

BATCH NEURAL GAS

Marie Cottrell¹, Barbara Hammer², Alexander Hasenfuß², Thomas Villmann³

¹Université Paris I, SAMOS-MATISSE, Paris, France, cottrell@univ-paris1.fr

²Clausthal University of Technology, Computer Science Institute, Clausthal-Zellerfeld, Germany, hammer/hasenfuss@in.tu-clausthal.de

³University of Leipzig, Clinic of Psychotherapy, Leipzig, Germany, villmann@informatik.uni-leipzig.de

Abstract - *We introduce a batch variant of the neural gas (NG) clustering algorithm which optimizes the same cost function as NG but shows faster convergence. It has the additional benefit that, based on the notion of the generalized median, a variant for non-vectorial proximity data can be introduced in analogy to the median self-organizing map (SOM). We prove convergence of batch and median NG and demonstrate its behaviour in experiments.*

Key words - Neural gas, batch algorithm, proximity data, cost function

1 Introduction

Clustering constitutes a fundamental problem in various fields such as pattern recognition, image processing, or machine learning [17]. Often, the goal is to represent data by a small number of representative prototypes. Popular algorithms related to neural networks include k-means, the self-organizing map (SOM), neural gas (NG), and alternatives [8, 19]. K-means clustering directly aims at a minimization of the quantization error [4]. However, its update scheme is local such that it easily gets stuck in local optima. Neighborhood cooperation as for SOM and NG offers one biologically plausible solution. Apart from a reduction of the influence of initialization, additional semantical insight is gained: browsing within the map and, if a prior low dimensional lattice is chosen, data visualization. NG optimizes a cost function which, as a limit case, yields the quantization error [15]. SOM, on the contrary, possesses a cost function in the continuous case only for a variation of the original learning rule [6, 12]. In addition, a prior lattice might be suboptimal for a given task [22].

There exist mainly two different optimization schemes for these objectives: online variants, which adapt the prototypes after each pattern, and batch variants which adapt the prototypes according to all patterns at once. Batch approaches are much faster, since only one adaptation is necessary in each cycle and convergence can usually be observed after few steps. However, topological ordering of SOM might be destroyed as shown in [9] such that a good initialization is necessary. Batch variants are often taken for SOM or k-means if data are available a priori. For NG, we derive a batch version and prove convergence in this contribution.

In a variety of tasks such as classification of protein structures, documents, surveys, or biological signals, an explicit metric structure such as the euclidean metric is not available, rather only the proximity of data points is given [7, 10, 20]. In such cases, a clustering method which does not rely on a vector space has to be applied such as spectral clustering [1]. Several alternatives to SOM have been proposed which can deal with more general, mostly discrete data [7, 10, 20]. The article [14] proposes a particularly simple possibility for proximity data: the mean value of the batch SOM is substituted by the generalized median. Naturally, the same idea can be transferred to batch NG and k-means as we will demonstrate in this contribution.

2 Neural gas

Assume data points $\vec{x} \in \mathbb{R}^m$ are distributed according to P , the goal of NG as introduced in [15] is to find prototype locations $\vec{w}^i \in \mathbb{R}^m$, $i = 1, \dots, n$, such that these prototypes represent the distribution P as accurately as possible, minimizing the cost function

$$E_{\text{NG}}(\vec{w}) = \frac{1}{2C(\lambda)} \sum_{i=1}^n \int h_{\lambda}(k_i(\vec{x}, \vec{w})) \cdot d(\vec{x}, \vec{w}^i) P(d\vec{x})$$

