Single pass clustering for large data sets

Nikolai Alex[1], Barbara Hammer[2], Frank Klawonn[1]
1 - University of Applied Science Braunschweig/Wolfenbüttel, Department of Computer Science
2 - Clausthal University of Technology, Department of Informatics
Keywords: clustering, neural gas, k-means, large data sets, batch optimisation

***Abstract*— The presence of very large data sets poses new problems to standard neural clustering and visualisation algorithms such as Neural Gas (NG) and the Self-Organising-Map (SOM) due to memory and time constraints. In such situations, it is no longer possible to store all data points in the main memory at once and only a few, ideally only one run over the whole data set is still affordable to achieve a feasible training time. In this contribution we propose single pass extensions of the classical clustering algorithms NG and fuzzy-k-means which are based on a simple patch decomposition of the data set and fast batch optimisation schemes of the respective cost function. The algorithms maintain the benefits of the original ones including easy implementation and interpretation as well as large flexibility and adaptability because of the underlying cost function. We demonstrate the efficiency of the approach in a variety of experiments.

1 Introduction

The self organising map as proposed by Kohonen [12] constitutes an indispensable tool for visualisation and mining high dimensional data sets with numerous applications ranging from web data and picture processing to robotics and telecommunication. For clustering, a variety of alternatives exist such as the classical k-means algorithm, fuzzy extensions thereof, and neural gas [7, 9, 14]. Unlike SOM, which focuses on visualisation and topographic mapping using a fixed lattice structure, these alternatives do not assume a prior topology of neurons such that the resulting clustering follows the given data as accurately as possible. K-means, fuzzy-k-means, and neural gas (NG) have a clear mathematical foundation by means of a cost function which is optimised in batch (for k-means and fuzzy-k-means) or online (for NG) mode.

In recent years, the problem of mining very large data sets becomes more and more pronounced in diverse areas such as document and web mining, geoinformation and remote sensing, or bioinformatics and medicine. In these areas, massive volumes of data arise nearly on a daily base which have to be preprocessed and mined to allow further processing and human inspection. Clustering constitutes one valuable tool to compress such data in a meaningful way. Often, in these cases, data sets do no longer fit into the main memory and are stored in a distributed way in several databases or on several servers. In these cases, data access is costly and only a few, ideally at most one pass through the data set is still affordable. Thereby, only a fraction of the data fits into the main memory such that iterative approaches have to be used. Due to these circumstances, researchers have worked on modifications of various clustering algorithms such that they run in a single or few passes over the data and such that they require only a priorly fixed amount of allocated memory.

Most work in this area can be found in the context of heuristic (possibly hierarchical) clustering on the one side and classical k-means on the other side. Heuristic algorithms often directly assign data consecutively to clusters based on the distance within the cluster and allocate new clusters as required. Several popular methods include CURE, STING, and BIRCH [5, 17, 18]. Naturally, document ordering has an influence on the outcome of these algorithms, however, it is in practice not very pronounced as demonstrated in the work [11]. These methods, however, do not rely on a cost function such that e.g. an adaptation to relational data or the incorporation of label information into the clustering become difficult [3, 6].

Extensions of k-means clustering can be distinguished into methods which provide guarantees on the maximum difference of the result from classical k-means, such as presented in the approaches [4, 10]. However, these variants use resources which scale in the worst case with a factor depending on N (N being the number of points) with respect to memory requirements or passes through the data set. Alternatives are offered by variants of k-means which do not provide approximation guarantees, but which can be strictly limited with respect to space requirements and time. An early approach has been proposed in [1]: data are clustered consecutively in small patches, whereby the characteristics of the data and the possibility to compress subsets of data are taken into account. A simpler although almost as efficient method has been proposed in [2]: Standard k-means is performed consecutively for patches of the data whereby each new patch is enriched by the prototypes obtained in the previous patch. A sufficient statistics of the outcome of the last run can thereby easily be updated in a consecutive way, such that the algorithm provides cluster centres after only one pass through the data set, thereby processing the data consecutively in patches of predefined fixed size.

