Supervised Batch Neural Gas

Barbara Hammer[1], Alexander Hasenfuss[1],
Frank-Michael Schleif[2], and Thomas Villmann[3]

[1] Clausthal University of Technology, Institute of Computer Science,
Clausthal-Zellerfeld, Germany
[2] University of Leipzig, Institute of Computer Science, Germany
[3] University of Leipzig, Clinic for Psychotherapy, Leipzig, Germany

Abstract. Recently, two extensions of neural gas have been proposed: a fast batch version of neural gas for data given in advance, and extensions of neural gas to learn a (possibly fuzzy) supervised classification. Here we propose a batch version for supervised neural gas training which allows to efficiently learn a prototype-based classification, provided training data are given beforehand. The method relies on a simpler cost function than online supervised neural gas and leads to simpler update formulas. We prove convergence of the algorithm in a general framework, which also incorporates supervised k-means and supervised batch-SOM, and which opens the way towards metric adaptation as well as application to proximity data not embedded in a real-vector space.

1 Introduction

Prototype-based classification constitutes an intuitive machine learning technique which represents classes by typical prototype locations and assigns labels to new data points by means of a winner-takes-all rule. Unlike feedforward networks or support vector machines (SVM), the method provides insight into the classification behavior by an inspection of the prototypes. Interestingly, the generalization behavior of prototype-based techniques is quite robust, since generalization bounds which only depend on the hypothesis margin but not on the number of parameters of the model (in particular the input dimensionality) can be derived similar to SVMs [4].

One of the most popular learning techniques for prototype-based methods is Kohonen's learning vector quantization (LVQ) and variants and extensions thereof [7,10]. Thereby, LVQ is based on heuristics and applies Hebbian learning to the respective winning prototype. Several extensions of the basic algorithm substitute this heuristic by adaptation rules which are derived from a cost function [7]. However, these methods rely on local adaptations. Therefore they easily get stuck in local optima if dealing with multimodal data. Modifications using neighborhood cooperation avoid this problem, such as the proposal presented in [6], which integrates the dynamics of unsupervised neural gas (NG) [12] into the adaptation.

Another problem of LVQ type classifiers is given by the fact that class labels are necessarily crisp in these learning algorithms. Fuzzy-labeled data cannot be

F. Schwenker and S. Marinai (Eds.): ANNPR 2006, LNAI 4087, pp. 33–45, 2006.

learned and it is not possible to indicate ambiguous regions of the data space by a fuzzy labeling of the prototypes. Recently, an extension of unsupervised neural gas clustering to a supervised fuzzy classifier, namely fuzzy labeled neural gas (FLNG), has been proposed [16]. Obviously, NG can trivially be expanded to a classifier by posterior labeling of the prototypes. However, in this case the prototype locations are not adapted according to the labels, but according to the data statistics only. The basic idea of FLNG consists in an extension of the NG cost function by a term measuring the deviation of the class labels from the data points and prototypes, whereby the latter are automatically adapted during training. Adaptation takes place by means of a stochastic gradient descent on the resulting cost function, whereas the class labels influence the prototype locations.

This general idea faces the problem that the extension of the NG cost function by the quantization error of the labels as proposed in [16] for discrete data is not differentiable at the borders of the receptive fields. Therefore, modifications are necessary. For continuous data the article [16] proposes approximations which are differentiable and which lead to quite complicated update terms for the prototypes and class labels. Here, we consider a different extension of the cost function and a different optimization scheme, batch learning, which allows a direct optimization of the quantization error and which yields very simple and intuitive update rules. In addition, it shares the fast convergence of (unsupervised) batch NG as introduced in [3]. We test the method on several data sets.

Similar to supervised batch NG, it is possible to extend other popular batch clustering algorithms such as k-means and the batch self-organizing map (SOM) [10] to supervised classification. We integrate these approaches into a common framework by means of the cost function, and we show convergence of general batch optimization. Thereby, the possibility to adapt metric parameters as has been used in recent LVQ versions [6] is also included. In addition, the application to general proximity data which is not embedded in a euclidian vector space becomes possible by median versions of the optimization scheme as introduced in [11]. We shortly discuss these possibilities covered by the general framework.

