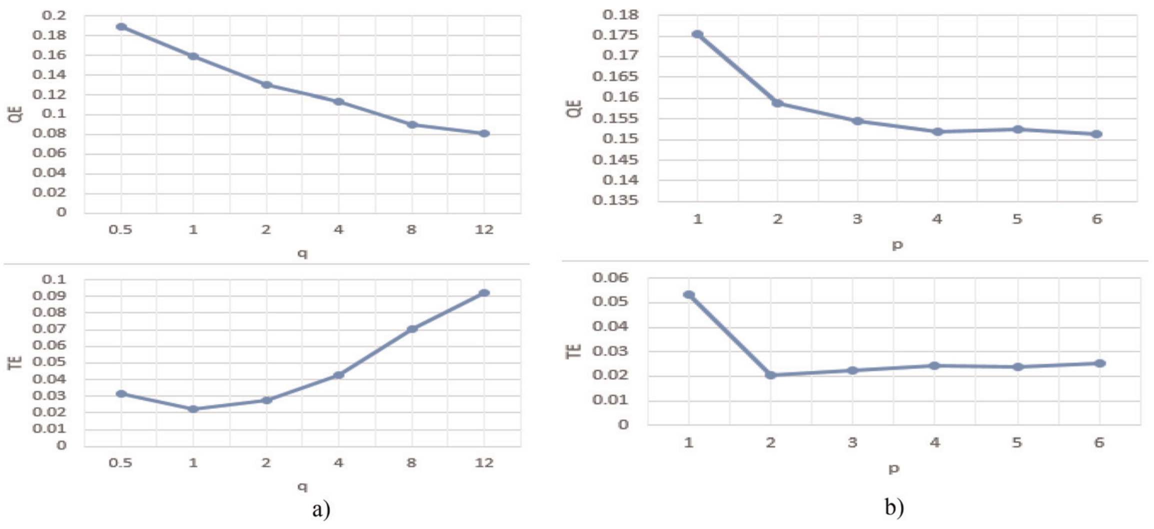

Figure 5.
XOR dataset. (a) p = 2 and q changes and (b) q = 1 and p changes.

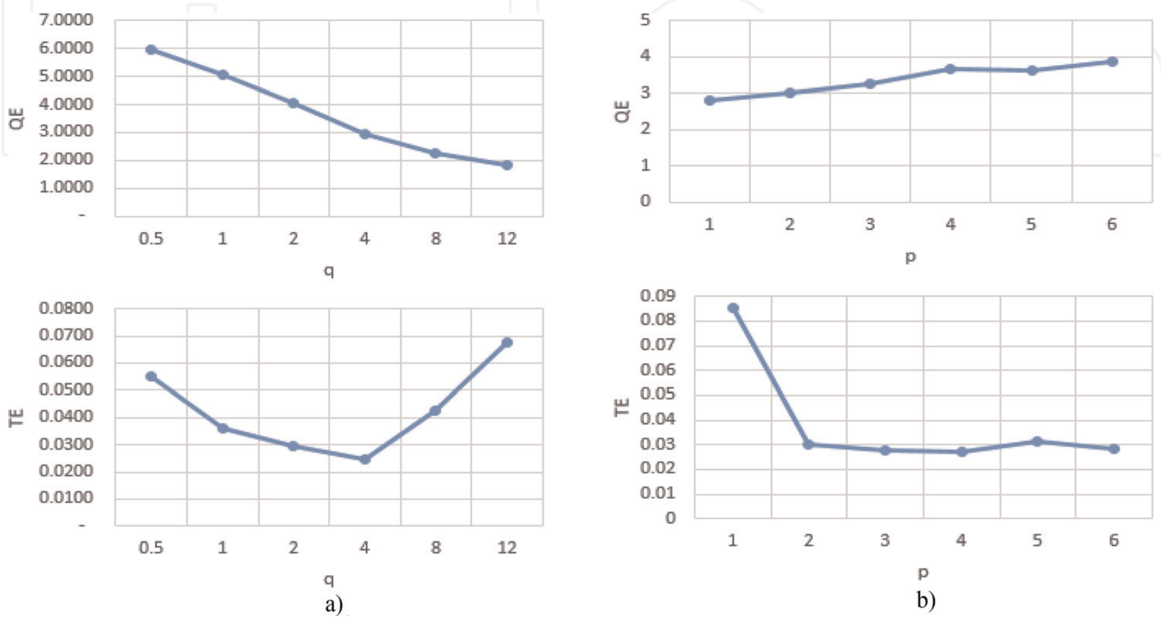

Figure 6.
Aggregation dataset. (a) p = 2 and q changes and (b) q = 4 and p changes.