where $d(\vec{x}, \vec{y}) = (\vec{x} - \vec{y})^2$ denotes the squared euclidean distance, $k_i(\vec{x}, \vec{w}^i) = |\{\vec{w}^j \mid d(\vec{x}, \vec{w}^j)^2 < d(\vec{x}, \vec{w}^i)\}|$ is the rank of the prototypes sorted according to the distances, $h_{\lambda}(t) = \exp(-t/\lambda)$ is a Gaussian shaped curve with neighborhood range $\lambda > 0$, and $C(\lambda)$ is the constant $\sum_{i=1}^n h_{\lambda}(k_i)$. The learning rule consists of a stochastic gradient descent, yielding $\Delta \vec{w}^i = \epsilon \cdot h_{\lambda}(k_i(\vec{x}^j, \vec{w})) \cdot (\vec{x}^j - \vec{w}^i)$ for all \vec{w}^i given \vec{x}^j . Thereby, the neighborhood range λ is decreased during training to ensure independence of initialization and optimization of the quantization error. As pointed out in [16], the result can be associated with a data optimum lattice.

NG is a simple and highly effective algorithm for data clustering. Popular alternative clustering algorithms are offered by the SOM as introduced by Kohonen [13] and k-means clustering [8]. SOM uses the adaptation strength $h_{\lambda}(nd(I(\vec{x}^j), i))$ instead of $h_{\lambda}(k_i(\vec{x}^j, \vec{w}))$, $I(\vec{x}^j)$ denoting the index of the closest prototype, the winner, for \vec{x}^j , and nd a priori chosen, often two-dimensional neighborhood structure of the neurons. A low-dimensional lattice offers the possibility to visualize data. However, if the primary goal is clustering, a fixed topology puts restrictions on the map and topology preservation often cannot be achieved [22]. SOM does not possess a cost function in the continuous case and its mathematical investigation is difficult [6]. However, if the winner is chosen as the neuron i with minimum averaged distance $\sum_{l=1}^n h_{\lambda}(nd(i, l))d(\vec{x}^j, \vec{w}^l)$, it optimizes the cost term

$$E_{\text{SOM}}(\vec{w}) \sim \sum_{i=1}^n \int \chi_{I^*(\vec{x})}(i) \cdot \sum_{l=1}^n h_{\lambda}(nd(i, l)) \cdot d(\vec{x}, \vec{w}^l) P(d\vec{x})$$

as pointed out by Heskes [12]. Here, $I^*(\vec{x})$ denotes the winner index according to the averaged distance and $\chi_j(i)$ is the characteristic function of j . K-means clustering adapts only the winner in each step, thus it optimizes the standard quantization error

$$E_{\text{kmeans}}(\vec{w}) \sim \sum_{i=1}^n \int \chi_{I(\vec{x})}(i) \cdot d(\vec{x}, \vec{w}^i) P(d\vec{x})$$

where $I(\vec{x})$ denotes the winner index for \vec{x} in the classical sense. Unlike SOM and NG, k-means is very sensitive to initialization of the prototypes.

2.1 Batch clustering

If training data $\vec{x}^1, \dots, \vec{x}^p$ are given priorly, fast alternative batch training schemes exist for k-means and SOM. Starting from random positions of the prototypes, batch learning iteratively perform the following two steps

- (1) determine the winner $I(\vec{x}^i)$ resp. $I^*(\vec{x}^i)$ for each data point \vec{x}^i ,
- (2) determine new prototypes as $\vec{w}^i = \sum_{j \mid I(\vec{x}^j)=i} \vec{x}^j / |\{j \mid I(\vec{x}^j) = i\}|$ for k-means and $\vec{w}^i = \sum_{j=1}^p h_{\lambda}(nd(I^*(\vec{x}^j), i)) \cdot \vec{x}^j / \sum_{j=1}^p h_{\lambda}(nd(I^*(\vec{x}^j), i))$ for SOM.

