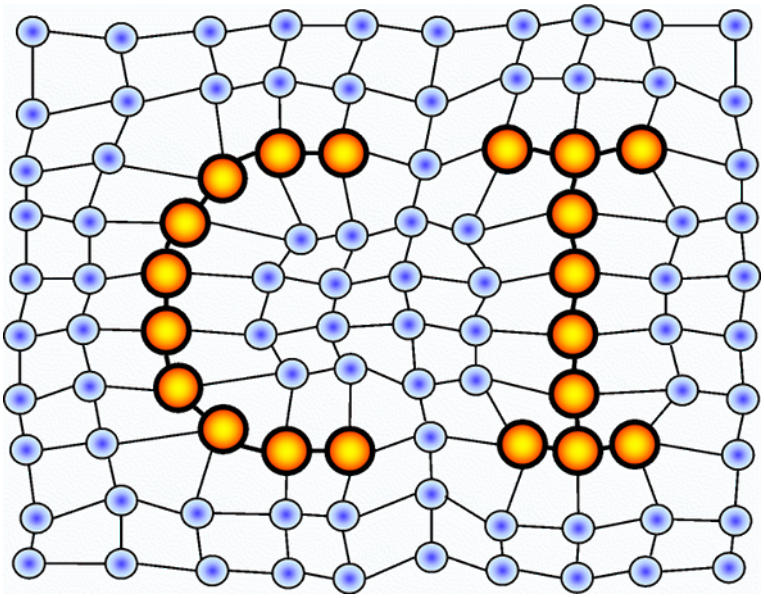
MIWOCI 2011, Mittweida Workshop on Computational Intelligence

F.-M. Schleif, T.Villmann (Eds.)

Machine Learning Reports MLR-2011-06



MIWOCI 2011, MITTWEIDA WORKSHOP ON COMPUTATIONAL INTELLIGENCE

Impressum

Publisher:	University of Applied Sciences Mittweida Technikumplatz 17, 09648 Mittweida, Germany
Editor:	Prof. Dr. Thomas Villmann Dr. Frank-Michael Schleif
Techical-Editor:	Dr. Frank-Michael Schleif
Contact:	fschleif@techfak.uni-bielefeld.de
URL:	http://techfak.uni-bielefeld.de/∼ fschleif/mlr/mlr.html
ISSN:	1865-3960

Figure 1: MiWoCi 2011

Contents

Third Mittweida Workshop on Computational Intelligence

F.-M. Schleif[1]

1 Third Mittweida Workshop on Computational Intelligence

From June 27th to June 29th, 2011, 15 scientists from the University of Bielefeld, University of Siegen, University of Groningen (NL), University of Birmingham (UK) and the University of Applied Sciences Mittweida met in Mittweida, Germany, to continue the tradition of the Mittweida Workshops on Computational Intelligence - *MiWoCi'2011*. The aim was to present their current research, discuss scientific questions, and exchange their ideas. The seminar centered around topics in machine learning, signal processing and data analysis, covering fundamental theoretical aspects as well as recent applications, partially in the frame of innovative industrial cooperations. This volume contains a collection of extended abstracts which accompany some of the talks to give insight into the research presented in Mittweida.

Apart from the scientific merrits, this year's seminar came up with a few highlights which demonstrate the excellent possibilities offered by the surroundings of Mittweida. This year adventures were explored under intensive sunlight and very good weather conditions. The participants climbed to the high forests of Mittweida (Kletterwald) and enjoyed the exciting and fearing adventures provided on the top of the trees. Multiple jump offs from the *Wahnsinn* tour at a height of at least 20 meters were reported, but no participants were harmed. During a *wild water* journey (Paddeltour) the outstanding fitness of the researchers was demonstrated and some of them also demonstrated their braveness by swimming in the rapids followed by a nice barbecue.

Our particular thanks for a perfect local organization of the workshop go to Thomas Villmann as spiritus movens of the seminar and his PhD and Master students.

Bielefeld, October, 2011
Frank-M. Schleif

[1]E-mail: fschleif@techfak.uni-bielefeld.de
[2]University of Bielefeld, CITEC, Theoretical Computer Science, Leipzig, Germany

Enhancing M|G|RLVQ by quasi step discriminatory functions using 2^{nd} order training

Marc Strickert[1,2]

Acknowledgements: Parts of this work were possible thanks to fundings in the DFG Graduiertenkolleg 1564 "Imaging New Modalities". Many thanks to Prof. Thomas Villmann for organizing and hosting the inspiring 3nd Mittweidaer Workshop on Computational Intelligence 2011. I also thank Michael Biehl for discussions and for presenting his alternative view on enhanced optimization of LVQ networks based on cost functions.

Abstract

By combining very steep squashing functions and normalization as parts of the cost function of generalized learning vector quantization (GLVQ) and its descendants, vector label misclassification gets directly minimized in the limit of class-separating sigmoids towards step functions. To cope with the resulting difficult optimization problem a switch from standard stochastic gradient descent to a quasi-2^{nd} order Newton batch optimization scheme is proposed. Results for weighted squared Euclidean distance (GRLVQ) and adaptive matrix metrics (MRLVQ) are faster obtained and usually show a better class discrimination than traditional implementations of GRLVQ and MRLVQ. Code is available online.
Keywords: [M|G]RLVQ, batch 2^{nd} order learning cost function minimization.

1 Introduction

Learning vector quantization (LVQ) is a powerful machine learning scheme for the classification of labeled data vectors [18]. Its main strength is the reduction of complex data clouds to usually a few class-specific prototypes. This representation

[1]E-mail: strickert@informatik.uni-siegen.de
[2]University of Siegen, Institute for Vision and Graphics

allows a fast execution of trained models for classifying unknown data. The formulation of Generalized LVQ (GLVQ) replaced heuristic updates of prototype by a cost function being optimized by stochastic gradient descent [17]. Further improvements, including attribute assessment, were obtained by introducing adaptive dimension scaling terms to the squared Euclidean distance subject to the GLVQ cost function, leading to generalized relevance LVQ (GRLVQ) [16]. Further extensions to other similarity measures followed, for example, for adaptive Pearson correlation [13] and for adaptive matrix metrics described by quadratic forms [9].

Adaptive matrix metrics enable very flexible comparison of data vectors, involving weighted pairs of vector attributes contributing to the distance. Thus, covariance structure can be dynamically weighted. Diagonal matrices are directly related to the weighted squared Euclidean distance. For general full-rank semi-definite matrices, affine transformations of the data space may allow to enhance class-related densities. For full rank matrices the large number of degrees of freedom may easily lead to overfitting distance dependent models. One regularization strategy puts constraints on the eigen value spectrum of the full attribute mixing matrix [6], another strategy directly expresses the weight matrix a matrix square of a low-rank rectangular matrix [6, 7]. For such rectangular matrices initialization is a challenge, because the induced metric properties are very different from the intuitive Euclidean distance, and interpretation of optimized matrices is an interesting problem, because many equally valid solutions are allowed by the cost function.

A MATLAB/GNU-Octave framework is presented for approximate 2^{nd} order batch optimization of GRLVQ and MRLVQ. Using this framework, one of the yet underrepresented aspects related to the logistic term in the underlying cost functions is highlighted.

2 Generalized Relevance LVQ (GRLVQ) revisited

Let $\mathbf{X} = \{(\mathbf{x}^i, \boldsymbol{l}^i) \in \mathbb{R}^M \times \{1, \ldots, c\} \mid i = 1, \ldots, N\}$ be a training data set with M-dimensional elements $\mathbf{x}^k = (x_1^k, \ldots, x_M^k)$ to be classified and c classes. Prototypes $\mathbf{W} = \{\boldsymbol{w}^1, \ldots, \boldsymbol{w}^K\}$, $\boldsymbol{w}^i = (w_1^i, \ldots, w_M^i, \boldsymbol{l}^i) \in \mathbb{R}^M \times \{1, \ldots, c\}$, are used for data representation in data space.

The *normalized* generic classification cost function to be minimized is [16]:

$$\mathrm{E}_{\mathsf{GRLVQ}} := \frac{1}{N} \cdot \sum_{i=1}^{N} \mathrm{g}_\sigma \left(\mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i) \right) \text{ with } \mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i) = \frac{\mathrm{d}^+_{\boldsymbol{\lambda}}(\mathbf{x}^i) - \mathrm{d}^-_{\boldsymbol{\lambda}}(\mathbf{x}^i)}{\mathrm{d}^+_{\boldsymbol{\lambda}}(\mathbf{x}^i) + \mathrm{d}^-_{\boldsymbol{\lambda}}(\mathbf{x}^i)}, \mathrm{d}_{\boldsymbol{\lambda}}(\mathbf{x}) := \mathrm{d}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}) \,. \tag{1}$$

The misclassification costs of all patterns are summed up, whereby $\mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i)$ serves as quality measure of the classification depending on the degree of fit of the presented pattern $\mathbf{x}^i$ and the two closest prototypes, $\boldsymbol{w}^{i+}$ representing the same label as $\mathbf{x}^i$ and $\boldsymbol{w}^{i-}$ representing a different label. Cost minimization depends on the prototype locations in the weight space and a set of adaptive parameters $\boldsymbol{\lambda}$ of the measure $\mathrm{d}_{\boldsymbol{\lambda}}(\mathbf{x}) = \mathrm{d}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w})$ comparing pattern and prototype. Since gradients shall be used for cost minimization, like Eqn. 1, d must be differentiable almost everywhere.