In this article, we extend this idea to fuzzy-k-means clustering and neural gas, two clustering algorithms which, unlike k-means, are very robust to initialisation. Both clustering algorithms can be derived from a cost function.

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

Thereby, we consider batch optimisation of NG as described e.g. in [3]. Patch clustering provides results which are competitive or only slightly inferior to the standard batch variants with respect to cluster formation whereby only a priorly limited memory is required and the scale of the convergence rate is faster. Patch clustering also opens the way towards parallel and distributed versions of these algorithms. Further, since patch clustering is based on the same cost function as the original clustering variant, the methods can easily be extended to incorporate label information or to deal with relational data.

2 Batch clustering

Given training data $\vec{x}_1, \ldots, \vec{x}_N \in \mathbb{R}^n$, the goal of prototype based clustering is to find prototypes $\vec{w}_1, \ldots, \vec{w}_k \in \mathbb{R}^n$ which represent the data points as accurately as possible. A common goal is the minimisation of the quantisation error

$$\frac{1}{2} \cdot \sum_{ij} \xi_i(\vec{x}_j)(\vec{w}_i - \vec{x}_j)^2$$

whereby $\xi_i(\vec{x}_j) \in \{0, 1\}$ characterises the receptive field of prototype $\vec{w}_i$, i.e. it is 1 iff the prototype $\vec{w}_j$ is closest to data point $\vec{x}_i$ as measured in the Euclidean distance. K-means (KM) clustering directly optimises the quantisation error by an iterative optimisation of the assignments $\xi_i(\vec{x}_j)$, mapping a data point to its respective closest prototype, and an optimisation of the prototype locations $\vec{w}_i = \sum_j \xi_i(\vec{x}_j)\vec{x}_j / \sum_j \xi_i(\vec{x}_j)$ [7]. This algorithm converges in a finite number of steps towards a local optimum of the quantisation error.

Since this cost function is multimodal, k-means is very sensitive to initialisation of the prototypes. A variety of alternatives has been proposed to overcome this problem and to extend k-means by additional useful information about the data, if appropriate. Fuzzy-k-means (FKM) clustering considers the objective function

$$\frac{1}{2} \cdot \sum_{ij} \chi_i(\vec{x}_j)^d (\vec{w}_i - \vec{x}_j)^2$$

where $d \geq 2$ is the fuzzifier (often chosen as $d = 2$), and the values $\chi_i(\vec{x}_j) \in [0, 1]$ constitute fuzzy assignments of the data points to prototypes which are no longer crisp but elements of the unit interval and which are optimised under the constraint $\sum_i \chi_i(\vec{x}_j) = 1$ for all j. An iterative optimisation of assignments and prototype locations using Lagrange optimisation yields the formulas

$$\chi_i(\vec{x}_j) = \frac{(\vec{w}_i - \vec{x}_j)^{-2/(d-1)}}{\sum_l (\vec{w}_l - \vec{x}_j)^{-2/(d-1)}}$$

for the assignments and

$$\vec{w}_i = \sum_j \chi_i(\vec{x}_j)^d \vec{x}_j / \sum_j \chi_i(\vec{x}_j)^d$$

for the prototype locations. In this simple form, fuzzy-k-means converges to a local optimum or saddle point of the cost function [9]. Unlike k-means, fuzzy-k-means is less sensitive to initialisation because of the gradual assignment of a data point to all prototypes; further, it provides more subtle fuzzy assignments which can give some clue about the clarity of the assignment.

Neural gas prevents initialisation sensitivity by means of neighbourhood cooperation but it uses crisp assignments [14]. It optimises the cost function (neglecting constant factors)

$$\frac{1}{2} \cdot \sum_{ij} h_\lambda(r_i(\vec{x}_j))(\vec{w}_i - \vec{x}_j)^2$$

where $r_i(\vec{x}_j) = |\{\vec{w}_l \mid (\vec{w}_l - \vec{x}_j)^2 \leq (\vec{w}_i - \vec{x}_j)^2\}|$ denotes the rank of prototype $\vec{w}_i$ measured according to the distance from $\vec{x}_j$ and $h_\lambda(t) = \exp(-t/\lambda^2)$ constitutes an exponential weighting function of the ranks. NG is usually optimised in an online mode by means of a stochastic gradient descent which iteratively adapts all neurons according to a chosen data point $\vec{x}_j$ by means of