It has been shown in [4, 5] that batch k-means resp. batch SOM optimize the same cost functions as their online variants, whereby the generalized winner notation is used for SOM. In addition, as pointed out in [12], this formulation allows to link the models to statistical formulations and it can be interpreted as a limit case of EM optimization schemes for appropriate mixture models. Often, batch training converges after only few (10-100) cycles such that this training mode offers considerable speedup in comparison to the online variants. Here, we introduce a batch variant of NG. As for SOM and k-means, it can be derived from the cost function of NG, which, for discrete data $\vec{x}^1, \dots, \vec{x}^p$, reads as

$$E_{\text{NG}}(\vec{w}) \sim \sum_{i=1}^n \sum_{j=1}^p h_{\lambda}(k_i(\vec{x}^j, \vec{w})) \cdot d(\vec{x}^j, \vec{w}^i).$$

For the batch algorithm, the quantities $k_{ij} := k_i(\vec{x}^j, \vec{w})$ are treated as hidden variables with the constraint that the values k_{ij} ($i = 1, \dots, n$) constitute a permutation of $\{0, \dots, n-1\}$ for each point \vec{x}^j . E_{NG} is interpreted as a function depending on \vec{w} and k_{ij} which is optimized in turn with respect to the hidden variables k_{ij} and with respect to the prototypes \vec{w}^i , yielding the two adaptation steps of *batch NG*:

1. determine $k_{ij} = k_i(\vec{x}^j, \vec{w}) = |\{\vec{w}^l \mid d(\vec{x}^j, \vec{w}^l) < d(\vec{x}^j, \vec{w}^i)\}|$ as the rank of prototype \vec{w}^i ,
2. based on the hidden variables k_{ij} , set

$$\vec{w}^i = \frac{\sum_{j=1}^p h_{\lambda}(k_{ij}) \cdot \vec{x}^j}{\sum_{j=1}^p h_{\lambda}(k_{ij})}.$$

Before proceeding to a generalization to proximity data, we unify the notation for the batch versions of NG, SOM, and batch NG. In the discrete setting, these three models optimize a cost function of the form

$$E := \sum_{i=1}^n \sum_{j=1}^p f_1(k_{ij}(\vec{w})) \cdot f_2^{ij}(\vec{w})$$

where $f_1(k_{ij}(\vec{w}))$ is the characteristic function of the winner, $\chi_{I(\vec{x}^j)}(i)$ resp. $\chi_{I^*(\vec{x}^j)}(i)$, for k-means and SOM, and it is $h_{\lambda}(k_i(\vec{x}^j, \vec{w}))$ for neural gas. $f_2^{ij}(\vec{w})$ equals the distance $d(\vec{x}^i, \vec{w}^j)$ for k-means and NG, and it is the averaged distance $\sum_{l=1}^n h_{\lambda}(nd(i, l)) \cdot d(\vec{x}^j, \vec{w}^l)$ for SOM. The batch algorithms optimize E with respect to k_{ij} in step **(1)** assuming fixed \vec{w} . Thereby, for each j , the vector k_{ij} ($i = 1, \dots, n$) is restricted to a vector with exactly one entry 1 and 0, otherwise, for k-means and SOM and it is restricted to a permutation of $\{0, \dots, n-1\}$ for NG. Thus, the elements k_{ij} come from a discrete set which we denote by K . In step **(2)**, E is optimized with respect to \vec{w}^j assuming fixed k_{ij} . The update formulas as introduced above can be derived by taking the derivative of f_2^{ij} with respect to \vec{w} .

2.2 Median clustering

For proximity data $\vec{x}^1, \dots, \vec{x}^n$, only the distance matrix $d_{ij} := d(\vec{x}^i, \vec{x}^j)$ is available but data are not embedded in a vector space. A solution proposed by Kohonen for batch-SOM is based on the notion of the generalized median [14]: prototypes are chosen from the discrete set given by the training points $X = \{\vec{x}^1, \dots, \vec{x}^p\}$ in an optimum way. Thus, E is optimized within X^n instead of $(\mathbb{R}^m)^n$, choosing \vec{w}^i in step **(2)** as $\vec{w}^i = \vec{x}^l$ where $l = \operatorname{argmin}_l \sum_{j=1}^p h_{\lambda}(nd(I^*(\vec{x}^j), i)) \cdot d(\vec{x}^j, \vec{x}^l)$. In [14], Kohonen considers only the data

points mapped to a neighborhood of neuron i as potential candidates for \vec{w}^i and, in addition, reduces the above sum to points mapped into a neighborhood of i . For small neighborhood range and approximately ordered maps, this does not change the result.