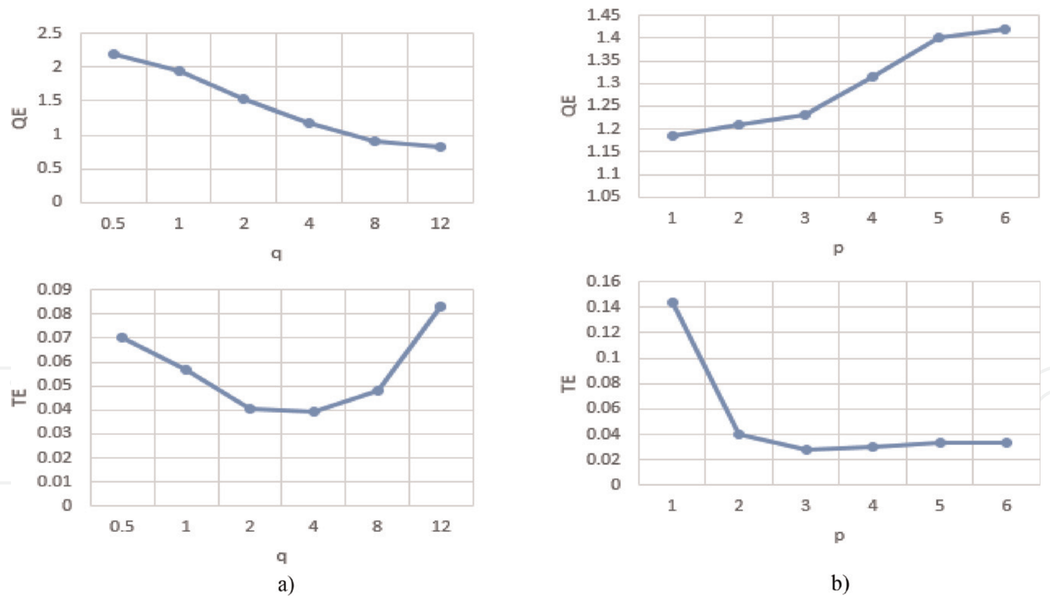

Figure 7.
Flame dataset. (a) p = 2 *and* q *changes and (b)* q = 4 *and* p *changes.*

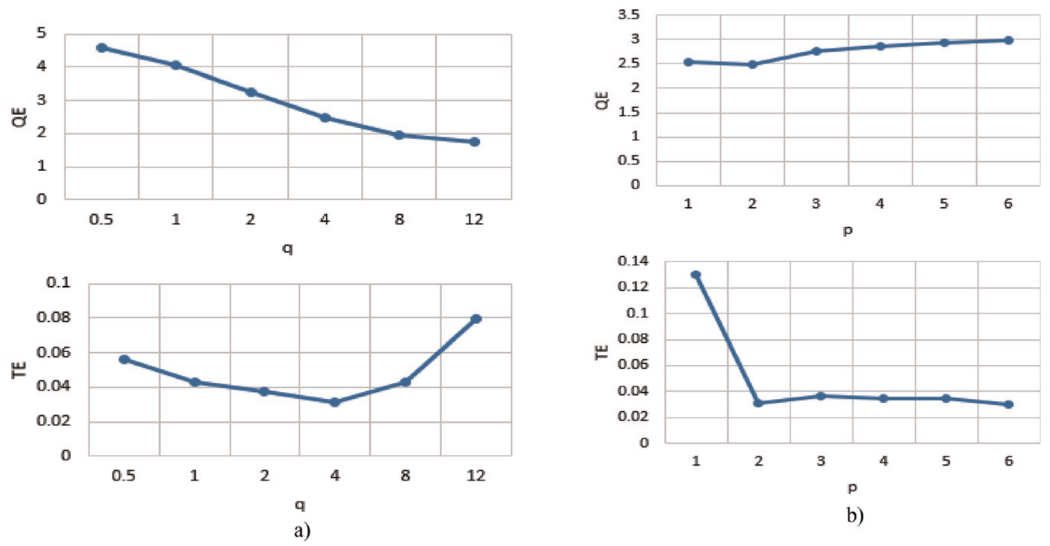

Figure 8.
Pathbased dataset. (a) p = 2 and q changes and (b) q = 4 and p changes.