$$\triangle w_i \sim -h_\lambda(r_i(\vec{x}_j))(\vec{w}_i - \vec{x}_j)\,.$$

Batch optimisation constitutes an alternative fast optimisation scheme which can be interpreted as Newton optimisation and which convergence in a finite number of steps towards a local optimum of the cost function. It iteratively assigns ranks to the neurons according to their distance from each data point $r_i(\vec{x}_j)$ and computes optimum prototype locations

$$\vec{w}_i = \sum_j h_\lambda(r_i(\vec{x}_j))\vec{x}_j / \sum_j h_\lambda(r_i(\vec{x}_j))\,.$$

Thereby, the neighbourhood range λ is annealed after every cycle towards zero. Due to the neighbourhood cooperation measured in terms of (scale free) ranks, neural gas is a very robust and initialisation insensitive classifier. In addition, it provides information about the data topology by linking every two prototypes iff they constitute the first two winners for at least one data point, i.e. they possess adjoining receptive fields within the data manifold.

Note that SOM does not possess a cost function, but a variant thereof as proposed by Heskes [8]. Thus, batch optimisation of SOM (using a fixed Euclidean or hyperbolic lattice structure, as appropriate) constitutes an alternative clustering method which provides visualisation of data simultaneous to clustering [13, 16].

Unlike online variants, NG requires only few runs over the data set. Since convergence is quadratic, the typical number of epochs until convergence is of order at most $\sqrt{N}$, N being the number of points, for all batch optimisation schemes. Online clustering typically requires a linear number of passes over the data set. One update step of batch clustering has an effort of order roughly Nk, k denoting the number of prototypes, whereby we are neglecting the complexity of computing the weighting terms (for

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

NG and fuzzy-k-means) and sorting (for NG). (The latter is reasonable for later runs of the algorithm where a comparably small contribution of data points to distant neurons can be observed such that only a constant number of neighbours must effectively be computed for a sufficient approximation of the result.) Thus, the overall time complexity of order $kN\sqrt{N}$ results for batch clustering.

3 Patch clustering for large data sets

Batch clustering requires all training data to be stored in the main memory which becomes infeasible for very large data sets. The article [2] proposes a simple and efficient strategy for k-means clustering with restricted buffer where data are processed consecutively in patches of predefined size. Here we transfer this strategy to NG and fuzzy-k-means.

Assume a fixed patch size P is chosen such that a number of P examples fits into the buffer. The main idea is to subsequently process patches of size P by batch optimisation, thereby enlarging the data set by patterns which stem from a sufficient statistic of the clusters obtained in the previous run. A clustering is represented by the cluster centres which is weighted according to the number of data points assigned to it. Note that, with respect to the quantisation error, an optimum cluster centre is represented by the mean of data points assigned to it. This is exactly reached in the convergence phase of NG; for fuzzy-k-means, the computed prototypes may slightly differ due to the fuzzification, however, we will also represent clusters obtained by fuzzy-k-means by their cluster means. To compute the cluster centres, it is sufficient to keep track of the sum of data points assigned to a cluster and the number, i.e. it is sufficient to store $(\mathrm{Sum}^{(A)}, n^{(A)})$ to represent cluster A, where $\mathrm{Sum}^{(A)}$ is the sum of points assigned to the cluster, and $n^{(A)}$ its number. Note that the merging of two clusters A and B is represented by $(\mathrm{Sum}^{(A)}+\mathrm{Sum}^{(B)}, n^{(A)}+n^{(B)})$ which can easily be computed iteratively.