The same principle can be applied to k-means and batch NG. In step **(2)**, instead of taking the vectors in $(\mathbb{R}^m)^n$ which minimize E , prototype i is chosen in X as

$$\vec{w}^i = \vec{x}^l \quad \text{where} \quad l = \operatorname{argmin}_{l'} \sum_{j=1}^p \chi_{I(\vec{x}^j)}(l) \cdot d(\vec{x}^j, \vec{x}^{l'})$$

assuming fixed $\chi_{I(\vec{x}^j)}(l)$ for k-means and

$$\vec{w}^i = \vec{x}^l \quad \text{where} \quad l = \operatorname{argmin}_{l'} \sum_{j=1}^p h_\lambda(k_{ij}) \cdot d(\vec{x}^j, \vec{x}^{l'})$$

assuming fixed $k_{ij} = k_i(\vec{x}^j, \vec{w})$. For roughly ordered maps, a restriction of potential candidates \vec{x}^l to data points mapped to a neighborhood of i can speed up training as for median SOM.

2.3 Convergence

These algorithms optimize $E = E(\vec{w})$ by consecutive optimization of the hidden variables $k_{ij}(\vec{w})$ and \vec{w} . We can assume that, for given \vec{w} , the values k_{ij} determined by the above algorithms are unique, introducing some order in case of ties. Note that the values k_{ij} come from a discrete set K . If k_{ij} are fixed, the choice of the optimum \vec{w} is unique in the algorithms for the continuous case, as is obvious from the formulas given above, and we can assume uniqueness for the median variants by introducing an order. Consider the function

$$Q(\vec{w}', \vec{w}) = \sum_{i=1}^n \sum_{j=1}^p f_1(k_{ij}(\vec{w})) \cdot f_2^{ij}(\vec{w}').$$

Note that $E(\vec{w}) = Q(\vec{w}, \vec{w})$. Assume prototypes \vec{w} are given, and new prototypes \vec{w}' are computed based on $k_{ij}(\vec{w})$ using one of the above batch or median algorithms. It holds $E(\vec{w}') = Q(\vec{w}', \vec{w}') \leq Q(\vec{w}', \vec{w})$ because $k_{ij}(\vec{w}')$ are optimum assignments for k_{ij} in E , given \vec{w}' . In addition, $Q(\vec{w}', \vec{w}) \leq Q(\vec{w}, \vec{w}) = E(\vec{w})$ because \vec{w}' are optimum assignments of the prototypes given $k_{ij}(\vec{w})$. Thus, $E(\vec{w}') - E(\vec{w}) = E(\vec{w}') - Q(\vec{w}', \vec{w}) + Q(\vec{w}', \vec{w}) - E(\vec{w}) \leq 0$, i.e., in each step of the algorithms, E is decreased. Since there exists only a finite number of different values k_{ij} and the assignments are unique, the algorithms converge in a finite number of steps toward a fixed point \vec{w}^* for which $(\vec{w}^*)' = \vec{w}^*$ holds.

Consider the case of continuous \vec{w} . Since k_{ij} are discrete, $k_{ij}(\vec{w})$ is constant in a vicinity of \vec{w}^* if no data points lie at the borders of two receptive fields. Then $E(\cdot)$ and $Q(\cdot, \vec{w}^*)$ are identical in a neighborhood of \vec{w}^* and thus, a local optimum of Q is also a local optimum of E . Therefore, in the continuous case, if no data points are directly located at the borders of receptive fields, the batch algorithms converge to a local optimum of E .