Partial derivatives of $\mathrm{E}_{\mathsf{GRLVQ}}$ yield the generic update formulas for the closest correct

and the closest wrong prototype and the metric weights:

$$\frac{\partial \mathrm{E}_{\mathsf{GRLVQ}}}{\partial \boldsymbol{w}^{i+}} = -\mathrm{g}'_\sigma\big(\mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\big) \cdot \frac{2 \cdot \mathrm{d}^-_{\boldsymbol{\lambda}}(\mathbf{x}^i)}{\big(\mathrm{d}^+_{\boldsymbol{\lambda}}(\mathbf{x}^i) + \mathrm{d}^-_{\boldsymbol{\lambda}}(\mathbf{x}^i)\big)^2} \cdot \frac{\partial \mathrm{d}^+_{\boldsymbol{\lambda}}(\mathbf{x}^i)}{\partial \boldsymbol{w}^{i+}}\,, \tag{2}$$

$$\frac{\partial \mathrm{E}_{\mathsf{GRLVQ}}}{\partial \boldsymbol{w}^{i-}} = \mathrm{g}'_\sigma\big(\mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\big) \cdot \frac{2 \cdot \mathrm{d}^+_{\boldsymbol{\lambda}}(\mathbf{x}^i)}{\big(\mathrm{d}^+_{\boldsymbol{\lambda}}(\mathbf{x}^i) + \mathrm{d}^-_{\boldsymbol{\lambda}}(\mathbf{x}^i)\big)^2} \cdot \frac{\partial \mathrm{d}^-_{\boldsymbol{\lambda}}(\mathbf{x}^i)}{\partial \boldsymbol{w}^{i-}}\,, \text{ and} \tag{3}$$

$$\frac{\partial \mathrm{E}_{\mathsf{GRLVQ}}}{\partial \boldsymbol{\lambda}} = -\mathrm{g}'_\sigma\big(\mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\big) \cdot \frac{2 \cdot \partial \mathrm{d}^+_{\boldsymbol{\lambda}}(\mathbf{x}^i)/\partial \boldsymbol{\lambda} \cdot \mathrm{d}^-_{\boldsymbol{\lambda}}(\mathbf{x}^i) \;-\; 2 \cdot \mathrm{d}^+_{\boldsymbol{\lambda}}(\mathbf{x}^i) \cdot \partial \mathrm{d}^-_{\boldsymbol{\lambda}}(\mathbf{x}^i)/\partial \boldsymbol{\lambda}}{\big(\mathrm{d}^+_{\boldsymbol{\lambda}}(\mathbf{x}^i) + \mathrm{d}^-_{\boldsymbol{\lambda}}(\mathbf{x}^i)\big)^2} \tag{4}$$

2.1 Two adaptive distance measures

Although there is currently strong progress on particular similarity measures such as adaptive Pearson correlation [13], divergence measures [5, 2] and functional metrics [1], we focus on weighted squared Euclidean distance and on matrix metrics based on parametric quadratic forms, both sharing the property that decision boundaries between prototypes are straight hyperplanes.

Weighted squared Euclidean distance

The weighted GRLVQ metric, except for weight squaring equal to the original one [16], is given by

$$\mathrm{d}^{\mathsf{E2}}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}) = \langle (\mathbf{x} - \boldsymbol{w})^{\cdot 2}, \boldsymbol{\lambda}^{\cdot 2} \rangle = \sum_{n=1}^{N} (x_n - w_n)^2 \cdot \lambda_n^2 \quad \text{with} \tag{5}$$

$$\frac{\partial \mathrm{d}^{\mathsf{E2}}_{\boldsymbol{\lambda}}}{\partial \boldsymbol{w}} = -2 \cdot (\mathbf{x} - \boldsymbol{w}) \circ \boldsymbol{\lambda}^{\cdot 2} \quad \text{and} \tag{6}$$

$$\frac{\partial \mathrm{d}^{\mathsf{E2}}_{\boldsymbol{\lambda}}}{\partial \boldsymbol{\lambda}} = 2 \cdot (\mathbf{x} - \boldsymbol{w})^{\cdot 2} \circ \boldsymbol{\lambda}\,. \tag{7}$$

Therein $\circ$ denotes the element-wise Hadamard multiplication and the exponent $^{\cdot 2}$ refers to the element-wise vector square.

Matrix metrics

Similar to the weighted squared Euclidean distance, matrix metrics can be defined as

$$\mathrm{d}^{\mathsf{M}}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}) = (\mathbf{x} - \boldsymbol{w}) \cdot \boldsymbol{\lambda} \cdot \boldsymbol{\lambda}^{\mathsf{T}} \cdot (\mathbf{x} - \boldsymbol{w})^{\mathsf{T}} \quad \text{with} \tag{8}$$

$$\frac{\partial \mathrm{d}^{\mathsf{M}}_{\boldsymbol{\lambda}}}{\partial \boldsymbol{w}} = -2 \cdot (\mathbf{x} - \boldsymbol{w}) \cdot \boldsymbol{\lambda} \cdot \boldsymbol{\lambda}^{\mathsf{T}} \quad \text{and} \tag{9}$$

$$\frac{\partial \mathrm{d}^{\mathsf{M}}_{\boldsymbol{\lambda}}}{\partial \boldsymbol{\lambda}} = 2 \cdot (\mathbf{x} - \boldsymbol{w})^{\mathsf{T}} \cdot \big((\mathbf{x} - \boldsymbol{w}) \cdot \boldsymbol{\lambda}\big)\,. \tag{10}$$

In these terms, $\boldsymbol{\lambda} \in \mathbb{R}^{M \times d}$ denotes a parameter matrix, inducing positive semi-definite matrices $\boldsymbol{\lambda} \cdot \boldsymbol{\lambda}^{\mathsf{T}}$ with a maximum rank of d. The special case of $\boldsymbol{\lambda}$ being the identity matrix relates back to the squared Euclidean distance. Usually low-rank matrix metrics are sufficient to provide good classification models [8]. This efficient formulation can be seen also as a direct answer to a work on GRLVQ stating [14]:

> "In the present work, the full matrix has not been used for three reasons: $\langle 1 \rangle$ its computational time complexity is $\mathcal{O}(M^2)$ instead of $\mathcal{O}(M)$ for each pattern; $\langle 2 \rangle$ data preprocessing, such as principal component analysis PCA, could be used beforehand to scale and rotate the input data in order to minimize the correlations between the dimensions; and $\langle 3 \rangle$ it is unclear how the positive definiteness of the distance matrix can be obtained as a result of the parameter update dynamic."

2.2 The squashing function

Although the sigmoid distance ratio transfer function

$$\mathrm{g}_\sigma(x) \quad = \quad \mathrm{sgd}(x) = \frac{1}{1+\exp(-\sigma \cdot x)} \in (0;1) \quad \text{with} \tag{11}$$

$$\mathrm{g}'_\sigma(x) \quad = \quad \sigma \cdot \mathrm{g}_\sigma(x) \cdot \big(1 - \mathrm{g}_\sigma(x)\big) \tag{12}$$

with increasing steepness $\sigma(t)$ proportional to time was originally proposed [17] for wrapping $\mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x})$ in Eqn. 1, not much attention has been paid to it in follow-up publications. One work was ignorant enough to state [14]:

> "However, the [GRLVQ] convergence considerations still hold, if the identity function $\mathrm{g}_\sigma(x) = \mathsf{id}(x)$ is chosen as wrapper."

In the context of Fig. 2 this statement is related to the fact that $\sigma = 1$ yields almost linear response. Here it is argued that sigmoids with steep slopes related to $\sigma \in [10; 500]$ are vital to creating good classifiers, because in the limit $\sigma \to \infty$ quasi-step functions yield almost 1 for misclassified samples and almost 0 for correctly assigned samples. This leads to an approximate error counting using the cost function in Eqn. 1, that is, optimization really *minimizes the number of misclassifications*.

Obviously, training gets more difficult the higher σ, because most errors create almost vanishing gradient $\mathrm{g}'_\sigma(x)$, while near threshold errors around $x = 0.5$ generate very strong gradient values. Thus, first-order optimization might not be the best choice for minimizing this error. Second-order off-the-shelf methods like those realized by the MATLAB optimization toolbox for unconstrained problems (fminunc) can be used. Here a free implementation for a method combining ideas from Broyden, Fletcher, Goldfarb, and Shanno (BFGS), and particularly its memory-limited counterpart (l-BFGS) with lightweight approximation of the Hessian matrix is preferred [19].

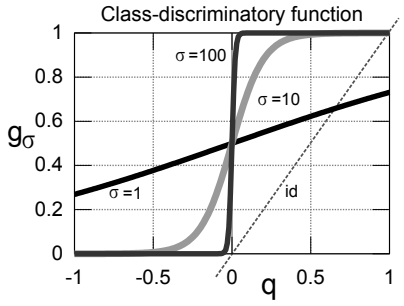


Figure 2: Sigmoid squashing function, controlled by σ.

3 Experiments

Results for two benchmark data sets are reported, the Tecator spectral data set and the UCI image segmentation data set.

3.1 Notes on Tecator data set

In an initial experiment the 100-dimensional Tecator meat spectral data set was used according to [11] focusing on high-fat vs. low fat content. Since another publication [13] indicated that Pearson correlation is good for classifying these data, each of the 215 spectra was mean-centered and scaled by its inverse standard variance, which in, combination with squared Euclidean distance, simulates Pearson correlation [10]. At a steepness of $\sigma = 100$, for the 100 random splits into 120 spectra for learning and 95 spectra for testing [11] average test errors of $2.00\% \pm 1.37$ were obtained by l-BFGS optimization for squared Euclidean distance with one prototype per class *without* relevance learning, while the functional SVM in [11] created an average error of 2.6%.
Because of inherent dependence of data splits, turning on relevance learning does not improve the error significantly and sometimes even creates worse test errors. Metric weight profiles learned by stochastic gradient descent and by l-BFGS are compared in Fig. 3. Therein, structurally similar though much more pronounced 'sparse' results are obtained from l-BFGS, which holds true in general, because stochastic gradient descent often cannot go far enough due to computing time limitations.
Since the Tecator data set is not complex enough for reliably highlighting differences between order of learning or choice of metrics, a more interesting data set is used in the next section for studying the GRLVQ behavior in more detail.