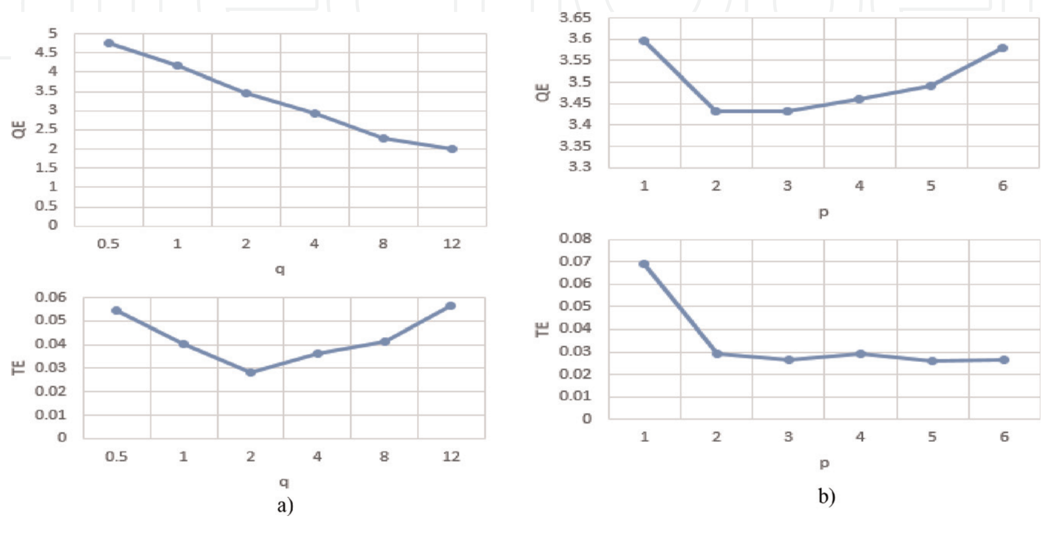

Figure 9.
Spiral dataset. (a) p = 2 and q changes and b) q = 2 and p changes.

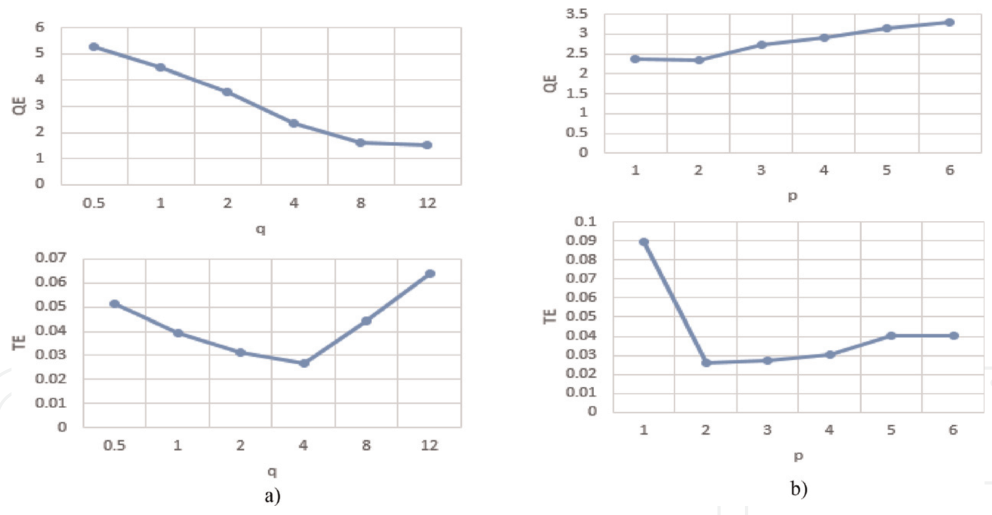

Figure 10.
Jain dataset. (a) p = 2 and q changes and (b) q = 4 and p changes.