2 Unsupervised Clustering

Assume data vectors $\mathbf{v} \in \mathbb{R}^d$ are given as stimuli, distributed according to an underlying probability distribution $P(\mathbf{v})$. The aim of prototype-based unsupervised clustering is to find a number of prototypes or weight vectors $\mathbf{w}_i \in \mathbb{R}^d$, $i = 1, \ldots, n$ representing the data points faithfully, e.g. measured in terms of the average deviation of a data point from its respective closest prototype. There exist different possibilities to achieve this goal: The objective of neural gas [12] is a minimization of the cost function

$$E_{\mathrm{NG}}(W) = \frac{1}{2C(\lambda)} \sum_{i=1}^{n} \int h_\lambda(k_i(\mathbf{v}, W)) \cdot (\mathbf{v} - \mathbf{w}_i)^2 P(\mathbf{v}) d\mathbf{v}$$

where $k_i(\mathbf{v}, W) = |\{\mathbf{w}_j \,|\, (\mathbf{v}-\mathbf{w}_j)^2 < (\mathbf{v}-\mathbf{w}_i)^2\}|$ is the rank of prototype i, $h_\lambda(t)$ is a Gaussian shaped curve such as $h_\lambda(t) = \exp(-t/\lambda)$ with neighborhood range $\lambda > 0$, and $C(\lambda)$ is a normalization constant. Typically, online adaptation takes place by means of a stochastic gradient descent method. The resulting learning rule adapts all prototypes after the presentation of each stimulus by a small step, whereby the rank determines the adaptation strength. Recently, an alternative batch adaptation scheme for this cost function has been proposed which, for a given finite training set, in turn, determines the rank $k_i(\mathbf{v}, W)$ according to fixed prototype locations and the prototype locations as average of all training points weighted according to their rank, until convergence. Batch adaptation can be interpreted as Newton optimization of the cost function, and often a fast convergence can be observed compared to online adaptation.

K-means directly minimizes the quantization error

$$E_{\mathrm{k-means}}(W) = \frac{1}{2}\sum_{i=1}^{n}\int \chi_i(\mathbf{v}, W)\cdot(\mathbf{v}-\mathbf{w}_i)^2 P(\mathbf{v})d\mathbf{v}$$

where W denotes the set of prototypes and $\chi_i(\mathbf{v}, W)$ indicates the receptive field of prototype $\mathbf{w}_i$; i.e. it is one iff the data point $\mathbf{v}$ is closest to $\mathbf{w}_i$ and it is zero, otherwise. Typically, the function is optimized by a batch update scheme for a given finite set of training points $\mathbf{v}_1, \ldots, \mathbf{v}_p$ drawn according to $P(\mathbf{v})$. The algorithm iteratively assigns the given training points to their respective closest prototypes and sets the prototype locations $\mathbf{w}_i$ to the centers of gravity of the current receptive fields as indicated by the assignment, until convergence. This scheme can be interpreted as Newton optimization of $E_{\mathrm{k-means}}$ [1]. Unlike NG, k-means relies on the initialization of prototypes and typically fails to find a global or even good local optimum of $E_{\mathrm{k-means}}$ if the cost function is multimodal. Neural gas offers a very robust alternative that is insensitive to initialization due to neighborhood cooperation.

A third popular unsupervised learning scheme is given by the self-organizing map which includes neighborhood cooperation according to a priorly fixed lattice structure of the neurons. SOM is less flexible than NG, since the lattice topology need not fit the data topology. However, a fixed lattice offers the possibility of easy visualization e.g. using a two-dimensional regular lattice. The original SOM does not possess a cost function. A slight variation of SOM proposed by Heskes [8] has the cost function

$$E_{\mathrm{SOM}}(W) = \frac{1}{2C(\lambda)}\sum_{i=1}^{n}\int \chi_i^*(\mathbf{v}, W)\cdot\sum_{l=1}^{n} h_\lambda(\mathrm{nd}(i,l))\cdot(\mathbf{v}-\mathbf{w}_l)^2 P(\mathbf{v})d\mathbf{v}$$

where $C(\lambda)$ is again a constant, $\mathrm{nd}(i,j)$ denotes the neighborhood range of neuron i and j on the priorly fixed lattice, and $\chi_i^*(\mathbf{v}, W)$ is one for neuron i iff the average $\sum_{l=1}^{n} h_\lambda(\mathrm{nd}(i,l))\cdot(\mathbf{v}-\mathbf{w}_i)^2$ is minimum, otherwise it is zero. For SOM, batch as well as online adaptation schemes are used in practice. SOM constitutes a very popular method for data mining and data visualization [10].

For all formulations, batch adaptation schemes provide an interface towards clustering general proximity data for which only pairwise distances of the data

points are given but in general no embedding within a real-vector space is available. The euclidian distance is substituted by the given proximities, and optimization of prototypes takes place within the discrete space given by the data points, as proposed in [3,11].