3 Experiments

All algorithms have been implemented based on the SOM Toolbox for Matlab [18]. We used k-means, SOM, batch-SOM, and NG with default parameters. Batch NG and median versions of NG, SOM, and k-means have been implemented using the above formulas. Since

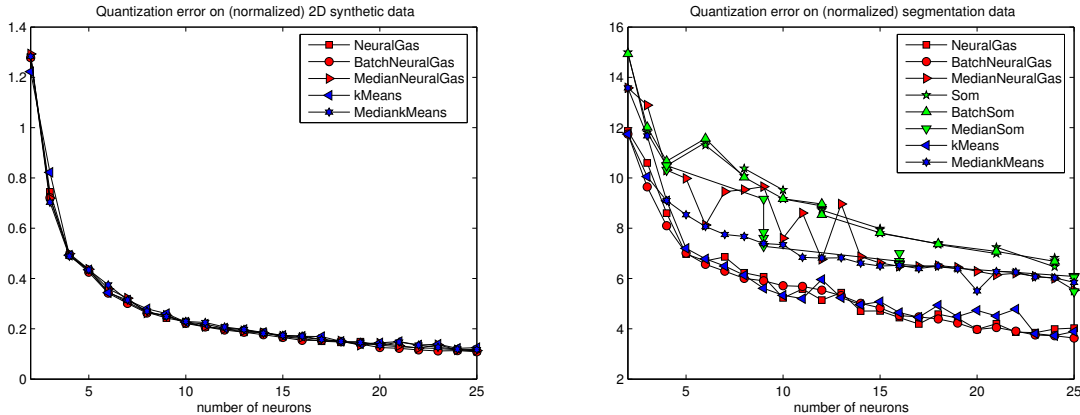


Figure 1: Mean quantization error of the methods for the synthetic data set (left) and the segmentation data set (right).

prototypes can easily become identical due to a limited number of different positions for the median versions, a small amount of noise has been added in each epoch. Vectorial training sets are normalized prior to training using z-transformation. Initialization of prototypes takes place using small random values. The initial neighborhood rate for neural gas is $\lambda = n/2$, n being the number of neurons, and it is multiplicatively decreased during training. For median SOM, we restrict to square lattices of $n = \sqrt{n} \times \sqrt{n}$ neurons and a rectangular neighborhood structure. Here the initial neighborhood rate is $\sqrt{n}/2$.

3.1 Synthetic data

The first data set is the two-dimensional synthetic data set from [19] consisting of 250 data points and 1000 training points. Clustering has been done using $n = 2, \dots, 25$ prototypes, resp. the closest number of prototypes implemented by a rectangular lattice for SOM, training for $5n$ epochs. The mean quantization error $\sum_{i=1}^n \sum_{j=1}^p \chi_{I(\bar{x}^j)}(i) \cdot d(\bar{x}^j, \bar{w}^i)/p$ on the test set and the location of prototypes within the training set are depicted in Figs. 1 and 2. Obviously, the location of prototypes coincides for different versions of NG. This observation also holds for different numbers of prototypes, whereby the result is subject to random fluctuations for larger numbers. For k-means, idle prototypes can be observed for large n . For batch-SOM and SOM, the quantization error is worse (ranging from 1.7 for 2 neurons up to 0.3 for 24 neurons), which can be attributed to the fact that the map does not fully unfold upon the data set and edge effects remain. Median SOM (which has been directly implemented in analogy to median NG) yields a quantization error competitive to NG. Thus, batch and median NG allow to achieve results competitive to NG in this case, however, using less effort.

3.2 Segmentation data

The segmentation data set from UCI consists of 210 (training set) resp. 2100 (test set) 19 dimensional data points which are obtained as pixels from outdoor images preprocessed by standard filters such as averaging, saturation, intensity, etc. The problem is interesting since it contains high dimensional and only sparsely covered data. The quantization error obtained for the test set is depicted in Fig. 1. As beforehand, SOM suffers from the restriction of the topology. Neural gas yields very robust behavior, whereas for k-means, idle prototypes can be observed. The median versions yield a larger quantization error compared to the vector-based algorithms. The reason lies in the fact that a high dimensional data set with only few