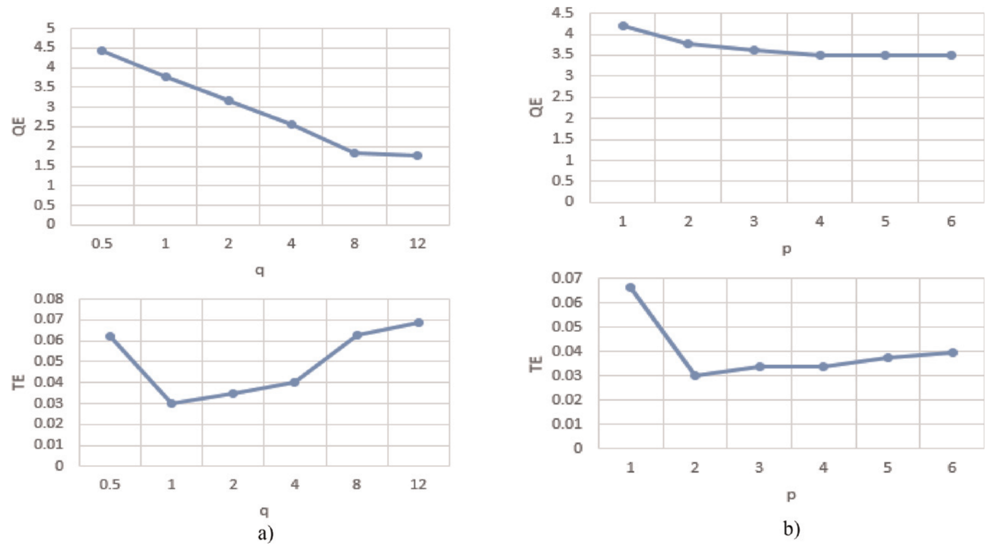

Figure 11.
Compound dataset. (a) p = 2 and q changes and (b) q = 1 and p changes.

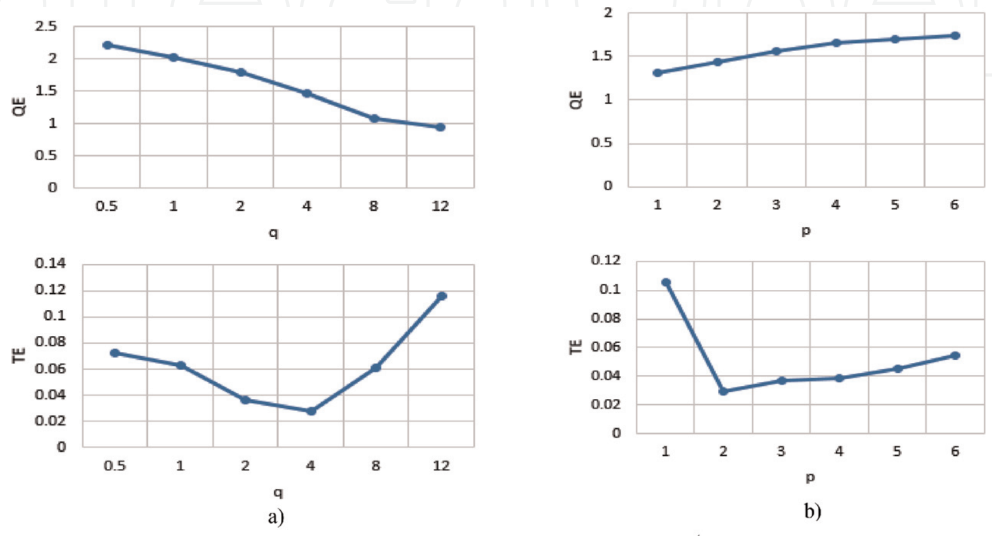

Figure 12.
R15 dataset. (a) p = 2 and q changes and (b) q = 4 and p changes.

