Micro-SOM: A Linear-Time Multivariate Microaggregation Algorithm Based on Self-Organizing Maps

Agusti Solanas, Arnau Gavalda, and Robert Rallo

Department of Computer Engineering and Mathematics,
Rovira i Virgili University,
Av.Països Catalans, 26,
43007 Tarragona, Catalonia, Spain
{agusti.solanas,arnau.gavalda,robert.rallo}@urv.cat

Abstract. The protection of personal privacy is paramount, and consequently many efforts have been devoted to the study of data protection techniques. Governments, statistical agencies and corporations must protect the privacy of the individuals while guaranteeing the right of the society to knowledge. Microaggregation is one of the most promising solutions to deal with this praiseworthy task. However, its high computational cost prevents its use with large amounts of data. In this article we propose a new microaggregation algorithm that uses self-organizing maps to scale down the computational costs while maintaining a reasonable loss of information.

Keywords: Self-Organizing Maps, Privacy, k-Anonymity, Microaggregation.

1 Introduction

The Information and Communications Technologies (ICT) pave the way for the storage and analysis of huge amounts of data. It is paramount that the collection and, specifically, the analysis of all these data take into account the privacy of individuals. Hence, there must be a balance between the right of the society to information and the right of individuals to privacy. Some common examples of information gathering and analysis using ICT are the following: (i) **E-commerce**: Commercial information that is collected from fidelity cards, e-commerce logs, sales software, etc. From this information, customer's habits can be inferred, so these data analysis can feed marketing strategies; (ii) **Statistical Data:** Statistical agencies collect data from individuals to analyze them and publish statistical reports. These data, which are kept in large databases, might be released to third parties such as marketing companies and, as a result, statistical agencies lose control on the data once they are released; (iii) **Location Information:** Location-based services use the location of the users to provide them with personalized services. These locations could be used to infer the habits of the users, thus, these data must be managed carefully; and (iv) **Internet search engines:** Internet advertising companies use browsing information of Internet users to provide them with personalized advertisements. This practice could be seen as a privacy invasion if users are not properly informed.

These examples above are just the tip of the iceberg. ICT users have to face these situations daily and, although privacy invasion is becoming a general practice, privacy is a fundamental right[1] that must be protected; Governments, public institutions and corporations must provide the means to do so.

Due to the importance of individual privacy protection, several techniques have been proposed to cope with this problem, namely noise addition [1], rank swapping [2], statistical obfuscation [3], microaggregation [4], etc. A more general and comprehensive survey on security-control techniques for statistical databases can be found in [5]. To the best of our knowledge, microaggregation is one of the most promising techniques for microdata protection because it achieves a good balance between information loss and disclosure risk. Unfortunately, microaggregation has a high computational cost (i.e. $O(n^2)$) that prevents its use on large data sets.

In this article, we propose a new linear-cost microaggregation algorithm (i.e. $O(n)$) based on self-organizing maps (SOM). Although microaggregation and SOM are well-known techniques in the fields of statistical disclosure control (SDC) and artificial intelligence respectively, this is the first time in which they are considered together to protect individual privacy.

The rest of the article is organized as follows: In Section 2 we provide the reader with some background on microaggregation and self-organizing maps. Our linear-time microaggregation algorithm is described in Section 3 and, the experimental results are shown in Section 4. Finally, in Section 5 we conclude the article with some final comments and future research lines.

2 Background

In this section we provide the reader with some basic concepts on microaggregation and k-anonymity (Section 2.1), and self-organizing maps (Section 2.2).

2.1 Microaggregation and k-Anonymity

k-Anonymity is an interesting approach to face the conflict between information loss and disclosure risk, suggested by Samarati and Sweeney [6,7].