3.2 UCI segmentation data

The image segmentation data set [3] consists of 19-dimensional features collected from outdoor images for classification of the 7 classes brickface, sky, foliage, ce-

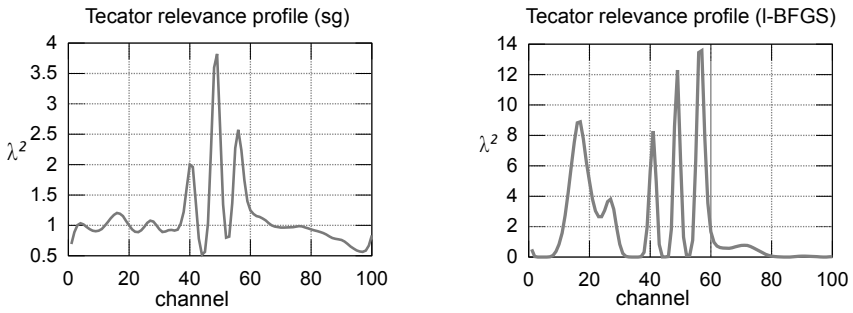


Figure 3: Euclidean relevance profiles for Tecator data set computed by stochastic gradient descent (left) and l-BFGS (right).

ment, window, path, and grass. The data set comes with a predefined split into 30 training samples and 300 test samples per class.

An initial study showed that even smallest GRLVQ networks with one prototype per class without relevance learning overfit the data if trained to final convergence. For example, training errors around 2.5% could be reached, while the test errors exceeded 7%. Due to the small amount of training data, no early stopping with validation data taken from the training could be reasonably used for avoiding overfitting. Instead, 100 stratified random data splits of all available data into 70% training and 30% test were used for cross-validation and for better reflecting conditions of a sufficient data set. Thus, in the following the focus is put on how on average the GRLVQ and optimization methods compare among each other rather than on comparison with results reported in literature.

Figure 4 summarizes the test classification results. The nine leftmost bars corresponding to LVQ results are of major interest, among which, the first three results with stochastic gradient were obtained by using the supervised relevance neural gas for general metrics (SRNGGM) package written in C publicly available from the web site `http://www.informatik.uni-osnabrueck.de/lnm/upload/` was run for 25000 epochs (=$25000 \cdot 0.7 \cdot (30+300) \cdot 7 = 40.4 \cdot 10^6$ iterations) in GRLVQ mode, that is, with a neural gas neighborhood of one and using learning rates of 0.1 for the prototypes and of 10^{-9} for the metric. For these three results, a dramatic drop can be observed for using the steep class discriminant function at $\sigma = 100$ instead of using standard GRLVQ settings with $\sigma = 1$ or $\mathrm{g}_\sigma = \mathsf{id}(x)$.

Moving on to the fourth column denoted 'ML best' this refers to the best result of steepest gradient with a linear change from $\sigma = 1$ to $\sigma = 100$, calculated in GNU-Octave, again using 25000 epochs. Although MATLAB takes about half of the time –that is, roughly 2 hours– the calculation is about 2 orders of magnitudes slower than the C-implementation, which inhibits in-depth cross validation here.

As shown for the GRLVQ (l-)BFGS columns run time can be dramatically decreased for MATLAB/GNU-Octave if second-order batch-optimization with full-fledged code vectorization is used. Here, two CPU threads where allowed to run in

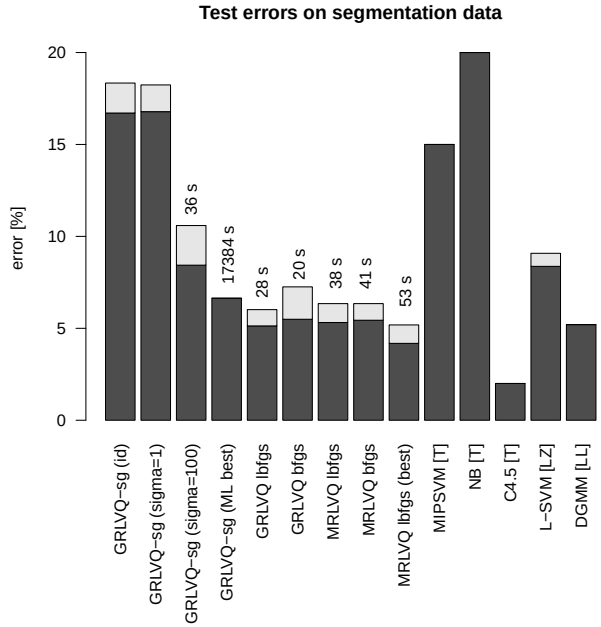


Figure 4: Segmentation classification results of GRLVQ and MRLVQ using different optimization routines and 100-fold stratified random cross validation at split of 7:3 for training vs. test samples. The '-sg' postfix denotes stochastic gradient descent. The five rightmost reference results for other methods from literature use different splits into training and test. [T]=[15], [LZ]=[12], and [LL]=[4]. If available light gray areas denote added standard deviations over the 100 runs. No run times for training were reported for the reference methods. The leftmost three GRLVQ results were obtained by a program written in C, all taking equal training times.

parallel. Even results are a bit better than for stochastic gradient, despite of fixing $\sigma = 100$ during whole optimization which induces difficult almost zero responses for many data points distant from the receptive field boundaries. The sequential optimization schedule for using (l-)BFGS is: $\langle 1 \rangle$ optimize only prototypes until rough convergence, $\langle 2 \rangle$ optimize only metric parameters until rough convergence, $\langle 3 \rangle$ optimize everything simultaneously.

For matrix learning MRLVQ similar results are obtained like for the Euclidean distance, if the rank of the rectangular matrix $\boldsymbol{\lambda}$ is set to $d = 5$. Lower ranks yield more than 2% worse results. Training times are a bit longer than GRLVQ, because of the parameter matrix being 5 times larger than Euclidean GRLVQ weights. Also the convergence benefit of full Hessian BFGS over memory-limited BFGS in GRLVQ is eaten up here by the larger number of metric parameters contributing in quadratic order to the full Hessian.

Finally, the best MRLVQ results are reported. In contrast to all the other LVQ variants, two prototypes per class are being optimized, and the rank of $\boldsymbol{\lambda}$ is set to $d = 7$. Since the corresponding training error were at about 2% in contrast to the roughly 5% test error a high level of overfitting is observed. Yet, the Euclidean GRLVQ did not significantly profit from using two prototypes per class, while this is the case for MRLVQ.

The five rightmost bars are related to results for the segmentation data set reported in literature. Notably, none of the papers explicitly states that the predefined split of 1:10 into training and test set was used in their experiments. 10-fold cross-validation is stated in [15], 20 randomizations are mentioned in [12], and either the predefined split or a 50:50 split was used in [4]. None of the methods reported time requirements.

4 Conclusions and Outlook

GRLVQ and MRLVQ have been revisited in the light of steep discriminant transfer functions g_σ and 2^{nd} order cost function optimization. Nothing relly new is added, except for the calculated gradients not being plugged into a stochastic gradient descent method but into an off-the-shelf batch optimization routine, (l-)BFGS, taking into account second order derivatives. Good classification improvements are gained by using l-BFGS in combination with steep sigmoid transfer functions ($\sigma \geq 10$).

The optimizer's standard settings can be used but the user is free to provide termination criteria, such as the minimum gradient change or cost-function changes, or the user can implement early stopping criteria at will. By experience, cost function formulations for unconstrained optimization methods work more satisfactorily than constrained optimization, which is possible for GRLVQ and MRLVQ because of their built-in metric weight squaring

that prevent negative or indefinite parameters. Thanks to batch processing, parallel optimization can be considered and is realized for MATLAB/GNU-Octave to some degree by the underlying matrix algebra libraries. Yet, efficient batch processing requires the data set to be completely stored in main memory.

By selecting very steep transfer functions and a normalized version of the underlying GLVQ cost function, cost function directly approximates misclassification rates. This is a very desirable criterion in many cases. Additionally, it allows to take direct influence on the class confusion matrix, and thus to control the rate of false positives or false negatives by adaptations towards weighted cost function formulations. Taking this way, classifiers can be formally derived for directly optimizing receiver operating characteristics and precision-recall properties.

The results reported for the segmentation data set are not well comparable to other result found in the literature. For better upcoming inter-method comparisons a classification task with consistently reported results on data preprocessing, classification capabilities and timings would be needed. For further experiments and general use of GRLVQ and MRLVQ the MATLAB/GNU-Octave source code 'GRLVQ' is available at `http://mloss.org`.

References

[1] Marika Kästner, Barbara Hammer, Michael Biehl, and Thomas Villmann. Generalized functional relevance learning vector quantization. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 93–98. D-side Publications, 2011.

[2] Thomas Villmann and Sven Haase. Divergence-based vector quantization. *Neural Computation*, 23:1343–1392, 2011.