3 Supervised Batch NG

Often, additional information in the form of class labels is available. That means, additional class labels y_i are given for every data point $\mathbf{v}_i$. We assume that $y_i \in \mathbb{R}^d$, d being the number of classes. This notion subsumes crisp classification with unary encoded class information as well as fuzzy assignments to d classes. The general aim of clustering provided additional label information can be twofold: either unsupervised clustering, mining, or visualization of the given data whereby the additional cluster information should be taken into account as much as possible to achieve a meaningful result; or direct supervised classification according to the given class labels. Both aims can be incorporated into prototype-based clustering by an extension of the overall cost function such that the information given by the class labels is taken into account. First promising steps into this direction can be found in the aproach [16] for online NG. Each prototype $\mathbf{w}_i$ is equipped with an additional vector $Y_i \in \mathbb{R}^d$ which should represent the class labels of data points in the receptive field as accurately as possible. The cost function of NG is extended to

$$\alpha \cdot E_{\mathrm{NG}}(W) + (1-\alpha) \cdot E_{\mathrm{NG-Y}}(W, Y)$$

whereby $E_{\mathrm{NG}}(W)$, as beforehand, measures the quantization error of prototypes, $E_{\mathrm{NG-Y}}(W, Y)$ measures the error introduced by the difference of class labels of data points and prototypes, and $\alpha \in [0, 1]$ constitutes a weighting of the two objectives. A natural choice of the latter term is

$$E_{\mathrm{NG-Y}}(W, Y) = \frac{1}{2C(\lambda)} \sum_{i=1}^{n} \int h_\lambda(k_i(\mathbf{v}, W)) \cdot (y - Y_i)^2 P(\mathbf{v}) d\mathbf{v}$$

However, as pointed out in [16], this cost function is not differentiable at the borders of the receptive fields. The approach presented in [16] proposes to substitute the term $h_\lambda(k_i(\mathbf{v}, W))$ by an analytic approximation, which yields update rules for W and Y.

Here, we consider a simpler solution. The cost function of supervised NG becomes

$$\begin{aligned} E_{\mathrm{BSNG}}(W, Y) = \alpha \cdot \frac{1}{2C(\lambda)} \sum_{i=1}^{n} \int h_\lambda(k_i(\mathbf{v}, y, W, Y)) \cdot (\mathbf{v} - \mathbf{w}_i)^2 P(\mathbf{v}) d\mathbf{v} \\ + (1-\alpha) \cdot \frac{1}{2C(\lambda)} \sum_{i=1}^{n} \int h_\lambda(k_i(\mathbf{v}, y, W, Y)) \cdot (y - Y_i)^2 P(\mathbf{v}) d\mathbf{v} \end{aligned}$$

where $k_i(\mathbf{v}, y, W, Y) = |\{\mathbf{w}_j \mid \alpha(\mathbf{v} - \mathbf{w}_j)^2 + (1-\alpha)(y - Y_j)^2 < \alpha(\mathbf{v} - \mathbf{w}_i)^2 + (1-\alpha)(y - Y_i)^2\}|$ denotes the rank of prototype i measured according to the closeness of the current data point and the prototype weight and labeling. Thus, it constitutes an average over the quantization error of the prototypes and labels, weighted according to the rank taken over both, prototype and label closeness. This cost function corresponds to original (unsupervised) NG applied to the embedded data points $(\sqrt{\alpha} \cdot \mathbf{v}, \sqrt{1-\alpha} \cdot y)$, thus it is differentiable at borders of receptive fields.

Batch adaptation schemes suppose that a finite set of training data $(\mathbf{v}_1, y_1)$, $\ldots$, $(\mathbf{v}_p, y_p)$ is given in advance. The cost function becomes

$$\hat{E}_{\mathrm{BSNG}}(W, Y) = \frac{1}{2C(\lambda)} \sum_{i=1}^{n} \sum_{j=1}^{p} h_\lambda(k_i(\mathbf{v}_j, y_j, W, Y)) \\ \left(\alpha \cdot (\mathbf{v}_j - \mathbf{w}_i)^2 + (1-\alpha) \cdot (y_j - Y_i)^2\right)$$

Batch optimization determines in turn the hidden variables $k_{ij} := k_i(\mathbf{v}_j, y_j, W, Y)$ and the weights and labels W and Y until convergence. This yields the following update rules of **supervised-batch-NG (BSNG)**:

(1) For given W, Y, set $k_{ij} = |\{\mathbf{w}_l \mid \alpha \cdot (\mathbf{v}_j - \mathbf{w}_l)^2 + (1-\alpha) \cdot (y_j - Y_l)^2 \leq \alpha \cdot (\mathbf{v}_j - \mathbf{w}_i)^2 + (1-\alpha) \cdot (y_j - Y_i)^2\}|$ as the rank of prototype i given $\mathbf{v}_j$.
(2) For fixed k_{ij}, set $\mathbf{w}_i = \sum_j h_\lambda(k_{ij}) \cdot \mathbf{v}_j / \sum_j h_\lambda(k_{ij})$, and $Y_i = \sum_j h_\lambda(k_{ij}) \cdot y_j / \sum_j h_\lambda(k_{ij})$.