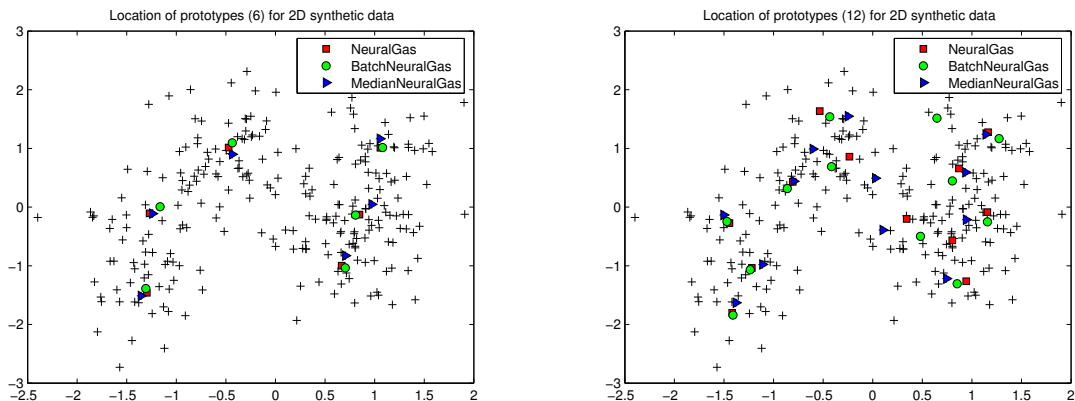


Figure 2: Location of the prototypes for the synthetic data set for different variants of NG.

training patterns is considered, such that the search space for median algorithms is small in these cases and random effects and restrictions account for the increased error.

3.3 Checkerboard

This data set is taken from [11]. Two-dimensional data are arranged on a checkerboard, resulting in 10 times 10 clusters, each consisting of 15 to 20 points. For each algorithm, we train 5 times 100 epochs for 100 prototypes. Obviously, the problem is highly multimodal and, usually the algorithms do not find all clusters. The number of missed clusters can easily be judged in the following way: the clusters are labeled consecutively using labels 1 and 2. prototypes are labeled a posteriori based on a majority vote on the training set. The number of errors which arise from this classification on an independent test set count the number of missed clusters, since 1% error roughly corresponds to one missed cluster.

The results are collected in Tab. 1. The smallest quantization error is obtained by batch NG, the smallest classification error can be found for median SOM. As beforehand, the implementations for SOM and batch SOM do not fully unfold the map among the data. In the same way the online NG does not achieve a small error because of the restricted number of epochs. K-means also shows a quite high error (it misses more than 10 clusters) which can be explained by the existence of multiple local optima. In contrast, batch NG and median NG find all but 3 to 4 clusters. Median SOM even finds all but only 1 or 2 clusters. Surprisingly, also median k-means shows quite good behaviour, unlike k-means itself, which might be due to the fact that the generalized medians enforce the prototypes to settle within the clusters.

	NG	batch NG	median NG	SOM	batch SOM	median SOM	kmeans	median kmeans
quantization error								
train	0.0043	0.0028	0.0043	0.0127	0.0126	0.0040	0.0043	0.0046
test	0.0051	0.0033	0.0048	0.0125	0.0124	0.0043	0.0050	0.0052
classification error								
train	0.1032	0.0330	0.0338	0.2744	0.2770	0.0088	0.1136	0.0464
test	0.1207	0.0426	0.0473	0.2944	0.2926	0.0111	0.1376	0.0606

Table 1: Quantization error and classification error for posterior labeling for training and test set (both are of size about 1800). The mean over 5 runs is reported.