Thus, patch clustering proceeds like follows:

choose the number of clusters k;
init $(\mathrm{Sum}^{(A_i)} = \vec{0}, n^{(A_i)} = 0)$ for all clusters A_i
repeat until all data are processed
  read the next P data points $X = \{\vec{x}_1, \ldots, \vec{x}_p\}$
  cluster on X combined with $\mathrm{Sum}^{(A_i)}/n^{(A_i)}$
    (multiplicity $n^{(A_i)}$) with batch clustering,
  update the statistics $(\mathrm{Sum}^{(A)}, n^{(A)})$ by the
    cluster centres (including multiplicities)

Obviously, if a data point is assigned to a cluster within a patch, it will remain in the statistics of this cluster. Thus, the order of the processing plays a role and the quantisation error is likely a bit larger for patch clustering compared to batch clustering. However, this effect is not very pronounced depending on the size of the patches as we will see in experiments.

The complexity of patch clustering is reduced compared to batch clustering depending on the size of the patches: for patch size P, one epoch takes time $\sim Pk$ (neglecting sorting, as beforehand), and it needs about $\sqrt{P}$ steps until convergence. Thus, the overall effort is of order $N/P \cdot kP\sqrt{P} = kN\sqrt{P}$ as opposed to $kN\sqrt{N}$ for batch clustering.

4 Experiments

We test patch and batch versions of k-means, fuzzy k-means, and NG for three different data sets, a very simple four-mode clustering problem, a highly-multimodal benchmark dataset from [3], and a large data set from the 1998 KDD cup data mining contest which was also tested in [2].

Four clouds

Data stem from a mixture of four Gaussians with unit variance in two dimensions as depicted in Fig. 1. The set consists of 40000 data points. Training is done by ten-fold crossvalidation using four clusters and a random order of the points. Thereby, the results on the training and test sets differ only slightly. The main effect of a crossvalidation consists in an easy evaluation of the robustness and sensitivity of the algorithm with respect to the data ordering. The patch size is 100, and each run over a patch includes 20 epochs, thereby annealing the neighbourhood of NG from $k/2 = 2$ to 0. The fuzzifier of fuzzy-k-means is chosen as $d = 2$. The mean value of the quantisation error is given in Tab. 1. Obviously, the results are almost identical for all runs due to the simplicity of the data set. Very slight differences of the quantisation error are due to the fact that the points at the (overlapping) cluster borders are assigned differently in patch runs. For all methods, the four cluster centres have been found in every run, showing no differences between batch- and patch-clustering and different patch sizes, respectively, in the principled location of the cluster centres.

Checkerboard

Data constitute a multimodal distribution with 100 clusters in two dimensions as depicted in Fig. 2 separated into a training and test set. The overall number of data points in both, training and test set is about 2000. Training is done using 100 neurons. The patch size is chosen as 200, 400, and 600, respectively. The initial neighbourhood size of NG, 10, is annealed to 0 during training. The fuzzifier of fuzzy-k-means is chosen as $d = 2$. The number of epochs is 20 for each run. It is easily possible to evaluate the number of missed clusters in this task by referring to the classification error of the underlying checkerboard: we assign the label 0 and 1 to the data points such that a checkerboard-pattern with 100 fields arises. We label the