Note that the assignments of the receptive fields and the rank depend on the closeness of the prototype as well as the correctness of its class label. The new prototypes are determined as center of gravity of all data points weighted according to this rank, the same holds for the class labels.

One can consider this scheme as Newton optimization of the cost term $\hat{E}_{\mathrm{BSNG}}(W, Y)$: the updates within a Newton scheme are $\Delta(W, Y) = -J \cdot H^{-1}$ where J is the Jacobian of $\hat{E}$ and H is the Hessian. Since k_{ij} is locally constant, we get up to sets of measure zero $\partial \hat{E}_{\mathrm{BSNG}} / \partial \mathbf{w}_i = (\alpha / C(\lambda)) \cdot \sum_j h_\lambda(k_{ij})(\mathbf{w}_i - \mathbf{v}_j)$ and $\partial \hat{E}_{\mathrm{BSNG}} / \partial Y_i = (\alpha / C(\lambda)) \cdot \sum_j h_\lambda(k_{ij})(Y_i - y_j)$. The Hessian equals a diagonal matrix with entries $\partial^2 \hat{E}_{\mathrm{BSNG}} / \partial \mathbf{w}_i^2 = (\alpha / C(\lambda)) \cdot \sum_j h_\lambda(k_{ij})$ and $\partial \hat{E}_{\mathrm{BSNG}} / \partial Y_i^2 = (\alpha / C(\lambda)) \cdot \sum_j h_\lambda(k_{ij})$. Obviously, this corresponds to the batch updates of W and Y.

Before proving convergence of supervised batch NG in a general framework, which also incorporates supervised batch SOM and supervised k-means, we test the algorithm on several datasets. So far, the experiments are preliminary, and further studies, in particular in combination with more powerful approaches as will be discussed in section 5, are the subject of future work.

3.1 Artificial Data

The main difference of NG with posterior labeling and BSNG consists in the fact that the rank assignments also take the fact into account whether the labels

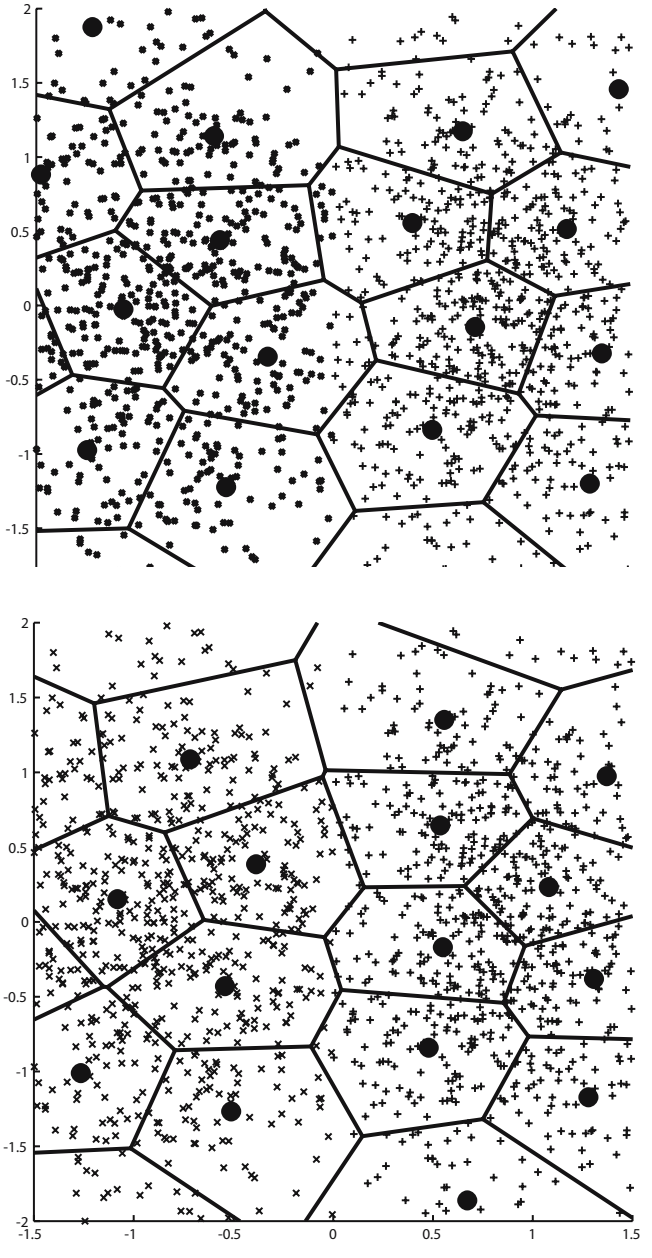

Fig. 1. Receptive fields obtained by batch NG (top) and BSNG (bottom) on an artificial two-dimensional data set. Obviously, the incorporation of the label information for BSNG yields a better separation of the two classes; the prototype locations follow the classification boundary