Definition. *A protected data set is said to satisfy k-anonymity for $k > 1$ if, for each combination of attributes, at least k records exist in the data set sharing that combination.*

[1] "No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor or reputation ..." ***Universal Declaration of Human Rights.***

The original computational approach to achieve k-anonymity relied on suppressions and generalizations, so that minimizing information loss translates to reducing the number and/or the magnitude of suppressions and generalizations. Unfortunately, the original k-anonymity approach has several drawbacks when applied to numerical data, being the most relevant one the categorization of the numerical non-categorical data. It has been shown that k-anonymity can be achieved without categorization on numerical data by means of microaggregation [4].

Microaggregation is a statistical disclosure control (SDC) sub-discipline devoted to the protection of individual numerical data, also called *microdata.* It can be understood as a clustering problem. However, instead of considering the number of clusters as the main constraint, microaggregation is constrained by the size of the clusters. The microaggregation problem can be stated as follow:

> *Given a data set* $\mathbf{D}$ *with* $\mathbf{n}$ *records in a characteristic space* $\mathbb{R}^d$, *the problem consists in obtaining a* k*-partition*[2] $\mathcal{P}$ *of* $\mathbf{D}$, *so that the sum of squared errors in each part of* $\mathcal{P}$ *is minimized. Once* $\mathcal{P}$ *is obtained, each record of every part of* $\mathcal{P}$ *is replaced by the average record of the part.*

In order to determine the information loss produced by microaggregation the sum of squared errors (SSE) is used (*cf.* Expression 1).

$$\mathbf{SSE} = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (\mathbf{x_{ij}} - \mathbf{\bar{x}_i})'(\mathbf{x_{ij}} - \mathbf{\bar{x}_i}) \tag{1}$$

where g is the number of groups/parts, n_i is the number of records in the i-th group/part, $\mathbf{x_{ij}}$ is the j-th record in the i-th group/part and $\mathbf{\bar{x}_i}$ is the average record of the i-th group/part.

The SSE is generally compared with the total error (SST) defined in Equation 2 to obtain a measure of the information loss.

$$\mathbf{SST} = \sum_{i=1}^{n} (\mathbf{x_i} - \mathbf{\bar{x}})'(\mathbf{x_i} - \mathbf{\bar{x}}) \tag{2}$$

where n is the number of records in the data set, x_i is a record of the data set and $\bar{x}$ is the average record of the data set.

The microaggregation problem is known to be NP-hard [8] for multivariate data sets, therefore heuristic methods should be used to solve it. There is a plethora of methods to address the multivariate microaggregation problem. Some of them are based on building tree structures connecting the records in the data set and partition the tree to generate a k-partition [4][9]. Their main limitation is the high computational cost due to the computation of the distances between *all* pairs of records in the data set.

Instead of structuring the data in trees or graphs, an alternative way to tackle the problem is to build groups of similar records greedily. Examples of this approach are the Maximum Distance to Average Vector (MDAV) method [10], and

[2] A k-partition of $\mathbf{D}$ is a partition where its parts have, at least, k records of $\mathbf{D}$.

the Variable Maximum Distance to Average Vector (V-MDAV) method [11]. The main advantage of these approaches is their simplicity but their computational cost (i.e. $O(n^2)$) prevents their use with very large data sets.

Recent articles have tackled the problem of improving the k-partition obtained by previous microaggregation methods with the aim to reduce the information loss (but not the computational cost) [12][13]. On the other hand, microaggregation methods with low computational costs have been barely studied. A very recent example on this line can be found in [14], where the authors propose a linear-time microaggregation algorithm that works on the assumption that the input data are uniformly distributed.

The reduction of the computational cost of microaggregation algorithms in which the distribution of the input data is unknown is a key problem that we address in this article.