[3] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[4] Xiao-Hua Liu and Cheng-Lin Liu. Discriminative training of subspace Gaussian mixture model for pattern classification. In D.S. Huang, Z. Zhao, V. Bevilacqua, and J.C. Figueroa, editors, *6th International Conference on Intelligent Computing*, volume 6215 of *Lecture Notes in Computer Science*, pages 213–221. Springer, 2010.

[5] Ernest Mwebaze, Petra Schneider, Frank Michael Schleif, Sven Haase, Thomas Villmann, and Michael Biehl. Divergence based learning vector quantization. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 247–252. D-side Publications, 2010.

[6] P. Schneider, K. Bunte, H. Stiekema, B. Hammer, T. Villmann, and M. Biehl. Regularization in matrix relevance learning. *IEEE Transactions on Neural Networks*, 21(5):831–840, 2010.

[7] Marc Strickert, Axel J. Soto, and Gustavo E. Vazquez. Adaptive matrix distances aiming at optimum regression subspaces. In Michel Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 93–98. D-facto Publications, 2010.

[8] Kerstin Bunte, Barbara Hammer, and Michael Biehl. Nonlinear dimension reduction and visualization of labeled data. In Xiaoyi Jiang and Nicolai Petkov, editors, *Computer Analysis of Images and Patterns*, volume 5702 of *Lecture Notes in Computer Science*, pages 1162–1170. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-03767-2_141.

[9] Petra Schneider, Michael Biehl, and Barbara Hammer. Distance learning in discriminative vector quantization. *Neural Computation*, 21(10):2942–2969, 2009.

[10] Marc Strickert, Frank-M. Schleif, Thomas Villmann, and Udo Seiffert. Similarity-based clustering - recent developments and biomedical applications. In M. Biehl, B. Hammer, M. Verleysen, and T. Villmann, editors, *Similarity-Based Clustering – Recent Developments and Biomedical Applications*, volume 5400 of *Lecture Notes in Computer Science*, pages 70–91. Springer, 2009.

[11] Fabrice Rossi and Nathalie Villa. Support vector machine for functional data classification. *Neurocomputing*, 69(7–9):730–742, March 2006.

[12] Zhizheng Liang and Tuo Zhao. Feature selection for linear support vector machines. In *18th International Conference on Pattern Recognition*, volume 2, pages 606–609, 2006.

[13] M. Strickert, U. Seiffert, N. Sreenivasulu, W. Weschke, T. Villmann, and B. Hammer. Generalized relevance LVQ (GRLVQ) with correlation measures for gene expression data. *Neurocomputing*, 69:651–659, 2006.

[14] Marc Strickert. *Self-Organizing Neural Networks for Sequence Processing*. PhD thesis, Institute of Computer Science, Universität Osnabrück, 2004.

[15] Amund Tveit. Empirical comparison of accuracy and performance for the MIPSVM classifier with existing classifiers. Technical report, IDI, NTNU, Trondheim, 2003.

[16] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15:1059–1068, 2002.

[17] A.S. Sato and K. Yamada. Generalized Learning Vector Quantization. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7 (NIPS)*, volume 7, pages 423–429. MIT Press, 1995.

[18] T. Kohonen. Learning Vector Quantization for Pattern Recognition. Report TKK-F-A601, Helsinki University of Technology, Espoo, Finland, 1986.

[19] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

Waypoint averaging and step size control in learning by gradient descent

***G. Papari*[1], *K. Bunte*[2], *M. Biehl*[2,3]**

Acknowledgements: This work was supported by the *Nederlandse Organisatie voor Weetenschappelijke Onderzoek* (NWO) under project code 612.066.620.

Abstract

We introduce a modification of batch gradient descent, which aims at better convergence properties and more robust minimization. In the course of the descent, the procedure compares the performance of the actual configuration with that of a gliding average over the most recent positions. If the latter corresponds to a lower value of the optimization objective, minimization proceeds from there and the step size of the descent is decreased.
Here we present the prescription from a practitioner's point of view and refrain from a detailed mathematical analysis. First, the method is illustrated in terms of a low dimensional example. Moreover, we discuss its application in the context of machine learning, examples corresponding to multilayered neural networks and a recent extension of Learning Vector Quantization (LVQ) termed Matrix Relevance LVQ.

1 Introduction

Gradient based minimization is one of the most popular, basic techniques in non-linear optimization [18]. While many, more sophisticated methods are also gradient based, plain gradient descent faces a number of significant problems. First of all, the success of steepest descent depends crucially on the choice of an appropriate magnitude of the

[1]National Institute of Research in Informatics and Automatics (INRIA)
Department CLIME, BP 105, 78153 Le Chesnay Cedex, France
[2]Johann Bernoulli Institute for Mathematics and Computer Science
University of Groningen, P.O. Box 407, 9700 AK Groningen, The Netherlands
[3]E-mail: m.biehl@rug.nl

update step. Too careful updates will cause slow convergence, while large steps may result in oscillatory or even divergent behavior.

Methods for automated step size control and so-called *line-search* procedures have been designed which can overcome this difficulty to a large extent. Similarly, in the well-known *conjugate gradient descent* a coefficient is determined which controls the superposition of two orthogonal descent steps [18]. Higher order methods which employ second or further derivatives include, among others, Newton and Quasi-Newton methods. Arguably, the latter play the most important role in practical optimization of non-convex non-linear cost functions, nowadays. Frequently, these methods are computationally expensive and difficult to implement in high-dimensional search spaces. In addition, they often require the tuning of algorithm parameters which further complicate their use in practice.

In particular in the specific context of machine learning, plain gradient descent has played a key role and continues to do so for several reasons. Gradient descent gained significant importance when multi-layered neural networks were introduced and studied, initially. The availability of simple and efficient implementations of gradient descent, e.g. the well-known *backpropagation of error* [19, 12, 17, 13, 5], contributed immensely to the popularity of neural networks and machine learning in general.

To date, gradient descent is a popular tool in many machine learning tasks that can be formulated in terms of, frequently non-convex, non-linear optimization problems. Due to its simplicity and flexibility, gradient descent is often the first choice in initial investigations of novel learning paradigms. It has been employed in, both, supervised and unsupervised learning. Examples for the former comprise the already mentioned training of multi-layered neural networks by means of backpropagation and, more recently, prototype-based Learning Vector Quantization and variants [11]. Competitive learning in Vector Quantization [17] and cost function based variants of Neural Gas [15, 3], constitute important examples for the application of gradient descent in unsupervised learning.

In the machine learning domain, most frequently, the cost function and, thus, its gradient can be written as a sum over the available example data. This facilitates the use of a particularly simple and efficient scheme termed *stochastic gradient descent*, which is also known as the *Robbins Monro* procedure [20, 13] in a more general context. Here, the actual gradient is approximated by the contribution of a single training example. The noise introduced by its random selection is believed to be beneficial, for instance with respect to escaping local minima. For a discussion of various training prescriptions which are based on the stochastic approximation of gradients, see [7].

On the other hand, *batch* gradient procedures make use of all examples in every iteration, which increases the computational effort per step, but may be advantageous in terms of efficiency.

Generic problems of gradient descent are also present, and sometimes particularly pronounced, in both variants of gradient descent training. While the effect of local minima on the actual performance of the resulting system is not always clear, their presence certainly complicates the training process. Local minima result in, for instance, high

sensitivity to initial conditions of the training process.

Flat regions in the search space, where the gradient of the cost function displays low magnitude can also constitute a problem in practice. They can result in so-called quasi-stationary *plateau states* which can drastically slow down the learning process and, frequently, dominate the shape of learning curves in gradient based training. For a mathematical analysis of this phenomenon, borrowing concepts from statistical physics, see for instance [10, 14, 6, 9, 4].

As a consequence, many modifications of plain gradient descent have been introduced and investigated within the machine learning community. The choice of appropriate learning rates and learning rate schedules plays a key role, obviously. For stochastic gradient descent, for instance, exact criteria are known for schedules which realize convergence to a (local) minimum. In practice, one has to compromise between the desired approach to a potentially global minimum on the one hand and constraints on the tolerated computational effort on the other.

The problem of flat regions of the cost function, in which steepest descent without normalization of the gradient is slow, has attracted considerable interest in the machine learning community. One the most popular extensions of gradient based training introduces a memory term which is supposed to facilitate persistent moves along previously found directions of descent. The term *momentum* has been coined for this popular concept [19, 12, 17]. Other modifications of gradient descent concern the design of so-called *well-behaved cost functions* which modify the original objective, aiming at fast initial training and better convergence properties, see for instance [17].

The use of higher order methods has been explored also in the context of machine learning. For reviews, concrete examples, and further references we suggest to consult, for instance, [7, 17, 12]. Obviously, the evaluation or estimation of higher order derivatives poses a practical problem in high-dimensional spaces and limits the usefulness of the approach in many learning problems.

In the context of stochastic gradient descent, an averaging procedure has been suggested which does not modify the descent itself, but interprets the mean over all performed descent steps as the actual outcome of training [16, 1]. Obviously, this will reduce the influence of random fluctuations while keeping the presumed advantages of stochastic descent. Indeed, the approach has been shown to yield favorable convergence properties in [1].

In the following we suggest an approach which combines the basic idea of *waypoint averages*, here over a limited history, with an appropriate step size adaption. It provides a conceptually simple and computationally efficient extension of steepest descent. It is easy to implement and bears the promise to yield robust performance in, for instance, practical learning problems or more general optimization tasks.

In this report we focus on a heuristic motivation and present the algorithm from a practitioner's point of view, with particular emphasis on machine learning applications. More mathematical aspects of the method will be presented elsewhere. We illustrate the approach in terms of low-dimensional optimization problems as well as an example machine learning problem.