fit. This has the effect that the prototypes of BSNG better account for cluster borders of labeled data points, whereas NG only follows the overall statistics. The parameter α controls the strength of the label contribution, $\alpha = 1$ corresponds to standard NG. This effect can be clearly observed in the following example. We consider two Gaussian clusters labeled by 0 resp. 1, whereby points with x-component at least 0 are dropped for class 0, and points with x-component at most 0 are dropped for class 1. Hence, the classes are well separated, whereby a couple of data points lies close to the decision boundary. Fig. 1 shows a typical result of the receptive fields of the prototypes obtained by batch NG and BSNG with mixing parameter $\alpha = 0.1$, respectively. Thereby, prototypes of NG are labeled by a majority vote within the receptive field. Fuzzy labels for BSNG arise automatically during training, these are turned into crisp classes based on the largest component of the label vector. Obviously, BSNG well approximates the decision border, whereas NG yields a couple of errors at this region. This corresponds to the classification accuracy of 99.1% for BSNG and 97.8% for NG.

3.2 Iris Data

We train batch NG and BSNG using 9 prototypes on the well-known iris dataset [13], which consists in the task to classify 150 points characterized by 4 real-valued attributes into 3 classes of equal size. Class 1 is well separated from class 2 and 3, but classes 2 and 3 slightly overlap. For each run, the set is randomly divided into a training and test set of equal size, and averages over 50 runs are reported. The neighborhood range λ is multiplicatively annealed starting from 4.5 over 100 training epochs. Different values of the mixing parameter α are

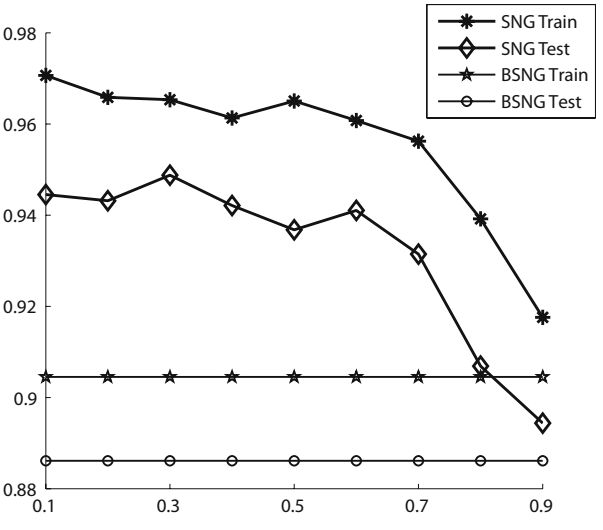

Fig. 2. Accuracy on the training and test set achieved by supervised batch NG (BSNG) and batch NG (BNG) on the iris dataset for different mixing parameters α

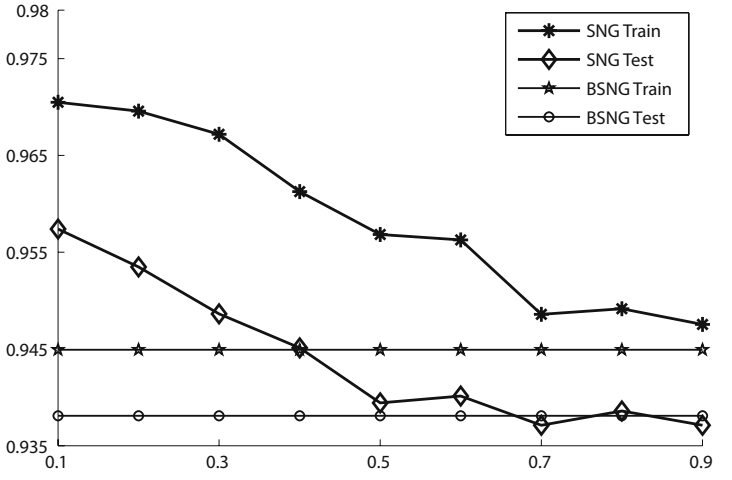

Fig. 3. Accuracy on the training and test set achieved by supervised batch NG (BSNG) and batch NG (BNG) on the Wisconsin breast cancer dataset for different mixing parameters α

reported. The classification accuracy on the training and test set for BSNG and, in comparison, for NG with posterior labeling can be observed in Fig.2. Obviously, the classification accuracy becomes better for smaller α, i.e. more emphasis of the given data labels. Thereby α must not become 0 which corresponds to a pure label adaptation without adaptation of the prototypes. $\alpha = 1$ corresponds to standard NG. Obviously, the classification accuracy of simple NG is inferior compared to the supervised version due to the overlap of classes 2 and 3 which is not accounted for by the overall statistics of the input vectors. The algorithm reported in [16] (Gaussian approximation) achieves (in a single run with parameter $\alpha = 0.5$) a training set accuracy of 0.85 and test set accuracy of 0.91 using 9 prototypes, thus it is better than post labeled NG, but worse compared to supervised batch NG for this parameter choice.