2.2 Self-Organizing Maps

The Self-Organizing Map (SOM) algorithm is based on an unsupervised competitive learning approach. The training process is entirely data-driven and map units compete to become specific detectors of certain data features. Each map unit is represented by an n-dimensional weight vector, where n is equal to the dimension of the input space. As in vector quantization, every weight vector describing a class is called a codebook. Each unit i has a topological neighborhood N_i determined by the shape of the SOM grid lattice which can be either rectangular or hexagonal. The number of units as well as their topological relations are defined during the initial map formation phase. The granularity (*i.e.* the size) of the map determines its accuracy and generalization capabilities. The number of units should usually be selected to accommodate all training data, with the neighborhood size controlling the smoothness and generalization of the mapping. The use of a hexagonal lattice is usually recommended, because all six neighbors of a unit are at the same distance, as opposed to the eight neighbors in a rectangular lattice configuration. The shape of the map grid should correspond to the shape of the data manifold whenever possible. To avoid the border effects in the mapping process, i.e., units with a reduced neighborhood, a periodic shape such as a torus is used. Additional details about the algorithm and its implementation can be found in [15]. The quality of a SOM can be evaluated from the resolution of the map and from the preservation of the topology of the native data set. The most important issue regarding the accuracy of the SOM projection is the "true" dimension of data. If it is larger than the dimension of the map grid the SOM may not follow the distribution of the data set. In this case topology preservation and map resolution become contradictory goals and cannot be simultaneously attained. When this occurs, a map with high resolution folds into itself and topology is broken. SOM resolution is measured by means of the average quantization error over the complete data set. SOM quality can also be estimated with a combined strategy that mixes both topology and resolution [16]. Once the training completes, the SOM approaches the clustering structure of the data. Clusters are formed by groups of codebook vectors which are close to each other compared with their distance to other vectors. The clustering structure of the input space can be visualized over the SOM grid by displaying the distances between these reference vectors. Several methods have been proposed to display the clustering structure; the most common is the unified distance matrix (*U-matrix*), i.e., the matrix of distances between each codebook vector and its neighbors. The visualization of these clusters could be enhanced by labeling the map with auxiliary data.

Since it is difficult to detect clusters by visual inspection of the U-matrix, SOM's reference vectors can in turn be clustered to detect coherent sets of units with similar structural characteristics. Simpler clustering algorithms such as the K-means procedure are used to cluster SOM vectors. The clustering parameters are optimized using some clustering quality criteria such as the minimization of the Davies-Bouldin index. This index is a function of the ratio between the sum of cluster compactness and inter-cluster separations [17] and permits the selection of the optimal number of clusters to obtain a good partitioning.

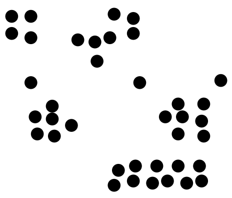
There exist many other variants and applications of the basic SOM reported in the literature. For applications see e.g. [18]. Possible variations include the use of neuron specific learning rates and neighborhood sizes, and growing map structures. The goal of all these variations is to enable the SOM to follow the topology of the underlying data set better and to achieve good quantization results [19]. The *Tree Structured* SOM [20] is a fast version of the SOM that consists in a set of layers that perform a complete quantization of the data space. Data from upper layers is used to train lower layers reducing the amount of distance calculations needed to find the winner unit. The *Minimum Spanning Tree* SOM [21] uses a tree structure as neighborhood function which defines the minimal set of connections needed to link together a related set of codebook vectors.

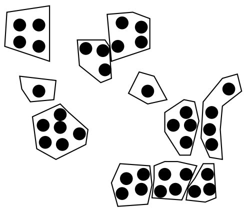
3 Our Proposal: Micro-SOM

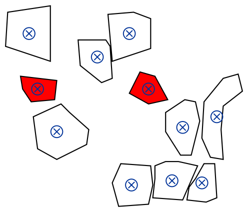
Given an input data set $\mathbf{D}$ with n records, we want to obtain a partition $\mathcal{P}$ of $\mathbf{D}$ so that each part of $\mathcal{P}$ has, at least, k records (i.e. a k-partition). After determining $\mathcal{P}$ we replace each record $r_i \in \mathbf{D}$ by the centroid of the part to which it belongs, thus obtaining a k-anonymous microaggregated data set $\bar{\mathbf{D}}$.