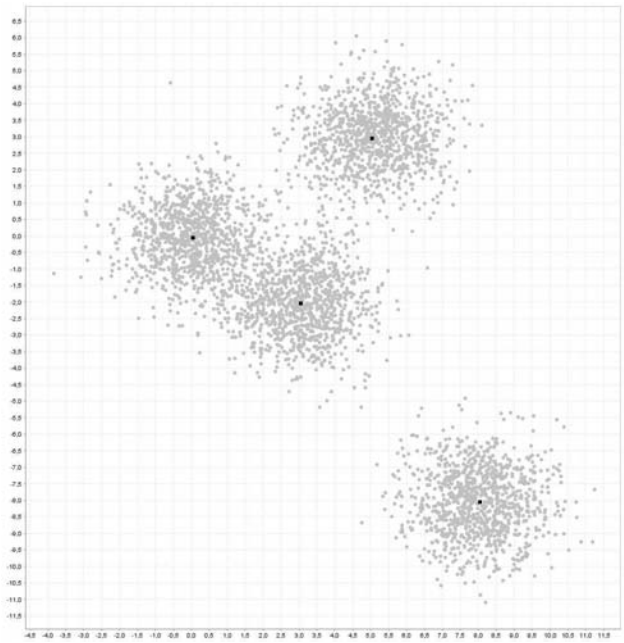

Figure 1: Mixture of four Gaussian clusters

	Batch KM	Patch KM	Batch NG	Patch NG	FKM	Patch FKM
Mean quantisation error (Four Clouds)						
	1.25	1.28	1.25	1.26	1.25	1.28
Variance $\cdot 10^3$						
	0.07	0.37	0.07	0.1	0.08	0.37
Mean classification accuracy in % (Checkerboard)						
patch size 200						
	87.32	86.38	93.35	90.18	94.14	90.80
patch size 400						
		85.43		90.76		90.61
patch size 600						
		84.47		91.80		90.49
Mean quantisation error (KDD)						
patch size 1000						
	0.468	0.464	0.460	0.462	0.511	0.540
variance $\cdot 10^3$						
	0.265	0.265	0.03	0.012		
patch size 10000						
		0.468		0.461		0.520
variance $\cdot 10^3$						
		0.114		0.011		

Table 1: Quantization error or classification accuracy, respectively, as obtained by the different clustering algorithms for the Four Clouds data set, the Checkerboard data set, and the KDD data set for batch and patch clustering using different patch sizes

cluster centres based on the training set and evaluate the classification error of this clustering on the test set. Each percentage of misclassification corresponds to one missed cluster (of 100 total clusters). The mean correct classification (in percentage) obtained in ten runs is reported in Tab. 1. In this case, the quantisation error does not allow to infer the quality of the clustering due to the large number of comparably small clusters: it is around 0.055 for batch and patch variants of k-means and NG and around 0.064 for all fuzzy variants, hardly showing any difference between batch and patch clustering. Fuzzy clustering has a larger quantisation error due to the fact that the cost function is always different from the quantisation error whereas both, NG (for vanishing neighbourhood) and k-means directly optimise this objective.

The clustering results clearly show the following: the task is quite hard and all methods miss a few cluster centres (ranging from 15 for k-means to 7 for NG). Note that each of the 100 clusters is only represented, on average, by 20 data points, and the number of neurons is chosen exactly as 100, i.e. every neuron must represent exactly one centre for optimum classification accuracy. In these cases, a clear difference of patch and batch clustering can be observed: overall, patch clustering finds about 3-4 clusters less compared to batch clustering. This effect depends slightly on the size of the patches, as can be observed in particular for NG. However, for reasonable patch size the loss in accuracy is only minor and it could easily be accounted for by using a slightly larger number of cluster centres than necessary. In this scenario the dependency of k-means on initialisation pops out for both, batch and patch clustering.

Due to the intuitive evaluation of the clustering result by means of the classification error, a comparison to online-neural gas, which can directly be applied to large data sets since it adapts the prototypes directly after every pattern, is easily possible in this scenario: After only one pass through the data, no convergence can be observed in the sense that the neurons are not located in the cluster centres at all. After about 5 epochs, convergence can be observed. On average, 15 clusters are missed after 5 epochs, about 10 clusters are missed after 10 epochs, about 8 clusters are missed after 20 epochs (this setting is comparable to the setting tested for batch NG, whereby batch NG obtains, on average, slightly better performance), about 5 clusters after 50 epochs, and about 3 after 100 epochs. Thus, several passes over the entire data set are necessary for online NG to show competitive results to patch NG.

KDD cup data mining contest

This data set stems from the 1998 KDD cup data mining contest, and we use the same setting as proposed in [2]. Data contains 95412 records with 481 statistical fields which describe statistical information about people who made charitable donations in response to direct mailing requests. For our experiments, 56 features from these fields have been selected, including numerical features such as donation amount, income, age; date values, such as donation date, date of birth; and binary values such as income category. Data are preprocessed such that only numerical values with zero mean and unit variance result. The number