3.3 Wisconsin Breast Cancer

The Wisconsin breast cancer data consists of nearly 600 data points described by 30 real-valued input features which are to be separated into 2 classes. Training uses 20 prototypes and the same parameters as beforehand, starting with an initial neighborhood range 10. The results are presented in Fig.3. As before, a larger emphasis on the correctness of the labels yields a better classification accuracy which is superior to neural gas. Interestingly, the approach presented in [16] which relies on a different supervised extension of NG (Gaussian approximation of the rank) achieves an accuracy of 0.92 for the training set and 0.91 on the test set for a mixing parameter 0.5, which is in this case worse than the result obtained by NG with posterior labeling.

4 Perspectives – General Supervised Batch Clustering

Now, we introduce a general cost function and a batch adaptation scheme for supervised prototype-based clustering which extends SOM, NG, and k-means by two aspects: the integration and adaptation of a (possibly fuzzy) labeling and potential adaptation of metric parameters according to the overall objective. The proposal includes BSNG introduced above as a special case.

Note that, for a finite set of training data $\mathbf{v}_1, \ldots, \mathbf{v}_p$, the cost function of SOM, NG, and k-means can be written as

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

where (neglecting constants) $f_1(k_{ij}(W))$ is given by $\chi_i(\mathbf{v}_j, W)$, $h_\lambda(k_i(\mathbf{v}_j, W))$, or $\chi_i^*(\mathbf{v}_j, W)$, respectively, and $f_2^{ij}(W)$ constitutes the squared distance $(\mathbf{v}_j - \mathbf{w}_i)^2$ or the average $\sum_{l=1}^{n} h_\lambda(\mathrm{nd}(i,l)) \cdot (\mathbf{v}_j - \mathbf{w}_l)^2$, respectively. We extend this cost function in two respects: we assume that each training vector $\mathbf{v}_j$ is accompanied by a priorly given label y_j and each prototype $\mathbf{w}_i$ is equipped by a label Y_i. The labels Y are adapted according to the data labels y during training such that Y_i represents the class information of the data points in the receptive field of $\mathbf{w}_i$, as introduced for batch NG in the previous section. In addition, we extend the metric by possibly adaptive metric parameters Λ which allow a better shaping of the classifier according to the given task. Adaptive metric parameters are quite common in supervised as well as unsupervised clustering [5,9,14,6,7], since the choice of the metric severely influences the final classification ability of the models. Therefore, this possibility should be included in a general framework.

The general cost function of supervised clustering becomes

$$\hat{E}(W, \Lambda, Y) = \sum_{i=1}^{n} \sum_{j=1}^{p} f_1(k_{ij}(W, \Lambda, Y)) \cdot \Big(\alpha \cdot f_2^{ij}(W, \Lambda) + (1 - \alpha) \cdot d(Y_i, y_j) \Big)$$

where d measures the distance of the two labels (e.g. the squared euclidian distance) and $\alpha \in [0, 1]$ controls the weighting of the two parts of this cost function, unsupervised vector quantization and correct labeling of the prototypes. Thereby, the parameters might be subject to constraints, i.e. $\lambda \in X_\Lambda$, $W \in X_W$, and $Y \in X_Y$.

We introduce a general batch optimization scheme of $\hat{E}(W, \Lambda, Y)$ and we show its convergence. We assume that $k_{ij}(W, \Lambda, Y)$ constitutes a function which maps given parameter values to unique assignments (possibly after introducing some order) within a finite set X_K such that it optimizes $\hat{E}$ for fixed parameters W, Λ, and Y. Thereby, constraints might apply: for SOM and k-means, the assignments k_{ij} constitute unary vectors for fixed j; for NG, they constitute a permutation of $\{0, \ldots, n-1\}$. This assumption allows us to optimize the cost function $\hat{E}$ consecutively for the parameters of $\hat{E}$ and hidden variables k_{ij} connected to the function $k_{ij}(W, \Lambda, Y)$. General batch optimization of $\hat{E}$ proceeds in two steps;

after an initialization of the parameters, the following two optimization steps are performed until convergence