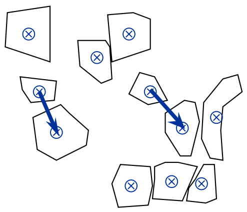
Our algorithm, called Micro-SOM, can be divided in the following three steps:

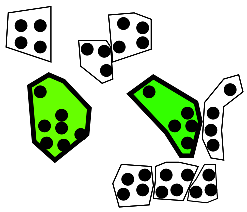
- **STEP 1 – Obtain a partition of D:** Previous methods compute the distance between all pairs of records (i.e. n^2 distances) so that they can heuristically determine subsets of $\mathbf{D}$ built of relatively close records (cf. Section 2.1 for further details and references). Instead of computing the n^2 distances, we use a self-organized map to group records in rough subsets (by doing so, we reduce the computational cost from $O(n^2)$ to $O(n)$). Note that the number of units/neurons is clearly smaller than the number of records. In this step, the algorithm proceeds as follows:
  - **STEP 1.1 – Standardize D:** In order to avert the influence of the difference between the scale of the attributes in the data set, it must be standardized before training the SOM (See Figure 1-a).


(a) Input data set


(b) Partition $\mathcal{P}'$ obtained by SOM


(c) Parts of $\mathcal{P}'$ with less than k elements (assume $k = 3$)


(d) Find the closest part to each small part


(e) Fuse small parts with their closest part

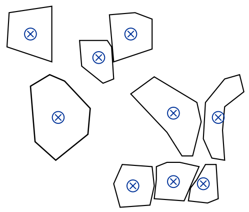

(f) Recompute centroids and microaggregate

Fig. 1. Graphical representation of different steps of the proposed algorithm

- **STEP 1.2 – Initialize and train the SOM:** The size of the map is heuristically determined by using the algorithm defined in [22]. By doing so, it can be guaranteed that the number of neurons in the map is some orders of magnitude smaller than the number of records (n). Also the dimensionality of the data is some orders of magnitude smaller than the number of records. Note that thanks to these differences in the orders of magnitude the computational cost of the algorithm is linear (i.e. $O(n)$) instead of quadratic (i.e. $O(n^2)$) [23,24]. Once the SOM is initialized, it is trained by using the standardized data.
- **STEP 1.3 – Assign each record to a map unit:** After training the SOM, we can assign each record in **D** to a map unit. Thus, the set of non-empty units represents a partition $\mathcal{P}'$ of **D** (See Figure 1-b).

– **STEP 2 – Obtain a k-partition:** In the previous step we get a partition, but some of the parts may have less than k elements. If that happens, low-cardinality parts must be fused with other parts so as to obtain a k-partition. To do so, the algorithm proceeds as follows:
  - **STEP 2.1 – Find low-cardinality parts:** All parts with less than k records are labelled as "low-cardinality parts" (See Figure 1-c). If there are low-cardinality parts then go to step 2.2, otherwise go to step 3.
  - **STEP 2.2 – Compute centroids and find the closest part to each low-cardinality part:** For each part, its centroid is computed. If there are low-cardinality parts, the closest part to each low-cardinality part is found by comparing the distances to its neighbors. Finding the closest

part is easy and computationally cheap due to the topological properties of SOM. (See Figure 1-d)

- **STEP 2.3 – Fuse each low-cardinality part with its closest part:** Using the information obtained in the previous step, low-cardinality parts are fused with their closest parts (See Figure 1-e). Before moving to the step 3, the algorithms goes back to step 2.1 to check whether low-cardinality parts still exist.

– **STEP 3 – Generate a microaggregated data set:** When we reach this step each record in the data set belongs to a part having at least k records. Also, the centroid of each part has been computed so that it can be used to replace the records belonging to that part. By replacing all records in $\mathbf{D}$ by the centroid of the parts to which they belong, a k-anonymous microaggregated data set $\bar{\mathbf{D}}$ is obtained and the algorithm terminates (See Figure 1-f).

4 Results

In order to test our solution, we have used the SOM Toolbox [22] under Matlab to implement the first step of the algorithm (i.e. Obtain a partition of $\mathbf{D}$). The second and third steps (i.e. Obtain a k-partition and generate a microaggregated data set) have been implemented using Java. All experiments have been carried out on an Intel Core2 Duo at 1.4Ghz with 2 GB of RAM running Windows XP.