2 The Algorithm

First we consider the case of an objective function E which depends on a d-dim. vector $\mathbf{x} \in I\!R^d$. A gradient descent procedure, initialized in $\mathbf{x}_o$, generates a sequence of positions $\mathbf{x}_t$ by an iteration of the form

$$\mathbf{x}_{t+1} \;=\; \mathbf{x}_t \;-\; \alpha_t \, \frac{\nabla E_t}{|\nabla E_t|}. \tag{1}$$

Here and in the following we use the shorthands $\nabla E_t \;=\; \nabla E|_{\mathbf{x}=\mathbf{x_t}}$. Note that the gradient is normalized in Eq. (1). Hence, α_t controls explicitly the step length in terms of Euclidean distance in $I\!R^d$: $|\mathbf{x}_{t+1} - \mathbf{x}_t| = \alpha_t$. Accordingly, we will refer to α_t as the *step size* at iteration step t. The related quantity $\eta_t \;=\; \alpha_t / \left|\nabla E_t\right|$ corresponds to the *learning rate* in standard machine learning jargon, i.e. a pre-factor of the unnormalized gradient.

In order to ensure convergence one has to set the learning rate or step size, respectively, small enough. For constant $\eta_t = \eta$ it is straightforward to work conditions for the convergence of gradient descent close to a (local) minimum $\mathbf{x}^*$. Let us assume that we can expand E as

$$E(\mathbf{x}) \;\approx\; E(\mathbf{x}^*) \;+ \frac{1}{2} \; (\mathbf{x} - \mathbf{x}^*)^\top \; H^* \; (\mathbf{x} - \mathbf{x}^*) \tag{2}$$

where the elements of the Hesse-Matrix H^* are given by $H^*_{ij} \;=\; \left. \frac{\partial^2 E}{\partial x_i \, \partial x_j} \right|_{\mathbf{x}=\mathbf{x}^*}$.

The largest eigenvalue λ_{max} of H^* corresponds to the largest curvature observed in $\mathbf{x}^*$. One can show that for $\eta < 2/\lambda_{max}$ the deviation $|\mathbf{x}_t - \mathbf{x}_*|$ vanishes as $t \to \infty$. However, in practical situations, the properties of the unknown minimum are not known and H^* itself is not available. A variety of schemes exist, which resort to the evaluation of the local Hesse matrix H for automatic step size adaptation in machine learning, see [7] for further references. More frequently, simple heuristic *annealing schemes* are used which reduce η_t explicitly with time, see [20, 13, 17, 7] for a discussion and examples. Note that these schemes inevitably introduce a number of algorithm parameters which have to be fine-tuned to the concrete practical learning problem at hand.

Here we present a simple and robust extension of gradient descent which improves convergence by considering *waypoint averages* over the latest iteration steps and implements an efficient step size adaptation at the same time. It does not require the costly evaluation of higher order derivatives and the number of additional control parameters is very small compared to some of the other approaches mentioned above.

Waypoint averaging and step size adaptation
The iteration is initialized in $\mathbf{x}_o$ and the initial step size is α_o. First, a number k of

unmodified gradient steps is performed, i.e.

$$\mathbf{x}_{j+1} \;=\; \mathbf{x}_j \;-\; \alpha_j \,\frac{\nabla E_j}{|\nabla E_j|} \quad \text{for } \; j = 0, 1, 2, \ldots, k-1 \quad \text{with} \quad \alpha_j = \alpha_o. \tag{3}$$

Thereafter, the iteration proceeds as described in the following:

1. evaluate the *tentative* gradient step

$$\widetilde{\mathbf{x}}_{t+1} \;=\; \mathbf{x}_t \;-\; \alpha_t \,\frac{\nabla E_j}{|\nabla E_j|} \;\text{ and }\; E(\widetilde{\mathbf{x}}_{t+1}) \tag{4}$$

2. calculate the *waypoint average* over the previous k steps:

$$\widehat{\mathbf{x}}_{t+1} \;=\; \frac{1}{k}\sum_{i=0}^{k-1} \mathbf{x_{t-i}} \;\text{ and }\; E(\widehat{\mathbf{x}}_t) \tag{5}$$

3. determine new position and new step size as

$$\begin{cases} \mathbf{x}_{t+1} = \widetilde{\mathbf{x}}_{t+1} \;\text{ and }\; \alpha_{t+1} = \alpha_t & \text{if } \; E(\widetilde{\mathbf{x}}_{t+1}) \le E(\widehat{x}_{t+1}) \\ \mathbf{x}_{t+1} = \widehat{\mathbf{x}}_{t+1} \;\text{ and }\; \alpha_{t+1} = r \cdot \alpha_t & \text{else.} \end{cases} \tag{6}$$

with the parameter $r < 1$.

As long as the plain gradient descent step yields a position which corresponds to lower costs than the *waypoint average* $\widehat{\mathbf{x}}_{t+1}$ over the last k steps, the iteration proceeds unmodified.

On the contrary, $E(\widehat{\mathbf{x}}_{t+1}) < E(\widetilde{\mathbf{x}}_{t+1})$ signals that the procedure has *overshot* and displayed oscillatory behavior because the step size has been *too large* for smooth convergence. As a consequence, one may expect that the positions $\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-k+1}$ fluctuate about a local minimum and the *waypoint average* should provide a better estimate than the tentative $\widetilde{\mathbf{x}}_{t+1}$. In this case, the iteration proceeds from $\widehat{\mathbf{x}}_{t+1}$ and the step size is reduced by a factor $r < 1$.

In a forthcoming publication we will discuss favorable settings of the parameter r. In addition, several extensions and modifications of the basic prescription are possible. For instance, an additional parameter $q > 1$ could be introduced to increase the step size as $\alpha_{t+1} = q \cdot \alpha_t$ whenever the *tentative* step is accepted, thus avoiding slow convergence due to inappropriately small step sizes. Here we restrict the discussion to the case $q = 1$ and refer to forthcoming studies for the discussion of the extension.

Figure 5 shows a simple example in $d = 2$ dimensions and illustrates the method by comparing updates with constant step size and the procedure with waypoint averaging and step size control.

3 A machine learning example

Frequently, subsets of variables can be identified which play qualitatively different roles in the optimization problem with significantly different gradient magnitudes and curvatures of E. In the suggested descent procedure, meaningful groups of variables can

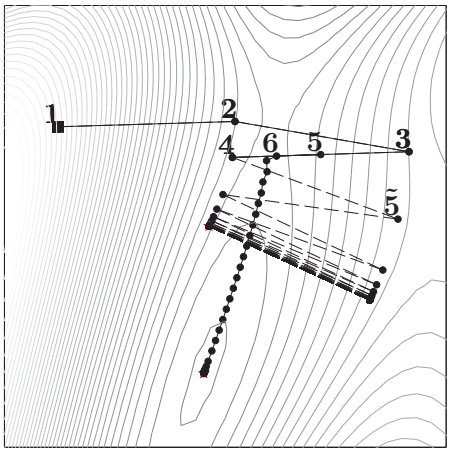


Figure 5: A simple optimization in $d = 2$ dimensions. Symbols and connecting lines mark the trajectories of two gradient based iterations, we display 500 steps for each procedure. Both trajectories start from the same initial position, marked as "1", and employ the same initial step size α_o. Unmodified gradient descent according to Eq. (1) with constant α displays strongly oscillating behavior. The modification with waypoint averaging (here: $k = 2$) and step size adaptation is identical up to step 4, but then replaces the tentative position $\widetilde{5}$ by the mean $(x_3 + x_4)/2$. The step size is then reduced by a factor $r = 1/4$. Also position 6 results from an average over x_4 and x_5, which is very close to the tentative $\widetilde{x}_6$ (not shown). Subsequently the iteration approaches the minimum with step size $\alpha_o/16$ for a number of steps. Close to the minimum many waypoint averages are performed and the step size decreases very rapidly.

be taken into account by normalizing the partial gradients separately and assigning different step sizes to them. In the context of machine learning such subsets could be, for instance, first and second layer weights in a layered neural network. Another example are prototype vectors and relevance matrices in Matrix Relevance LVQ [2]. We employ the latter framework to illustrate an appropriate modification of our method.

As an example data set we consider the *Segmentation* data set as provided by the UCI repository of Machine Learning [8]. The data set contains $(d = 18)$-dim. feature vectors $\mathbf{x}_i$ which are assigned to one of 7 classes denoted by $c(\mathbf{x}_i) \in \{1, 2, \ldots, 7\}$. Note that one of the nominally 19 features does not vary at all and has been omitted here. The training set contains 210 samples (30 per class), 2100 data points (300 per class) serve as a test set. For a more detailed description of the data consult [8] or, for instance, [2].

We consider the simplest setting of GMLVQ with one prototype representing each class. We denote by $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_7]$ a $(7 \cdot 18)$-dim. vector which contains the concatenated prototypes. Classification is parameterized in terms of a nearest prototype

scheme which employs the generalized distance measure

$$d(\mathbf{w_k}, \mathbf{x}) \;=\; (\mathbf{w_k} - \mathbf{x})^{\top}\; \Omega^{\top}\Omega\; (\mathbf{w_k} - \mathbf{x})\,. \tag{7}$$

Here, $\mathbf{x} \in I\!R^d$ represents a feature vector, $\mathbf{w}_k$ is one of the prototypes, and $\Omega \in I\!R^{d\times d}$ is a matrix of adaptive parameters which define the measure.