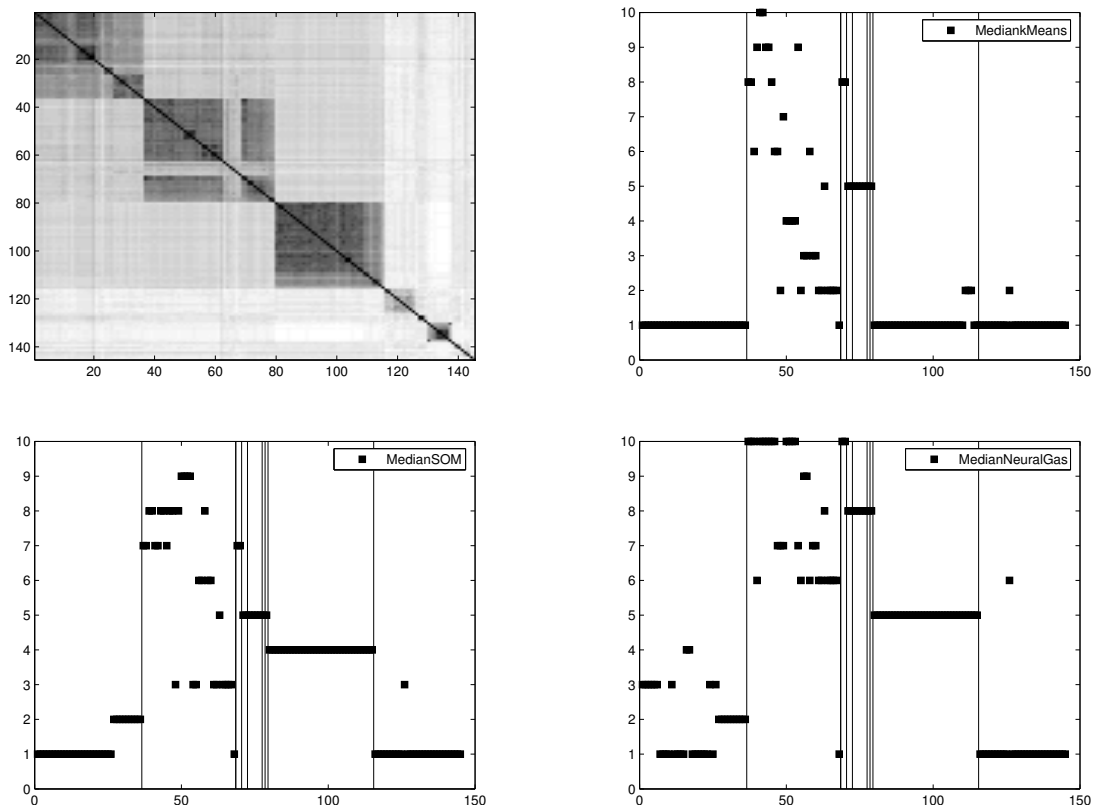


Figure 3: Distance matrix for protein data (upper left). Typical results for median classification and 10 prototypes. The x-axes shows the protein number, the y-axes its winner neuron. The vertical lines indicate an expert classification into different protein families (from left to right: hemoglobin α , β , δ , ϵ , γ , F, myoglobin, others).

3.4 Proximity data

We take the data set from [21] and [20]: the dissimilarity of 145 globin proteins of different families is given in matrix form. The matrix is determined based on sequence alignment using biochemical and structural information, as depicted in Fig. 3.

We train 10 times using 10 prototypes and 500 epochs. The mean quantization errors (and variances) are 3.7151 (0.0032) for median neural gas, 3.7236 (0.0026) for median SOM, and 4.5450 (0.0) for median k-means, thus k-means yields worse results compared to NG and SOM. Typical clustering results are depicted in Fig. 3. A classification provided by experts is indicated by vertical lines in the images. Thereby, the last elements also have a large intercluster distance such that they are grouped together into some (random) cluster for all methods. All methods detect the first cluster (hemoglobin α) and neural gas and SOM also detect the eighth cluster (myoglobin). In addition, SOM and NG group together elements of clusters two to seven in a reasonable sense. Thereby, according to the variance in the clusters, more than one prototype is used for one cluster. Note that the goal of NG and SOM is a minimization of their underlying cost function, such that the border can lie between semantic clusters for these methods. Thus, the results obtained by SOM and NG are reasonable and they detect several semantically meaningful clusters. This formation of clusters is also supported when training with a different number of prototypes (not shown here.)