With the aim to emphasize the low computational cost of the proposed algorithm, we have compared it with the μ-Approx algorithm [4] that is one of the most recent microaggregation algorithms proposed in the literature. We have used three real-life data sets that were used in the European CASC project [25] and two synthetic data sets with a larger number of records:

– The "Tarragona" data set contains 834 records with 13 numerical attributes.
– The "Census" data set contains 1080 records with 13 numerical attributes.
– The "EIA" data set contains 4092 records with 11 numerical attributes.
– The "Synthetic 1×10^4" data set contains 10.000 records with 3 numerical attributes uniformly distributed.
– The "Synthetic 1×10^5" data set contains 100.000 records with 3 numerical attributes uniformly distributed.

4.1 Information loss

The information loss (IL) is a widely used measure to determine how good a microaggregation algorithm is. The lower the IL the better the algorithm. Typically, the IL is defined as the ratio between SSE and SST in % (see Equation 3).

$$IL = 100 \times \frac{SSE}{SST} \tag{3}$$

We have studied the IL for the Tarragona, Census, EIA and Synthetic1×10^4 data sets varying the minimum cardinality parameter k from 2 to 10, which

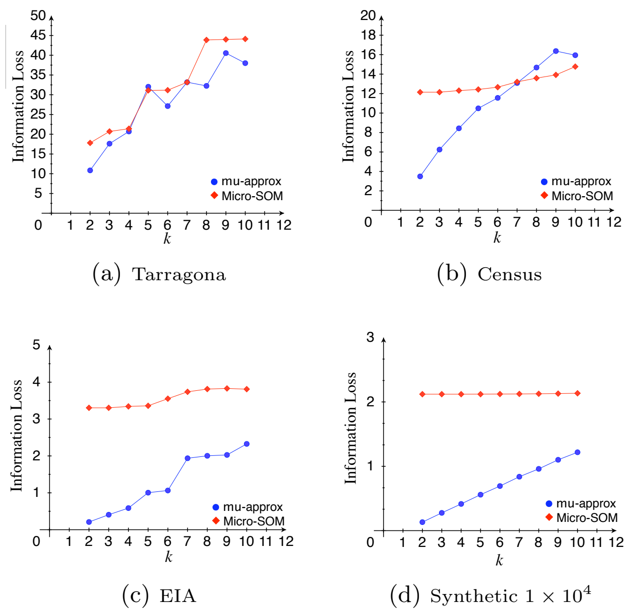


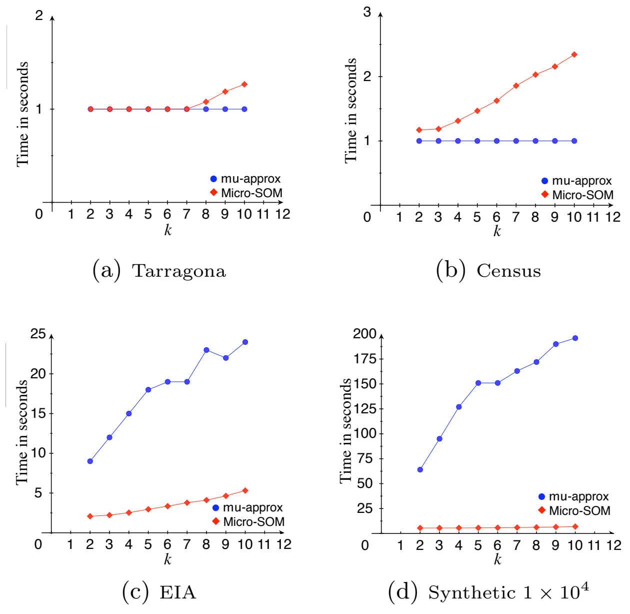
Fig. 2. Information loss for Micro-SOM and μ-Approx

are common values for k. The results are shown in Figure 2. From this figure, it is apparent that the behavior of both algorithms is pretty similar. Although μ-Approx achieves better results than Micro-SOM in general, the differences between them are very little specially when k grows. It is also clear that Micro-SOM is more resilient to the increase of the minimum cardinality parameter k, this is, it is more stable.