The training process is guided by the cost function

$$E = \sum_{i=1}^{210} \frac{d(\mathbf{w}_J, \mathbf{x}_i) - d(\mathbf{w}_K, \mathbf{x}_i)}{d(\mathbf{w}_J, \mathbf{x}_i) + d(\mathbf{w}_K, \mathbf{x}_i)} \tag{8}$$

where the sum is over the training examples and the vector $\mathbf{w}_J$ is the prototype representing the class $c(\mathbf{x_i})$. The vector $\mathbf{w}_K$ is the closest prototype representing one of the other classes, as determined according to the distance measure (7).

In GMLVQ the cost function is to be optimized with respect to, both, the prototype positions and the matrix Ω. When applying stochastic gradient descent, it has proven useful to update the elements of Ω with a learning rate different from that for the prototype components [2]. This reflects the fact that the dependence of E on the $\mathbf{w}_j$ and the matrix Ω is expected to be qualitatively different.

In batch descent based on normalized gradients, Eq. (1), we can take this idea into account by performing the normalization for the matrix Ω and the concatenated prototype vector $\mathbf{W}$ separately and using different step sizes in the tentative gradient update corresponding to Eq. (4):

$$\widetilde{\mathbf{W}}_{t+1} \;=\; \mathbf{W}_t \;-\alpha_t^{(W)}\; \frac{\partial E/\partial \mathbf{W}}{|\partial E/\partial \mathbf{W}|} \tag{9}$$

$$\widetilde{\Omega}_{t+1} \;=\; \Omega_t \;-\alpha_t^{(\Omega)}\; \frac{\partial E/\partial \Omega}{|\partial E/\partial \Omega|}. \tag{10}$$

Here we refrain from providing the gradient terms explicitly and refer the reader to [2] for details.

In complete analogy to the above described basic formulation, cf. Eq. 6), the cost function $E(\widetilde{\mathbf{W}}_{t+1}, \widetilde{\Omega}_{t+1})$ is compared with the corresponding costs achieved by

$$\widehat{\mathbf{W}}_{t+1} = \sum_{i=0}^{k-1} \mathbf{W}_{t-i} \;\text{ and }\; \widehat{\Omega}_{t+1} = \sum_{i=0}^{k-1} \Omega_{t-i}.$$

In case the latter is lower, the waypoint average is accepted as the new position and both step sizes are reduced:

$$\alpha_{t+1}^{(W)} = r \cdot \alpha_t^{(W)} \;\text{ and }\; \alpha_{t+1}^{(\Omega)} = r \cdot \alpha_t^{(\Omega)}. \tag{11}$$

As the free parameters of the prescription, one has to set the initial values $\alpha_o^{(W)}$ and $\alpha_o^{(\Omega)}$. Note that their ratio remains fixed in the course of the iteration.

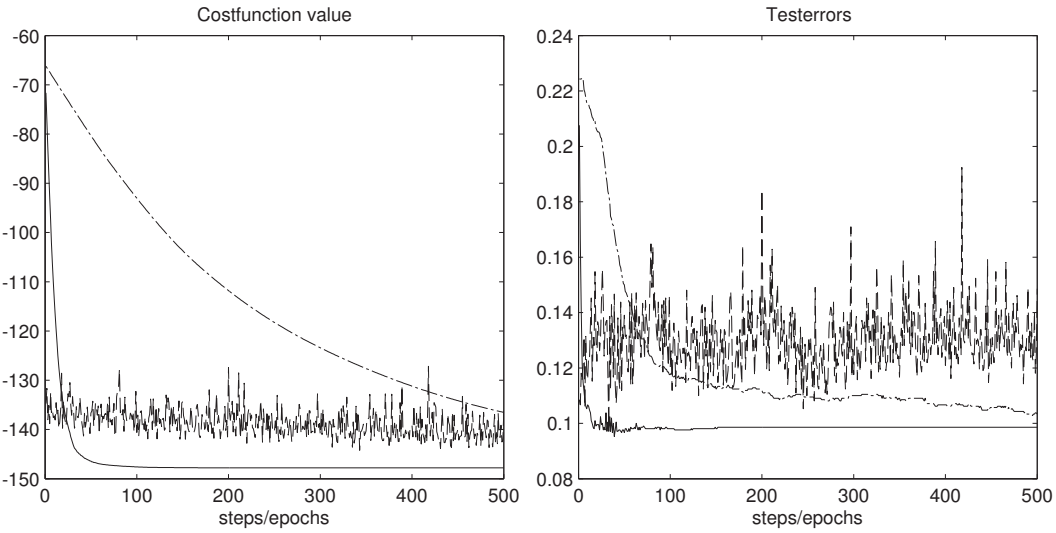


Figure 6: UCI segmentation data set: GMLVQ learning curves for batch gradient descent with constant step sizes (dash–dotted), stochastic gradient descent (dashed), and waypoint averaging with step size adaptation (solid lines). The left panel displays the evolution of the GMLVQ cost function vs. the number of steps (batch methods) or epochs (stochastic descent), respectively. The right panel shows the total classification error with respect to the test set. Parameter settings of the algorithms are specified in the text.

Figure 6 displays learning curves for different variants of gradient based GMLVQ training applied to the UCI segmentation data. We observe only a very weak dependence of the performance on the initial step sizes and on their ratio $\alpha_o^{(\Omega)}/\alpha_o^{(W)}$. The example shown corresponds to $\alpha_o^{(W)} = 1/18$ and $\alpha_o^{(\Omega)} = \alpha_o^{(W)}/2$; the other parameters were $k = 3$ and $r = 2/3$.

For comparison we display example curves for batch gradient descent with constant step sizes $\alpha_o^{(W)} = 1/180$ and $\alpha_o^{(\Omega)} = \alpha_o^{(W)}/5$. These values were chosen such that the outcome after 500 steps is comparable to that of the waypoint averaging procedure with adaptive step size.

Furthermore, display the results of stochastic gradient descent with learning rate schedules of the form

$$\eta(t) = a_1 \, \exp\left[-\ln\left(\frac{a_1}{a_2}\right) \, \frac{t}{t_{max}} \right]$$

where $t_{max=500}$ specifies the maximum number of epochs in the training process. Note that one epoch presents all training examples once and, hence, is to be com-

pared with one step of batch descent. The curves displayed were obtained for $a_1^{(W)} = 0.05, a_2^{(W)} = 0.001$ for the prototype vectors and $a_1^{(\Omega)} = 0.01, a_2^{(\Omega)} = 0.001$ for matrix updates. Note that the stochastic descent displays strong fluctuations which need to be controlled by proper annealing of the learning rate.

Clearly, the performance could be further optimized by choice of the constant step sizes in batch training or the learning rate schedules in the stochastic gradient descent. Potentially, a performance very similar to that of the waypoint averaging procedure could be achieved. However, the important point is that the latter yields very good optimization and classification performance without careful tuning of a number of parameters.

4 Summary and Conclusion

In this Technical Report we present a modification of gradient based optimization which constitutes a conceptually simple extension of steepest descent. The main ingredient is the consideration of *waypoint averages* over the most recent iteration steps in combination with an adaptive step size control. Here we merely present and illustrate the basic concept of the method. We discuss its application in the context of an example machine learning problem: gradient based Matrix Relevance LVQ.

The simple examples considered here already illustrate some of the most attractive features of the method. First of all, it is easy to implement, computationally cheap, and – in contrast to many other schemes – does not require the careful tuning of a large number of algorithm parameters. In particular, the combination of waypoint averages and step size control makes it unnecessary to define explicit learning rate schedules or to use higher derivatives for learning rate adaptation. The use of normalized gradients may appear merely technical at first sight. However, compared to standard steepest descent based on unnormalized gradients, it helps to overcome plateau states and flat regions of the cost function very efficiently.

In a forthcoming publication we will address these aspects in greater depth and demonstrate the flexibility and robustness of the approach in terms of various example problems. A more systematic comparison with alternative, popular methods will also be presented. In addition we will study further mathematical aspects of the approach, including, for instance, the optimal choice of parameters r and k. We will, furthermore, demonstrate that the method is suitable also in situations in which the cost function is not differentiable in the minimum.

A number of question deserves particular attention in the context of machine learning, e.g. the convergence behavior in the presence of extended plateaus or many local minima.

References

[1] Wei Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *CoRR*, abs/1107.2490, 2011.

[2] P. Schneider, M. Biehl, and B. Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561, 2009.

[3] T. Villmann, B. Hammer, and M. Biehl. Some theoretical aspects of the neural gas vector quantizer. In M. Biehl, B. Hammer, M. Verleysen, and T. Villmann, editors, *Similarity-Based Clustering*, volume 5400, pages 23–34. Springer Lecture Notes in Computer Science, 2009.

[4] A. Witoelar, M. Biehl, A. Ghosh, and B. Hammer. Learning dynamics and robustness of vector quantization and neural gas. *Neurocomputing*, 71(7-9):1210–1219, 2008.

[5] M.A. Arbib, editor. *The handbook of brain theory and neural networks*. MIT Press, Cambridge,MA, 2003.

[6] M. Biehl and N. Caticha. The statistical mechanics of on-line learning and generalization. *The handbook of brain theory and neural networks*, pages 1095–1098, 2003.

[7] D. Saad, editor. *Online learning in neural networks*. Cambridge University Press, Cambridge, UK, 1999.

[8] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases. `http://archive.ics.uci.edu/ml/`, 1998.

[9] M. Biehl, A. Freking, and G. Reents. Dynamics of on-line competitive learning. *Europhysics Letters*, 38:73–78, 1997.

[10] M. Biehl, P. Riegler, and C. Wöhler. Transient dynamics of on-line learning in two-layered neural networks. *Journal of Physics A: Mathematical and General*, 29:4769–4780, 1996.