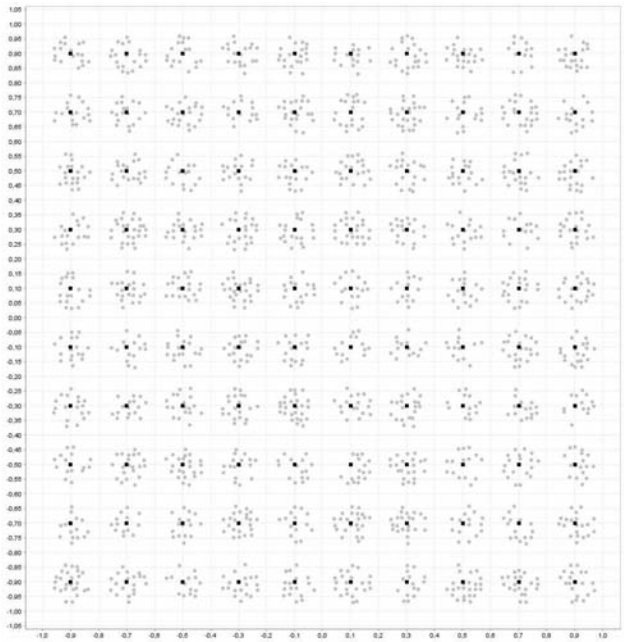

Figure 2: Checkerboard data

	Batch	Patch	Ratio
Checkerboard			
	k-means		
Accuracy	0.8707	0.8675	1.0037
Time (ms)	1053.4	452.5	2.3280
	NG		
Accuracy	0.9353	0.9074	1.0307
Time (ms)	102781.9	37074	2.7723
	fuzzy-k-means		
Accuracy	0.9046	0.8884	1.0182
Time (ms)	182065.5	82409.5	2.2093
11 Clouds			
	k-means		
Accuracy	0.9928	0.0.995	1
Time (ms)	76376.5	64465.6	1.18
	NG		
Accuracy	0.9984	0.9984	1
Time (ms)	94703.9	79134.5	1.2

Table 2: Classification results and time difference of batch and patch clustering with 20 (for batch) and 5 (for patch) epochs and patch size 200 for the Checkerboard data, and for batch and patch clustering with 8 (for batch) and 5 (for patch) epochs for the simpler 11 clouds data set.

of clusters was set to 10, as proposed in [2], and the number of epochs is 20. The mean quantisation error averaged in a ten-fold crossvalidation is reported in Tab. 1 for two different patch sizes corresponding to roughly 1% and 10% of the data, respectively. Since the preprocessing in [2] has been described only qualitatively, we cannot compare to the results reported in [2], but we test k-means in our setting. Obviously, a slight improvement of NG compared to k-means can be observed, whereby patch clustering using 10% of the data only slightly reduces the achieved results. Interestingly, the variance of the results can be reduced by patch clustering compared to batch clustering, and – as expected – NG compared to k-means. As beforehand, the results obtained by fuzzy clustering are worse because the objective of fuzzy clustering is different from the quantisation error. However, it is clearly demonstrated that for all variants, patch optimisation only slightly decreases the overall result whereby reducing the required buffer size to a fixed size and reducing the training to a single run over the overall data set (combined with a small number of epochs for each patch).

Comparison of the clustering time

Due to the smaller size of the data sets, patch clustering requires less iterations until convergence compared to batch clustering. This effect can be measured in experiments as follows:

For the Checkerboard data, we perform the same experiment as beforehand, thereby using 20 iterations for batch clustering and 5 iterations for patch clustering for a patch size 200. These numbers represent the necessary number of iterations until convergence for the respective scenario. As can be seen from Tab. 2, the quotient of the performance measured by means of the classification error is close to one, whereby the gain of the efficiency accounts for a factor larger than 2.

For a simpler data set consisting only of 11 clouds and 44000 data points, the effect is a bit less pronounced: for batch and patch clustering, the classification accuracy is the same while obtaining an efficiency gain of about 1.2 for patch compared to batch clustering. This gain is due to the reduced number of necessarz epochs until convergence, which are 5 for patch clustering and 8 for batch clustering, see Tab. 2.

5 Conclusions

We have presented an efficient patch optimisation scheme for NG and fuzzy-k-means in the presence of large data sets which requires only a priorly fixed amount of buffer space for optimisation and which reduces the training time by a factor roughly $\sqrt{N}$ due to the smaller number of epochs for batch optimisation of patches. The effectivity of the method has been demonstrated in a variety of experiments.