4.2 Time

The time required to microaggregate a data set is very relevant, specially when large data sets have to be managed. We have studied the time required to microaggregate the Tarragona, Census, EIA and Synthetic1×10^4 data sets varying the parameter k from 2 to 10, which are common values for k. The results are shown in Figure 3. From this figure we can conclude the following:

- When the number of records is small enough, μ-Approx performs very similar to Micro-SOM. See, for example, the results for the Tarragona data set (843 records) and the Census data set (1080 records).
- When the number of records in the data set grows, the performance of the μ-Approx algorithm degrades significantly while Micro-SOM properly copes with the microaggregation. See, for example, the results for the EIA data set (4092 records) and the Synthetic1×10^4 data set (1×10^4 records).



(a) Tarragona (b) Census

(c) EIA (d) Synthetic 1×10^4

Fig. 3. Computation time for Micro-SOM and μ-Approx

As it is shown in the next section, if the number of records grows enough, the μ-Approx algorithm is unable to microaggregate them due to its quadratic (i.e. $O(n^2)$) computational cost.

4.3 Results with Large Data Sets

In the previous sections we have compared the μ-Approx algorithm with our proposal. Unfortunately, the μ-Approx algorithm cannot manage a data set consisting of 1×10^5 records because it cannot cope with a matrix of 1×10^{10} distances (i.e. the computer runs out of memory). Consequently, we only summarize in Table 1 the results that Micro-SOM has obtained.

Table 1. Synthetic 1×10^5

k	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
SSE	7632.0	7632.3	7632.7	7633.0	7633.5	7632.7	7631.8	7631.5	7631.9
SST	3×10^5	3×10^5	3×10^5	3×10^5	3×10^5	3×10^5	3×10^5	3×10^5	3×10^5
IL	2.544	2.544	2.544	2.544	2.544	2.544	2.544	2.544	2.544
Time (s)	96.406	101.812	110.953	119.328	131.468	149.844	163.235	180.906	203.438

5 Conclusions and Further Work

We have presented Micro-SOM, a new microaggregation algorithm that uses self-organized maps to scale down the computational costs related to the computation of distances between all pairs of records in a data set. To the best of our knowledge, this is the first time in which a microaggregation method is fused with an artificial neural network.

The obtained results show that the information loss of Micro-SOM is similar to the information loss of μ-Approx that is one of the best microaggregation methods in the literature. The main contribution of Micro-SOM is its low-computational cost. From the results given in the previous section, it is apparent that Micro-SOM scales linearly with the number of elements in the data set and clearly outperforms the μ-Approx algorithm. We can conclude that globally (i.e. IL + Time) Micro-SOM is better than μ-Approx especially for large data sets.

In the near future we plan to explore the following lines:

– Use a hierarchical SOM to refine the partition obtained in STEP 1.
– Study the influence of different map topologies on the quality of the microaggregated data file.

References

1. Brand, R.: Microdata protection through noise addition. In: Inference Control in Statistical Databases, From Theory to Practice, London, UK, pp. 97–116. Springer, Heidelberg (2002)
2. Moore Jr., R.: Controlled data-swapping techniques for masking public use microdata sets. Technical report, Statistical Research Division Report Series, RR 96-04, US Bureau of the Census, Washington D.C. (1996)
3. Burridge, J.: Information preserving statistical obfuscation. Statistics and Computing 13(4), 321–327 (2003)
4. Domingo-Ferrer, J., Sebé, F., Solanas, A.: A polynomial-time approximation to optimal multivariate microaggregation. Comput. Math. Appl. 55(4), 714–732 (2008)
5. Adam, N.R., Worthmann, J.C.: Security-control methods for statistical databases: a comparative study. ACM Comput. Surv. 21(4), 515–556 (1989)
6. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)
7. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge Based Systems 10(5), 557–570 (2002)
8. Oganian, A., Domingo-Ferrer, J.: On the complexity of optimal microaggregation for statistical disclosure control. Statistical Journal of the United Nations Economic Comission for Europe 18(4), 345–354 (2001)
9. Laszlo, M., Mukherjee, S.: Minimum spanning tree partitioning algorithm for microaggregation. IEEE Transactions on Knowledge and Data Engineering 17(7), 902–911 (2005)
10. Domingo-Ferrer, J., Torra, V.: Ordinal, continuous and heterogenerous k-anonymity through microaggregation. Data Mining and Knowledge Discovery 11(2), 195–212 (2005)