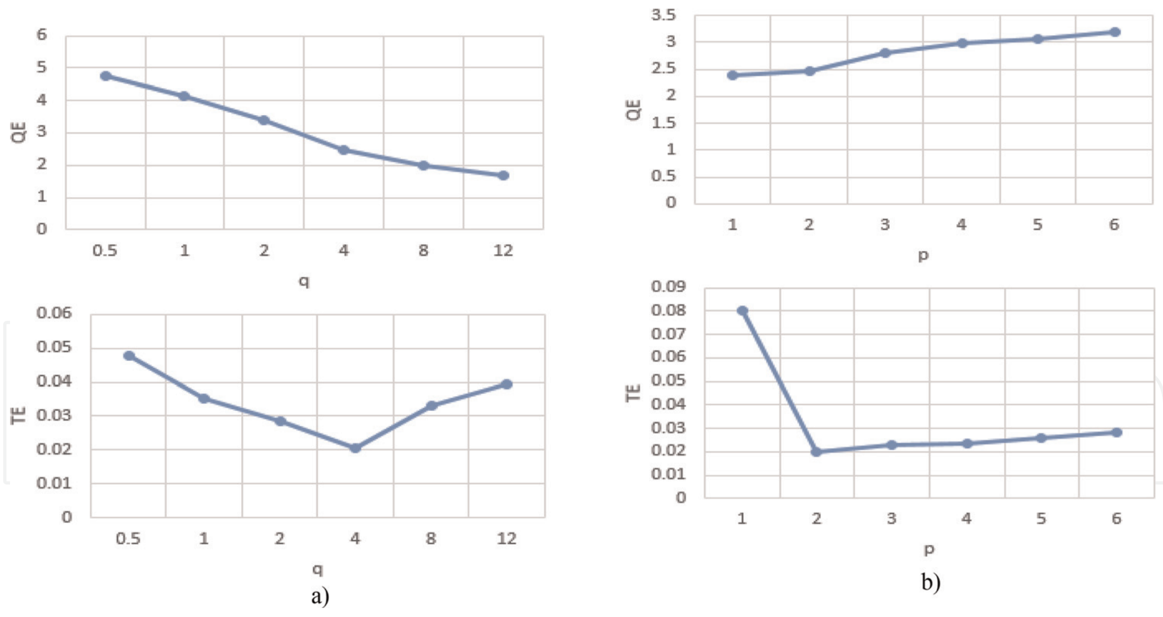

Figure 13.
D31 dataset. (a) p = 2 and q changes and (b) q = 4 and p changes.

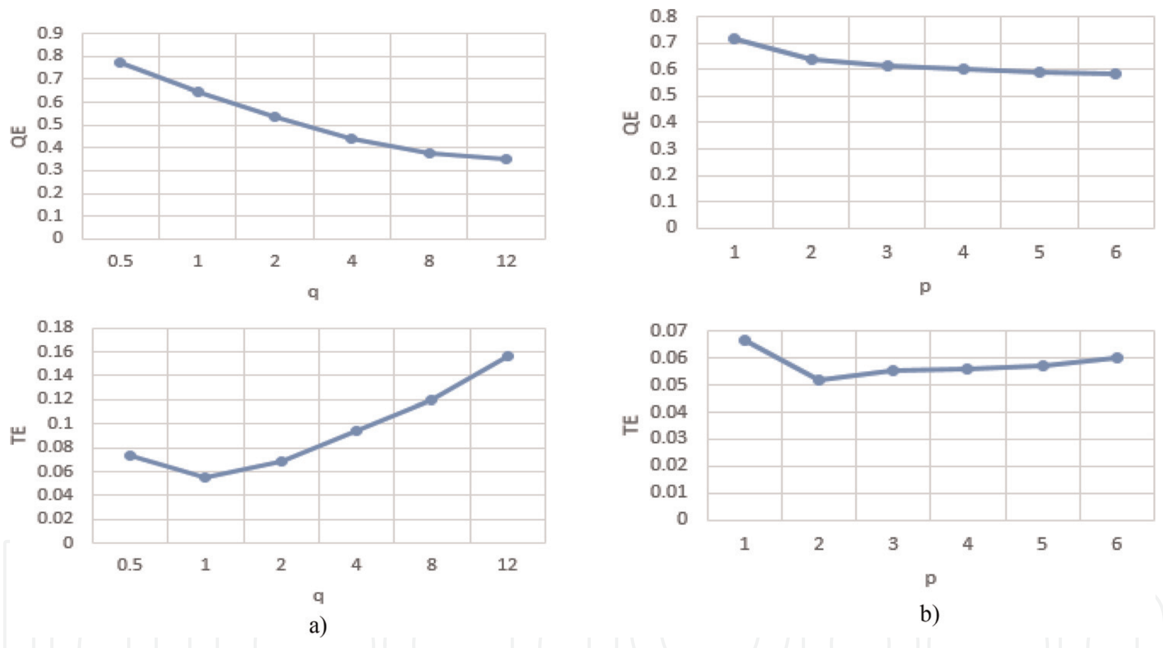

Figure 14.
Iris dataset. (a) p = 2 and q changes and (b) q = 1 and p changes.

When putting parameter $p = 2$ and changing parameter q, we see that the charts are similar (figure (a)—on the left), with *QE* is reduced gradually, *TE* reduced at first, then increased inversely with *QE* when parameter q increased gradually. *TE* reaches the lowest value when $q \in [10, 28]$.

When fixing parameter q and changing the parameter p, the charts also have similarities (figure (b)—on the right), including: *TE* is highest when $p = 1$; both graphs of *QE* and *TE* tend to stabilize or increase gradually with $p \geq 2$.

Conclusion: With $p = 2$ (default value), the adjustment of the parameter q has significantly impacted on the quality of the feature map. If q is bigger, the *QE* is smaller. However, *TE* is lowest when q is not too small or too large. Therefore, with $p = 2$, parameter q is the most suitable when its value is large enough to achieve the lowest value of *TE*. Conversely, if we have identified the most appropriate value of the parameter q, the parameter p has little significant impact on improving the quality of the feature map.