[11] A. Sato and K. Yamada. Generalized learning vector quantization. In M. C. Mozer D. S. Touretzky and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*, pages 423–9, Cambridge, MA, USA, 1996. MIT Press.

[12] Y. Chauvin and D.E. Rumelhart, editors. *Backpropagation: Theory, Architectures, and Applications*. Hillsdale, Erlbaum, 1995.

[13] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1 edition, 1995.

[14] D. Saad and S.A. Solla. Exact Solution for Online Learning in Multilayer Neural Networks. *Phys. Rev. Lett.*, 74:4337–4340, 1995.

[15] T. Martinetz, S. Berkovich, and K. Schulten. Neural gas network for vector quantization and its application to time series prediction. *IEEE Trans. Neural Networks*, 4:558–569, 1993.

[16] B.T. Polyak and A.B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control and Optimization*, 30:838–855, 1992.

[17] J.A. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison Wesley, Redwood City, 1991.

[18] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, 2nd edition, 1987.

[19] D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing*, volume 1. Bradford Books, Cambridge and London, 1987.

[20] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22:400–407, 1951.

Theory of Fuzzy Neural Gas for Unsupervised Vector Quantization

***T. Villmann*[1,2]*,T. Geweniger*[2]*,M. Kästner*[2]*,M. Lange*[2]

Abstract

In this paper we propose a new approach for fuzzy clustering based on fuzzy-c-means incorporating neighborhood cooperativeness according to the neural gas vector quantizer. This approach offers greater flexibility than the already known combination of fuzzy-c-means with self-organizing maps. We give in this article the theoretical justification of the new fuzzy neural gas algorithm. Further, we focus on the explicit control of the sparseness of the fuzzy assignments extending the cost function of fuzzy neural gas by an entropic penalty term.

1 Introduction

Clustering of data is a challenging task. An important class of clustering algorithms is prototype based vector quantization where prototype vectors represent the data. One can distinguish crisp (hard) clustering approaches where data are uniquely assigned to the representing prototypes and so-called fuzzy approaches where probabilistic or possibilistic assignments describe the representation of data by the prototypes. The most famous algorithm is the fuzzy-c-means algorithm (FCM,[35, 39]).

Beside this differentiation of vector quantization algorithms, another differentiation of prototype based vector quantization algorithms can be accomplished according to their learning strategies. Among others, neighborhood cooperativeness between prototypes during learning is a learning paradigm known from neural maps, which generally improves vector quantization performance and convergence speed as well as stability of the vector quantization solution.

[1]E-mail: thomas.villmann@hs-mittweida.de

[2]Computational Intelligence Group, University of Applied Sciences Mittweida, Technikumplatz 17, 09648 Mittweida, Germany

Soft-topographic vector quantization (STVQ,[17]) as well as fuzzy self-organizing map (FSOM,[21, 29, 26]) are approaches, which combine both strategies - fuzziness and neighborhood cooperation. Thereby, the neighborhood cooperativeness is cooped from the self-organizing map model (SOM,[24]) assuming an external grid structure between the prototypes, usually a regular hypercubical structure. Otherwise, for crisp vector quantization, the neural gas algorithm (NG) using a dynamic prototype based neighborhood generally shows better performance [28].

In this paper we propose a new fuzzy vector quantization scheme based on the dynamic neighborhood cooperativeness known from neural gas instead of the SOM-based neighborhood cooperativeness in STVQ resulting in the fuzzy neural gas (FNG) algorithm. Further, we investigate entropy based sparsity control for the fuzzy assignments which keeps the regularization restriction from FCM.

2 Fuzzy-Probabilistic, Fuzzy-Possibilistic and Soft Vector Quantization

In this section we briefly introduce two of the basic principles of fuzzy or soft vector quantization. The first is the classic fuzzy-c-means (FCM) as the basic fuzzy vector quantization scheme based on an expectation-maximization (EM) principle or the alternating optimization scheme. The other one is the soft-topographic vector quantization scheme (STVQ), which incorporates neighborhood cooperativeness for soft vector quantization.

In the following we assume a data set $V = \{\mathbf{v}_i\}_{i=1}^{N} \subseteq \mathbb{R}^n$ and a set $W = \{\mathbf{w}_k\}_{k=1}^{C} \subset \mathbb{R}^n$ of prototypes. Further, we suppose an inner product norm $d_{i,k} = d(\mathbf{v}_i, \mathbf{w}_k)$ between data and prototypes, frequently the Euclidean distance.

2.1 The Fuzzy-c-Means Algorithm for Fuzzy-Probabilistic Clustering

The Fuzzy-c-Means algorithm is one of the most prominent fuzzy clustering algorithms [35, 39]. Many variants are proposed such as for relational data [32] or median clustering [3] or using several kinds of dissimilarities like divergences [9, 1] or kernels [14].

The original FCM model determines for each data point $\mathbf{v}_i \in V$ and prototype $\mathbf{w}_k \in W$ an assignment $u_{i,k} \in [0, 1]$, which is interpreted as the possibility that this data vector is associated with this particular prototype. If the assignments are restricted to fulfill the constraints

$$\sum_k u_{i,k} = 1 \tag{1}$$

the model is called probabilistic. The crisp c-means model is obtained for the probabilistic model with the additional condition that $u_{i,k} \in \{0, 1\}$ [38, 37].

The FCM minimizes the cost function

$$E_{FCM}\left(\mathbf{U}, V, W\right) = \sum_{k=1}^{C}\sum_{i=1}^{N}\left(u_{i,k}\right)^{m}\left(d_{i,k}\right)^{2} \tag{2}$$

where $m \geq 1$ is the fuzziness parameter usually chosen as $m = 2$ [35, 39]. The iterated optimization scheme (expectation-maximization-approach) consists of a M-step

$$\mathbf{w}_k = \frac{\sum_{i=1}^{N}\left(u_{i,k}\right)^{m}\mathbf{v}_i}{\sum_{i=1}^{N}\left(u_{i,k}\right)^{m}} \tag{3}$$

together with the accompanying E-step

$$u_{i,k} = \frac{1}{\sum_{l=1}^{C}\left(\frac{d_{i,k}}{d_{i,l}}\right)^{\frac{2}{m-1}}} \tag{4}$$

derived as the solution of the Lagrange-minimization problem

$$J\left(\mathbf{U}, V, W, \lambda\right) = E_{FCM}\left(\mathbf{U}, V, W\right) - \sum_{i=1}^{N}\left(\lambda_i\left(\sum_{k=1}^{C}u_{i,k} - 1\right)\right) \tag{5}$$

with Lagrange multipliers $\lambda = (\lambda_1, \ldots, \lambda_N)$.

2.2 Fuzzy-Possibilistic Clustering based on Fuzzy c-means

Possibilistic c-means (PCM, [27]) differs from FCM in such a way that the assignment constraints (1) of the FCM are not longer valid. Thus the only remaining conditions are $u_{i,k} \in [0, 1]$. These assignments are called typicality assignments $t_{i,k} \in [0, 1]$ in PCM. The different definition causes a slightly modified cost function, which now reads as

$$E_{PCM}\left(\mathbf{U}, V, W, \delta\right) = \sum_{k=1}^{C}\sum_{i=1}^{N}\left(t_{i,k}\right)^{\eta}\left(d_{i,k}\right)^{2} + \sum_{k=1}^{C}\left(\delta_k\sum_{i=1}^{N}\left(t_{i,k} - 1\right)^{\eta}\right) \tag{6}$$

with user defined constants $\delta_i > 0$. Here, the M-step

$$\mathbf{w}_k = \frac{\sum_{i=1}^{N}\left(t_{i,k}\right)^{\eta}\mathbf{v}_i}{\sum_{i=1}^{N}\left(t_{i,k}\right)^{\eta}} \tag{7}$$

is equivalent to that of the FCM model, but the E-step for the typicality values $t_{i,k}$ is modified to

$$t_{i,k} = \frac{1}{1 + \left(\frac{\left(d_{i,k}\right)^2}{\delta_k}\right)^{\frac{1}{\eta-1}}} \tag{8}$$

taking the δ_i-values into account. It is recommended to initialize the PCM by FCM [20]. A suggested choice for the δ_i-values is

$$\delta_k = K \frac{\sum_{i=1}^{N} (u_{i,k})^{\eta} (d_{i,k})^2}{\sum_{i=1}^{N} (u_{i,k})^{\eta}} \tag{9}$$

with the $u_{i,k}$ obtained from FCM and $\eta = m$ [20].

Another probabilistic fuzzy clustering is Fuzzy PCM (FPCM), which tries to avoid the problem of coinciding clusters (prototypes) in FCM and PCM [13]. It is a merge of FCM and PCM such that the new cost function is a linear combination of both approaches:

$$E_{FPCM}(\mathbf{U}, V, W, \delta) = \sum_{k=1}^{C} \sum_{i=1}^{N} \left(u_{i,k}^{m} + t_{i,k}^{\eta} \right) (d_{i,k})^2 \tag{10}$$

with the constraints $\sum_k u_{i,k} = 1$ from FCM and $\sum_i t_{i,k} = 1$ reflecting the cluster typicality to be probabilistic. The term $\sum_{k=1}^{C} \sum_{i=1}^{N} (t_{i,k})^{\eta} (d_{i,k})^2$ is reported to be responsible for broader distribution of the $t_{i,k}$ to all data points but not with respect to all clusters. The resulting M-step

$$\mathbf{w}_k = \frac{\sum_{i=1}^{N} \left(u_{i,k}^{m} + t_{i,k}^{\eta} \right) \mathbf{v}_i}{\sum_{i=1}^{N} \left(u_{i,k}^{m} + t_{i,k}^{\eta} \right)} \tag{11}$$

is accompanied by the E-steps: the usual E-step for $u_{i,k}$ of FCM (1) and, analogously,

$$t_{i,k} = \frac{1}{\sum_{j=1}^{N} \left(\frac{d_{i,k}}{d_{j,k}} \right)^{\frac{2}{\eta - 1}}} \tag{12}$$

with summation over all data. However, in that case, we have the constraint $\sum_i t_{i,k} = 1$, which is unsatisfying compared to the idea of PCM.