11. Solanas, A., Martínez-Ballesté, A.: V-MDAV: Variable group size multivariate microaggregation. In: COMPSTAT 2006, Rome, pp. 917–925 (2006)
12. Solanas, A., Martinez-Balleste, A., Mateo-Sanz, J.M., Domingo-Ferrer, J.: Multivariate microaggregation based on genetic algorithms. In: 3rd International IEEE Conference on Intelligent Systems, pp. 65–70 (2006)
13. Martinez-Balleste, A., Solanas, A., Domingo-Ferrer, J., Mateo-Sanz, J.: A genetic approach to multivariate microaggregation for database privacy. In: IEEE 23rd International Conference on Data Engineering Workshop, April 17-20, pp. 180–185 (2007)
14. Solanas, A., Pietro, R.: A linear-time multivariate micro-aggregation for privacy protection in uniform very large data sets. In: Torra, V., Narukawa, Y. (eds.) MDAI 2008. LNCS (LNAI), vol. 5285, pp. 203–214. Springer, Heidelberg (2008)
15. Kohonen, T.: The self-organizing map. Proceedings of the IEEE 78(9), 1464–1480 (1990)
16. Kaski, S., Lagus, K.: Comparing self-organizing maps. In: Vorbrüggen, J.C., von Seelen, W., Sendhoff, B. (eds.) ICANN 1996. LNCS, vol. 1112, pp. 809–814. Springer, Heidelberg (1996)
17. Davies, D., Bouldin, D.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 2(1), 224–227 (1979)
18. Kangas, J.: Increasing the error tolerance in transmission of vector quantized images by self-organizing map. In: FogelmanSoulie, F., Gallinari, P. (eds.) Proc. ICANN 1995, Int. Conf. on Artificial Neural Networks, pp. 287–291 (1995)
19. Vesanto, J.: SOM-Based data visualization methods. In: Intelligent Data Analysis, vol. 3, pp. 111–126 (1999)
20. Koikkalainen, P., Oja, E.: Self-organizing hierarchical feature maps. In: International Joint Conference on Neural Networks (IJCNN 1990), pp. 279–284 (1990)
21. Kangas, J., Kohonen, T., Laaksonen, J.: Variants of the Self-organizing map. IEEE Transactions on Neural Networks 1(1), 93–99 (1990)
22. Vesanto, J., Himberg, J., Alhoniemi, E., Parhankangas, J.: Self-organizing map in Matlab: the SOM toolbox. In: Matlab DSP Conference, pp. 35–40 (1999)
23. Roussinov, D.G., Chen, H.: A scalable self-organizing map algorithm for textual classification: A neural network approach to thesaurus generation. Communication Cognition and Artificial Intelligence 15, 81–112 (Spring 1998)
24. Vesanto, J.: Neural network tool for data mining: Som toolbox. In: Proceedings of Symposium on Tool Environments and Development Methods for Intelligent Systems (TOOLMET 2000), Oulu, Finland, Oulun yliopistopiano, pp. 184–196 (2000)
25. Brand, R., Domingo-Ferrer, J., Mateo-Sanz, J.: Reference data sets to test and compare SDC methods for protection of numerical microdata. European Project IST-2000-25069 CASC (2002)