Table 3 shows the results of QE, TE when using neighborhood function $h'_{ci}(t)$ (with parameter $p = 2$ and q is determined for each dataset shown in **Table 2**) and some other neighborhood functions. Results show that the neighborhood function $h'_{ci}(t)$ achieved QE, TE smaller than the original Gaussian function, Bubbles function and asymmetric neighborhood function.

Note: The results in **Table 3** are the average value of 10 experiment times. The result of each dataset present in two rows: the first row shows QE and the second row displays TE.

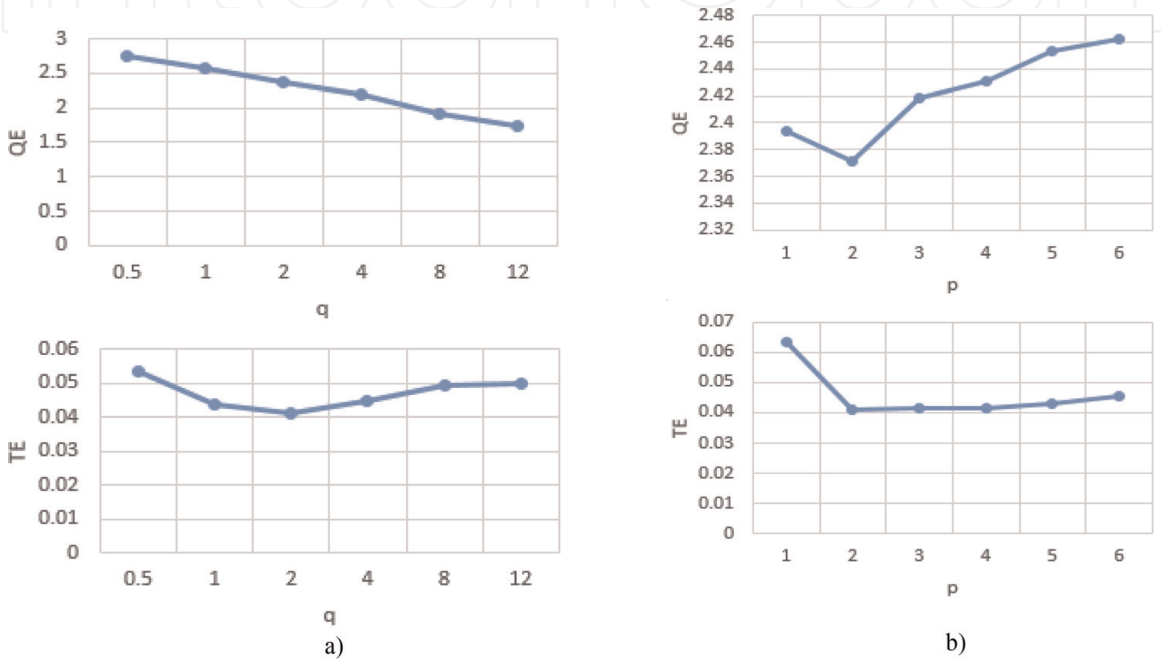

Figure 15.
Vowel dataset. (a) p = 2 and q changes and (b) q = 2 and p changes.