4 Conclusions

We have proposed a batch variant of NG which allows fast training for a priorly given data set and a transfer to proximity data. We have shown that the method converges and it optimizes the same cost function as NG. This theoretical finding was supported by experiments for vectorial data where the results of batch NG and NG are very similar. Unlike k-means, NG is not sensitive to initialization and, unlike SOM, it automatically determines a data optimum lattice, such that a small quantization error can be achieved and topological initialization is not crucial. Median NG restricts the adaptation to locations within the data set such that it can be applied to non-vectorial data. We compared median NG to its alternatives for vectorial data observing that competitive results arise if enough data are available. We added one experiment including proximity data where we could obtain semantically meaningful grouping. Unlike SOM, NG solely aims at data clustering and not data visualization, such that it can use a data optimum lattice and it is not restricted by topological constraints. If a visualization of the output of NG is desired, a subsequent visualization of the prototype vectors is possible also in the non-vectorial case using e.g. multidimensional scaling [3].

References

- [1] M. Belkin and P. Niyogi (2002), Laplacian eigenmaps and spectral techniques for embedding and clustering, *NIPS*.
- [2] C.L. Blake and C.J. Merz (1998), UCI Repository of machine learning databases, Irvine, CA, University of California, Department of Information and Computer Science.
- [3] I. Borg and P. Groenen (1997), Modern multidimensional scaling: theory and applications, Springer.
- [4] L. Bottou and Y. Bengio (1995), Convergence properties of the k-means algorithm, *NIPS*.
- [5] Y. Cheng (1997), Convergence and ordering of Kohonen's batch map, *Neur. Comp.* **9**:1667-1676.
- [6] M. Cottrell, J.C. Fort, and G. Pagès (1999), Theoretical aspects of the SOM algorithm, *Neurocomputing* **21**:119-138.
- [7] M. Cottrell, S. Ibbou, and P. Letrémy (2004), SOM-based algorithms for qualitative variables, *Neural Networks* **17**:1149-1168.
- [8] R.O. Duda, P.E. Hart, and D.G. Stork (2000), *Pattern Classification*, Wiley.
- [9] J.-C. Fort, P. Letrémy, and M. Cottrell (2002), Advantages and drawbacks of the Batch Kohonen algorithm, in *ESANN'2002*, M. Verleysen (ed.), 223-230, D Facto.
- [10] T. Graepel and K. Obermayer (1999), A self-organizing map for proximity data, *Neur. Comp.* **11**:139-155.
- [11] B. Hammer, M. Strickert, and T. Villmann (2005), Supervised neural gas with general similarity measure, *Neural Processing Letters* **21**:21-44.
- [12] T. Heskes (2001), Self-organizing maps, vector quantization, and mixture modeling, *IEEE Transactions on Neural Networks*, **12**:1299-1305.
- [13] T. Kohonen (1995), *Self-Organizing Maps*, Springer.
- [14] T. Kohonen and P. Somervuo (2002), How to make large self-organizing maps for nonvectorial data, *Neural Networks* **15**:945-952.
- [15] 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.
- [16] T. Martinetz and K.J. Schulten (1994), Topology representing networks, *Neural Networks* **7**:507-522.
- [17] M.N. Murty, A.K. Jain, and P.J. Flynn (1999), Data clustering: a review, *ACM Computing Surveys* **31**:264-323.
- [18] Neural Networks Research Centre, Helsinki University of Technology, SOM Toolbox, <http://www.cis.hut.fi/projects/somtoolbox/>
- [19] B.D. Ripley (1996), *Pattern Recognition and Neural Networks*, Cambridge.
- [20] S. Seo and K. Obermayer (2004), Self-organizing maps and clustering methods for matrix data, *Neural Networks* **17**:1211-1230.
- [21] H. Mevissen and M. Vingron (1996), Quantifying the local reliability of a sequence alignment, *Protein Engineering* **9**:127-132.
- [22] T. Villmann, R. Der, M. Herrmann, and T. Martinetz (1994), Topology preservation in self-organizing feature maps: exact definition and measurement, *IEEE Transactions on Neural Networks* **2**:256-266.