(1) for given W, Λ, Y, find k_{ij} in X_K (possibly subject to constraints) such that

$$\sum_{i=1}^{n}\sum_{j=1}^{p} f_1(k_{ij}) \cdot \Big(\alpha \cdot f_2^{ij}(W,\Lambda) + (1-\alpha)\cdot d(Y_i, y_j)\Big)$$

is minimum;

(2) for given k_{ij}, find $W \in X_W$, $\Lambda \in X_\Lambda$, and $Y \in X_Y$ such that

$$\sum_{i=1}^{n}\sum_{j=1}^{p} f_1(k_{ij}) \cdot \Big(\alpha \cdot f_2^{ij}(W,\Lambda) + (1-\alpha)\cdot d(Y_i, y_j)\Big)$$

is minimum.

To show convergence, consider the function

$$Q(W,\Lambda,Y,W',\Lambda',Y') := \\ \textstyle\sum_{i=1}^{n}\sum_{j=1}^{p} f_1(k_{ij}(W,\Lambda,Y)) \cdot \Big(\alpha \cdot f_2^{ij}(W',\Lambda') + (1-\alpha)\cdot d(Y_i', y_j)\Big)\ .$$

Note that $\hat{E}(W,\Lambda,Y) = Q(W,\Lambda,Y,W,\Lambda,Y)$. Assume batch optimization starts with W, Λ, Y and computes optimum k_{ij} and W', Y', λ' based thereon. We find

$$\hat{E}(W',\Lambda',Y') = Q(W',\Lambda',Y',W',\Lambda',Y') \le Q(W,\Lambda,Y,W',\Lambda',Y')$$

because $k_{ij}(W',\Lambda',Y')$ are optimum assignments given W', Y', λ'. Further,

$$\hat{E}(W,\Lambda,Y) = Q(W,\Lambda,Y,W,\Lambda,Y) \ge Q(W,\Lambda,Y,W',\Lambda',Y')$$

since W', Y', λ' are optimum assignments for given k_{ij}. Thus,

$$\hat{E}(W,\Lambda,Y) - \hat{E}(W',\Lambda',Y') = \\ Q(W,\Lambda,Y,W,\Lambda,Y) - Q(W,\Lambda,Y,W',\Lambda',Y') \\ + Q(W,\Lambda,Y,W',\Lambda',Y') - Q(W',\Lambda',Y',W',\Lambda',Y') \ge 0\,,$$

i.e. the cost function does not increase in batch optimization. Since the assignments k_{ij} are unique and they stem from a finite set, the algorithm must converge in a finite number of steps. This shows the convergence of the algorithm in a finite number of optimization steps for all optimization schemes of this form, in particular supervised batch NG.

Often, the values W and Λ stem from a real-vector space. In this case, it is possible to show that the algorithm, in general, converges to a local optimum of the cost function $\hat{E}$. For discrete W and Λ, this is not possible since the term 'local optimum' is not defined without specifying an additional neighborhood relation in the discrete sets X_W and X_Λ. Assume continuous X_W and X_Λ and assume, that for the final solution W, Λ, Y of batch optimization an open neighborhood exists such that $k_{ij}(W,\Lambda,Y)$ is constant in this neighborhood (this is usually the case for any given finite data set). In this case $\hat{E}(\cdot)$ and $Q(W,\Lambda,Y,\cdot)$ coincide in a vicinity of the solution, i.e. a local optimum of the latter is also a local optimum of the first one. Thus, a local optimum of $\hat{E}$ has been found in this case.

5 Future Approaches

This general formulation opens the way towards a couple of concrete algorithms for different application areas. We shortly mention a few possibilities which seem particularly promising and which experimental investigation will be the subject of future research.

5.1 Supervised Batch SOM

As already mentioned, the general formulation of batch optimization includes the possibility to extend SOM by a supervised component. Since SOM is subject to a fixed lattice structure which need not fit the data topology, it can be expected that the classification accuracy is usually worse compared to supervised batch NG. However, supervised SOM offers the possibility of data visualization if a low dimensional regular lattice is used. In this case, supervised components can be naturally integrated into the visualization. This is beneficial in particular for high-dimensional data where accumulated noise might blur the information hidden in the data. Additional label information allows to focus on the relevant parts of the data and visualize these aspects. The principle of integration of additional information in unsupervised learning has been introduced in [9,14], for example. Unlike this proposal, supervised batch SOM offers a simple alternative for the special case of additional label information.