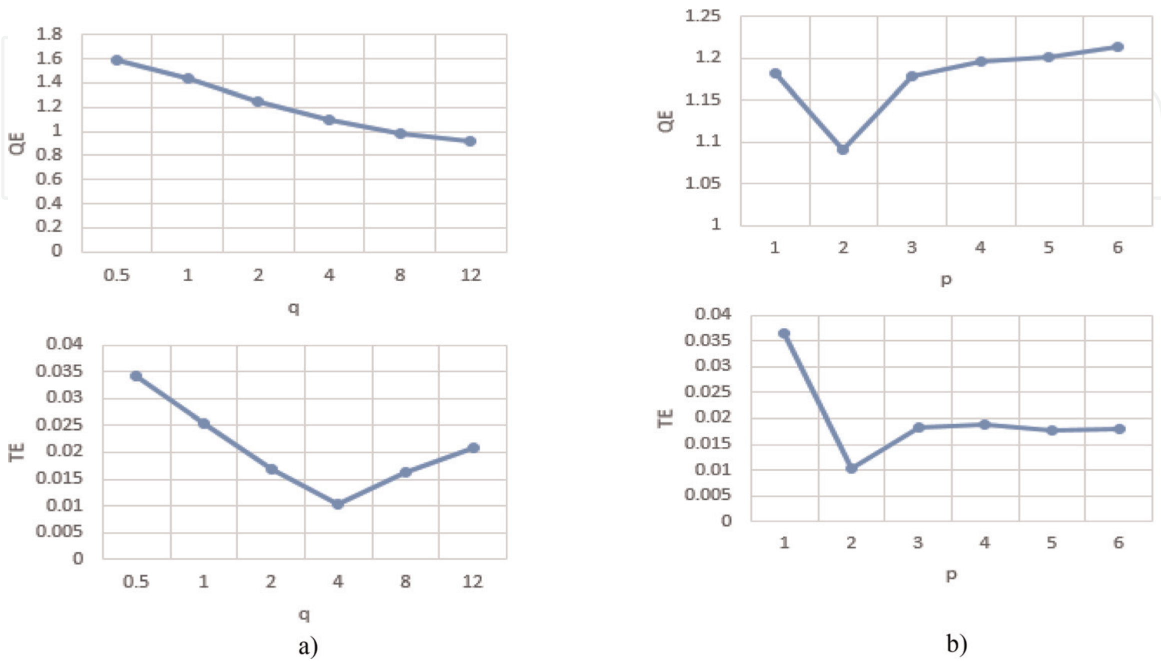

Figure 16.
Zoo dataset. (a) p = 2 and q changes and (b) q = 4 and p changes.

Dataset	$h_{ci}(t)$	$h'_{ci}(t)$	Bubble function	Asymmetric neighborhood function
XOR	0.1890	0.1585	0.2572	0.1808
	0.0318	0.0223	0.2708	0.4635
Aggregation	5.9702	2.9340	7.3092	4.9466
	0.0549	0.0245	0.1794	0.4476
Flame	2.1839	1.1822	2.6352	2.1916
	0.0700	0.0393	0.1642	0.6828
Pathbased	4.5859	2.4779	5.524	5.3888
	0.0561	0.0315	0.1981	0.2715
Spiral	4.7595	3.4675	5.6515	4.3775
	0.0543	0.0284	0.1502	0.6306
Jain	5.2745	2.3559	6.3026	5.4962
	0.0513	0.0269	0.2024	0.3172
Compound	4.4205	3.7595	5.5663	3.5529
	0.0624	0.0299	0.2199	0.4349
R15	2.2226	1.4606	2.5017	1.8911
	0.0722	0.0274	0.1384	0.6337
D31	4.7676	2.4569	5.6095	5.958
	0.0479	0.0207	0.2054	0.3506
Iris	0.7709	0.6430	1.001	0.9284
	0.0739	0.0548	0.2312	0.2610
Vowel	2.7459	2.3755	3.1022	2.8808
	0.0537	0.0412	0.1872	0.3965
Zoo	1.5841	1.0912	1.7182	1.7179
	0.0343	0.0104	0.2182	0.2210

Table 3.
Compares measures QE, TE of some neighborhood functions.

5. Conclusion

This chapter proposes the parameter for adjustment of the Gaussian symmetric neighborhood function. Our parameter adjusting method can reduce both *QE* and *TE* of the feature map. However, the value of parameter must be determined individually for each specific dataset. The improved Gaussian function is better than the original Gaussian function and some other neighborhood functions like Bubble function, asymmetric neighborhood function.

Author details

Le Anh Tu
Halong University, Vietnam

*Address all correspondence to: anhtucntt@gmail.com

References

[1] Lonsway B, Mulky AR. A self-organizing neural system for urban design. In: Proceedings of the Twenty First Annual Conference of the Association for Computer-Aided Design in Architecture; 11-14 October 2001; Buffalo (New York). pp. 386-391

[2] Arribas-Bel D, Schmidt CR. Self-organizing maps and the US urban spatial structure. Environment and Planning B: Urban Analytics and City Science. 2013;**40**(2):362-371

[3] Kropp J. A neural network approach to the analysis of city systems. Applied Geography. 1998;**18**(1):83-96

[4] Neme O, Pulido JRG, Neme A. Mining the city data: Making sense of cities with self-organizing maps. In: WSOM 2011: Advances in Self-Organizing Maps. Springer; 2011. pp. 168-177

[5] Mayaud JR, Anderson S, Tran M, Radić V. Insights from self-organizing maps for predicting accessibility demand for healthcare infrastructure. Urban Science. 2019;**3**(1):33. DOI: 10.3390/urbansci3010033

[6] Bauer H, Pawelzik K. Quantifying the neighborhood preservation of self organizing feature maps. IEEE Transactions on Neural Networks. 1992; **3**(4):570-579