Note that it is possible to extend the method directly to SOM and HSOM [12, 16] by using batch optimisation of these models assuming the formulation of a winner as proposed by Heskes [8] such that an underlying cost function is available. The clear foundation of the optimisation on a

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

cost function allows an immediate transfer of patch clustering to extended versions which take supervised label information into account such as proposed in [6], or which extend the applicability to general proximity data e.g. by means of median variants as proposed in [3, 13].

Further, patch clustering opens possibilities to speed up clustering by parallelisation: obviously, optimisation of different patches can be done independently on different CPUs whereby merging takes place by means of an integration of the sufficient statistics of clusterings into a new run. The decomposition in patches and recombination of results can be arranged e.g. in a tree-like manner such that a simple and efficient parallel implementation of NG and fuzzy-k-means results. Similar ideas, an iterative clustering and merging of results, have been proposed in the article [15] for k-means clustering. The efficiency of the method for NG will be the subject of future research.

References

[1] P.S.Bradley, U.Fayyad, C.Reina (1998), Scaling clustering algorithms to large data sets, in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 9-15, AAAI Press.

[2] F.Farnstrom, J.Lewis, C.Elkan (2000), Scalability for clustering algorithms revisited, *SIGKDD Explorations* 2(1):51-57.

[3] M. Cottrell, B. Hammer, A. Hasenfuss, and T. Villmann (2006), Batch and median neural gas, *Neural Networks*, 19:762-771.

[4] S. Guha, N. Mishra, R. Motwani, L. O'Callaghan (2000). Clustering Data Streams. In *IEEE Symposium on Foundations of Computer Science*, 359-366.

[5] S. Guha, R. Rastogi, K. Shim (1998). CURE: an efficient clustering algorithm for large datasets. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 73-84.

[6] B. Hammer, A. Hasenfuss, F.-M. Schleif, and T. Villmann (2006), Supervised batch neural gas, In *Proceedings of Conference Artificial Neural Networks in Pattern Recognition (ANNPR)*, F. Schwenker (ed.), Springer, pages 33-45.

[7] J. Hartigan (1975). *Clustering Algorithms*. Wiley.

[8] T. Heskes (2001). Self-organizing maps, vector quantization, and mixture modeling. *IEEE Transactions on Neural Networks* 12:1299-1305.

[9] F. Höppner, F. Klawonn, R. Kruse, T. Runkler (1999). *Fuzzy Cluster Analysis*, Wiley.

[10] R. Jin, A. Goswami, G. Agrawal (to appear). Fast and Exact Out-of-Core and Distributed K-Means Clustering, *Knowledge and Information System*.

[11] I.A. Klampanos, J.M. Jose, C.J.'Keith' van Rijsbergen (2006). Single-pass clustering for peer-to-peer information retrieval: the effect of document ordering. In *ACM International Conference Proceeding Series*, 152. Proceedings of the 1st international conference on Scalable information systems, Hong Kong, Article No. 36.

[12] T. Kohonen (1982), Self-Organized formation of topologically correct feature maps, *Biological Cybernetics*, 43:59-69.

[13] T. Kohonen and P. Somervuo (2002), How to make large self-organizing maps for nonvectorial data, *Neural Networks* 15:945-952.

[14] T. Martinetz, S.G. Berkovich, and K.J. Schulten (1993). 'Neural-gas' network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks* 4:558-569.

[15] S. Nittel, K.T. Leung (2004). Parallelizing clustering of geoscientific data sets using data streams. In: *Scientific and Statistical Database Management*, 73-84.

[16] H. Ritter (1999), Self-organizing Maps in non-euclidean Spaces, *Kohonen Maps*, 97-108, Eds.: E. Oja and S. Kaski.

[17] W. Wang, J. Yang, R.R. Muntz (1997). STING: a statistical information grid approach to spatial data mining. In *Proceedings of the 23rd VLDB Conference*, 186-195.

[18] T. Zhang, R. Ramakrishnan, M. Livny (1996). BIRCH: an efficient data clustering method for very large databases. In *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Databas Systems*, 103-114.