5.2 Relevance Learning

Clustering crucially depends on the choice of the underlying metrics which determines the winner for a given data point. If the chosen metric is not appropriate for the task at hand, the classification fails. This is particularly pronounced for high-dimensional data where noise can accumulate and disrupt the information available in the data. Because of this fact, the principle of relevance learning as introduced in [7] has proven beneficial for prototype based clustering. The basic idea is to substitute the standard euclidian metric by a weighted version

$$(\mathbf{v}_j - \mathbf{w}_i)^2_\Lambda = \sum_{l=1}^{k} \Lambda_l^2((v_j)_l - (w_i)_l)^2$$

where Λ_l are relevance parameters with $\sum_l \Lambda_l = 1$ which scale the dimensions according to their significance for the given task. The general formulation of batch optimization as introduced above allows us to adapt the relevance parameters automatically during training such that they are optimum adapted according to the given cost function. For the diagonal metric, the optimization task in **(2)** can be solved analytically for Λ and yields

$$\Lambda_l = \frac{\left(\sum_{ij} h_\lambda(k_{ij}) \cdot ((v_j)_l - (w_i)_l)^2\right)^{-1}}{\sum_{l'} \left(\sum_{ij} h_\lambda(k_{ij}) \cdot ((v_j)_{l'} - (w_i)_{l'})^2\right)^{-1}}$$

for NG, i.e. weight vectors similar to the diagonal Mahalanobis distance arise, whereby the rank of the prototypes weighted according to the appropriateness of the weight and label is taken into account.

5.3 Median Clustering

Often, data are not embedded in a euclidian vector space, rather, pairwise proximities d_{ij} which describe the distance of $\mathbf{v}_i$ and $\mathbf{v}_j$ are available. A variety of clustering algorithms for proximity data has been proposed [3,11,15,17], however, only few possibilities to train prototype-based methods for supervised classification of proximity data are available. The general framework as introduced above opens the way towards a very simple and intuitive method: supervised median clustering. Thereby, optimization of W is restricted to the discrete set $X_W = \{\mathbf{v}_1, \ldots, \mathbf{v}_p\}$ given by the training patterns. In the optimization step **(2)**, the generalized median, i.e. the data point $\mathbf{v}_i$ which minimizes the considered sum is taken as $\mathbf{w}_j$. This principle has been introduced in [11] for SOM and, including a proof of convergence, in [3] for NG. The transfer to supervised NG or SOM is immediate, whereby optimization can take place either by extensive search, or incorporating (exact or approximate) acceleration as discussed in [2].

References

1. L. Bottou and Y. Bengio (1995), Convergence properties of the k-means algorithm, in *NIPS 1994*, 585-592, G. Tesauro, D.S. Touretzky, and T.K. Leen (eds.), MIT.
2. B. Conan-Guez, F. Rossi, and A. El Golli (2005), A fast algorithm for the self-organizing map on dissimilarity data, in *Workshop on Self-Organizing Maps*, 561-568.
3. M. Cottrell, B. Hammer, A. Hasenfuss, and T. Villmann (2006), Batch and median neural gas, *Neural Networks*, to appear.
4. K. Crammer, R. Gilad-Bachrach, A. Navot, and N. Tishby (2002), Margin analysis of the LVQ algorithm, *NIPS'2002*.
5. I. Gath, and A.B. Geva (1989), Unsupervised optimal fuzzy clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**(7):773-781.
6. B. Hammer, M. Strickert, and T. Villmann (2005), Supervised neural gas with general similarity measure, *Neural Processing Letters* **21**(1), 21-44.
7. B. Hammer, and T. Villmann (2002), Generalized relevance learning vector quantization, *Neural Networks* **15**, 1059-1068.
8. T. Heskes (2001), Self-organizing maps, vector quantization, and mixture modeling, *IEEE Transactions on Neural Networks*, **12:**1299-1305.
9. S. Kaski and J. Sinkkonen (2004), Principle of learning metrics for data analysis, *Journal of VLSI Signal Processing*, special issue on Machine Learning for Signal Processing, **37:** 177-188.
10. T. Kohonen (1995), *Self-Organizing Maps*, Springer.
11. T. Kohonen and P. Somervuo (2002), How to make large self-organizing maps for nonvectorial data, *Neural Networks* **15:**945-952.
12. 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.

13. D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz (1998), UCI Repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.
14. J. Peltonen, A. Klami, and S. Kaski (2004), Improved learning of Riemannian metrics for exploratory analysis, *Neural Networks*, **17**:1087-1100.
15. S. Seo and K. Obermayer (2004), Self-organizing maps and clustering methods for matrix data, *Neural Networks* **17:**1211-1230.
16. T. Villmann, B. Hammer, F. Schleif, T. Geweniger, and W. Herrmann (2006), Fuzzy classification by fuzzy labeled neural gas, *Neural Networks*, accepted.
17. S. Zhong and J. Ghosh (2003), A unified framework for model-based clustering, *Journal of Machine Learning Research* **4**:1001-1037.