[7] Kahraman C. In: Kahraman CE, editor. Computational Intelligence Systems in Industrial Engineering. 1st ed. Vol. 6. Atlantis Press; 2012. pp. 295-315

[8] Polani D. Measures for the organization of self-organizing maps. In: Studies in Fuzziness and Soft Computing. Vol. 78. Springer; 2002. pp. 13-44

[9] Uriarte E, Martín DF. Topology preservation in SOM. International Journal of Applied Mathematics and Computer Science. 2005;**1**(1):19-22

[10] Berglund E, Sitte J. The parameterless self-organizing map algorithm. IEEE Transactions on Neural Networks. 2006;**17**(2):305-316

[11] Kohonen T. Self-Organizing Maps. 3rd ed. Springer-Verlag; 2001

[12] Bauer H, Herrmann M, Villmann T. Neural maps and topographic vector quantization. Neural Networks. 1999;**12** (4-5):659-676

[13] Kiviluoto K. Topology preservation in self-organizing maps. In: Neural Networks, IEEE International Conference on (ICNN96), Volume 1; June 3-6 1996; Washington, DC: IEEE; 1996. pp. 294-299

[14] Mwasiagi JI, XiuBao H, XinHou W, Qing-dong C. The use of K-means and Kohonen self organizing maps to classify cotton bales. In: Beltwide Cotton Conferences (BWCC'07); 9-12 January 2007; New Orleans, Louisiana; 2007

[15] Flanagan JA. Self-organization in Kohonen's SOM. Neural Networks. 1996;**9**(7):1185-1197

[16] Germen E. A novel approach for learning rate in self orginizing map (SOM), anadolu University. Journal of Science and Technology — Application of Science and Engineering. 2018;**19**(1): 144-152

[17] Mulier F, Cherkassky V. Statistical analyses of self-organization. Neural Networks. 1995;**8**(5):717-727

[18] Wang S, Wang H. Knowledge discovery through self-organizing maps: Data visualization and query processing. Knowledge and Information Systems. 2002;**4**(1):31-45

[19] Chattopadhyay M, Dan PK, Mazumdar S. Application of visual clustering properties of self organizing map in machine-part cell formation. Applied Soft Computing. 2012;**12**(2): 600-610

[20] Polzlbauer G. Survey and comparison of quality measures for self-organizing maps. In: Paralic J, Polzlbauer G, Rauber A, editors. In: Proceedings of the Fifth Workshop on Data Analysis (WDA-04); 24-27 June 2004; Sliezsky dom, Vysoke Tatry, Slovakia: Elfa Academic Press; 2004. pp. 67-82

[21] Sun Y. On quantization error of self-organizing map network. Neurocomputing. 2000;**34**(1-4):169-193

[22] Germen E. Increasing the topological quality of Kohonen's self organizing map by using a hit term. In: Neural Information Processing, Proceedings of the 9th International Conference on (ICONIP'02), Volume 2; 2002. pp. 930-934

[23] Germen E. Improving the resultant quality of Kohonens self organizing map using stiffness factor. In: Advances in Natural Computation, Lecture Notes in Computer Science (First International Conference, ICNC 2005), Volume 3610; August 27-29 2005; Changsha, China: Springer, Berlin, Heidelberg; 2005. pp. 353-357

[24] Neme A, Chavez E, Cervera A, Mireles V. Decreasing neighborhood revisited in selforganizing map. In: Artificial Neural Networks-ICANN 2008, Volume 5163; 3-6 September 2008; Prague, Czech Republic: Springer, Berlin, Heidelberg; 2008. pp. 671-679

[25] Neme A, Miramontes P. Self-organizing map formation with a selectively refractory neighborhood. Neural Processing Letters. 2014;**39**(1): 1-24

[26] Kamimura R. Input information maximization for improving self-organizing maps. Applied Intelligence. 2014;**41**(2):421-438

[27] Lopez-Rubio E. Improving the quality of self-organizing maps by self-intersection avoidance. IEEE Transactions on Neural Networks and Learning Systems. 2013;**24**(8):1253-1265

[28] Aoki T, Aoyagi T. Self-organizing maps with asymmetric neighborhood function. Neural Computation. 2007;**19**: 2515-2535

[29] Ota K, Aoki T, Kurata K, Aoyagi T. Asymmetric neighborhood functions accelerate ordering process of self-organizing maps. Physical Review. 2011; **83**(2-1):1-9

[30] Lee JA, Verleysen M. Self-organizing maps with recursive neighborhood adaptation. Neural Networks. 2002;**15**:993-1003