Adopting the idea from PCM also for FPCM the most general fuzzy clustering scheme is

$$E_{PFCM}(\mathbf{U}, V, W, \delta) = \sum_{k=1}^{C} \sum_{i=1}^{N} \left(a \cdot u_{i,k}^{m} + b \cdot t_{i,k}^{\eta} \right) (d_{i,k})^2 + \sum_{k=1}^{C} \left(\delta_k \sum_{i=1}^{N} (t_{i,k} - 1)^{\eta} \right) \tag{13}$$

with constants $a > 0$ and $b > 0$ balancing both models [13]. This model consists of the M-step

$$\mathbf{w}_k = \frac{\sum_{i=1}^{N} \left(a \cdot u_{i,k}^{m} + b \cdot t_{i,k}^{\eta} \right) \mathbf{v}_i}{\sum_{i=1}^{N} \left(a \cdot u_{i,k}^{m} + b \cdot t_{i,k}^{\eta} \right)} \tag{14}$$

and the E-steps for $u_{i,k}$ of FCM (1) and $t_{i,k}$ from the FPCM-model (12).

2.3 Soft Topographic Vector Quantization

The STVQ model uses probabilistic soft assignments $p(k|\mathbf{v}_i)$ of data vectors $\mathbf{v}_i$ to prototypes $\mathbf{w}_k$, i.e. $\sum_{k=1}^{C} p(k|\mathbf{v}_i) = 1$. Furthermore, an external topological structure A between the prototypes is assumed defining a dissimilarity $d_A(k,l)$ between the prototypes $\mathbf{w}_k$ and $\mathbf{w}_l$ with respect to the structure A. Usually, the structure A is assumed to be a regular hypercubical lattice as known from SOMs, and d_A is taken as the Euclidean distance in A for that case. Further, a neighborhood cooperativeness between the prototypes is installed using a neighborhood function defined on A in complete analogy to SOMs:

$$h_\sigma^{SOM}(k,l) = c_\sigma \cdot \exp\left(-\frac{(d_A(k,l))^2}{2\sigma^2}\right) \tag{15}$$

with neighborhood range σ and the constraint $\sum_l h_\sigma^{SOM}(k,l) = 1$ ensured by the constant c_σ. Thus the neighborhood range induces a range of interaction implicitly also in the data space dteremined by the location of the prototypes.

For this setting, topographic vector quantization (TVQ) is defined by minimization of the cost function

$$E_{TVQ}(W,\sigma) = \sum_{i=1}^{N}\sum_{k=1}^{C} p(k|\mathbf{v}_i) \cdot lc_\sigma(i,k) \tag{16}$$

with local costs

$$lc_\sigma^{SOM}(i,k) = \sum_{l=1}^{C} h_\sigma^{SOM}(k,l) \cdot \left(d_{i,l}^E\right)^2 \tag{17}$$

where $d_{i,k}^E$ is the Euclidean distance [25]. However, other dissimilarity measures $d_{i,k}$ are possible.

Minimization of the cost function E_{TVQ} is realized by a deterministic annealing approach based on the free energy $F_{TVQ}(W,\beta) = E_{STVQ}(W,\beta,\sigma)$ as a smoothed (soft) variant of the cost function $E_{TVQ}(W,\sigma)$ formally defined as [31]:

$$E_{STVQ}(W,\beta,\sigma) = -\ln\left(\sum_{\{lc_\sigma(i,k)\}} \exp\left(-\beta E_{TVQ}(W,\sigma)\right)\right) \tag{18}$$

for a given temperature $T = \frac{1}{\beta}$. A detailed description can be found in [18]. Minimization of this free energy leads to the soft probability assignments

$$p(k|\mathbf{v}_i) = \frac{\exp(-\beta lc_\sigma(i,k))}{\sum_{l=1}^{C}\exp(-\beta lc_\sigma(i,l))} \tag{19}$$

depending on the inverse temperature β and the neighborhood range σ (E-step). High temperatures result in soft assignments, while low temperatures yield crisp assignments.

The corresponding prototypes are obtained as weighted sums

$$\mathbf{w}_k = \frac{\sum_{i=1}^{N} \mathbf{v}_i \cdot \left(\sum_{l=1}^{C} h_\sigma^{SOM}(k,l) \cdot p(l|\mathbf{v}_i) \right)}{\sum_{i=1}^{N} \sum_{l=1}^{C} h_\sigma^{SOM}(k,l) \cdot p(l|\mathbf{v}_i)} \tag{20}$$

of the data vectors $\mathbf{v}_i$ (M-step) [17]. We remark that the prototypes explicitly depend on the neighborhood range σ and implicitly also on the inverse temperature β via the assignments $p(l|\mathbf{v}_i)$.

Optimum vector quantization results are obtained for adiabatic decreasing of the neighborhood range σ while in the inner loop the temperature T is reduced and the prototypes $\mathbf{w}_k$ as well as the assignments $p(k|\mathbf{v}_i)$ are determined according to the usual EM-fixed-point-iteration.

3 Fuzzy Neural Gas, Fuzzy-SOM and Annealed Neural Gas

A combination of FCM together with SOM neighborhood according to the neighborhood function h_σ^{SOM} from (15) was suggested in [23, 22, 30, 29, 21, 26]. In this Fuzzy-SOM (FSOM) approach the M-step adaptation (3) in the original FCM model is replaced by

$$\mathbf{w}_k = \frac{\sum_{i=1}^{N} \sum_{l=1}^{C} (u_{i,l})^m \cdot h_\sigma^{SOM}(k,l) \cdot \mathbf{v}_i}{\sum_{i=1}^{N} \sum_{l=1}^{C} (u_{i,l})^m \cdot h_\sigma^{SOM}(k,l)} \tag{21}$$

incorporating the neighborhood cooperativeness known from SOMs for better stability. This corresponds to the cost function

$$E_{FSOM}(\mathbf{U}, V, W) = \sum_{k=1}^{C} \sum_{i=1}^{N} (u_{i,k})^m \cdot \left(\sum_{l=1}^{C} h_\sigma^{SOM}(k,l) \cdot (d_{i,k})^2 \right) \tag{22}$$

replacing in (2) the quadratic distances $(d_{i,k})^2$ by the local costs $lc_\sigma^{SOM}(i,k)$ from (17). Hence, the assignments reads now as

$$u_{i,k} = \frac{1}{\sum_{l=1}^{C} \left(\frac{lc_\sigma^{SOM}(i,k)}{lc_\sigma^{SOM}(i,l)} \right)^{\frac{1}{m-1}}} \tag{23}$$

We emphasize that this neighborhood cooperativeness is triggered by the external topological structure in A like in the SOM. Obviously, it can also be applied to the other models PCM, FPCM and PFCM, adapting the M-steps (7), (11), and (14) accordingly and using the same arguments in the proof as for FCM in [21, 29, 26].

An alternative to the external grid enforced neighborhood cooperativeness between prototypes is provided for the neural gas (NG) vector quantizer [28]. This approach uses a dynamic neighborhood between the prototypes determined in the data space V. This flexible neighborhood leads to the fact that NG usually outperforms SOM in crisp

vector quantization. Therefore, the idea is to incorporate this kind of neighborhood instead of the SOM-neighborhood cooperativeness into fuzzy clustering schemes to improve their performances.

In NG the neighborhood between prototypes for a given data vector $\mathbf{v}_i \in V$ is based on the winning rank of each prototype $\mathbf{w}_k$

$$rk_k\left(\mathbf{v}_i, W\right) = \sum_{l=1}^{N} \Theta\left(d\left(\mathbf{v}_i, \mathbf{w}_k\right) - d\left(\mathbf{v}_i, \mathbf{w}_l\right)\right) \tag{24}$$

where

$$\Theta\left(x\right) = \begin{cases} 0 & if \quad x \leq 0 \\ 1 & else \end{cases} \tag{25}$$

is the Heaviside function [28]. The NG neighborhood function includes the ranks according to

$$\widehat{h}_{\sigma}^{NG}\left(k|\mathbf{v}\right) = c_{\sigma}^{NG} \cdot \exp\left(-\frac{\left(rk_k\left(\mathbf{v}, W\right)\right)^2}{2\sigma^2}\right) \tag{26}$$

with neighborhood range σ. This definition allows the declaration of a gradual neighborhood relation between prototypes $\mathbf{w}_k$ and $\mathbf{w}_l$ by

$$h_{\sigma}^{NG}\left(k, l\right) = c_{\sigma}^{NG} \cdot \exp\left(-\frac{\left(rk_k\left(\mathbf{w}_l, W\right)\right)^2}{2\sigma^2}\right) \tag{27}$$

for a given neighborhood range σ. The constraint $\sum_l h_{\sigma}^{NG}\left(k, l\right) = 1$ again is ensured by a constant c_{σ}^{NG} as before. If the training of the prototypes is completed, the data are not longer needed for the calculation of the new neighborhood degree $h_{\sigma}^{NG}\left(k, l\right)$. Altough this determination is independent from the data at that time, it reflects the data relations implicitly by the pairwise dissimilarities between the prototypes $\mathbf{w}_k$ and $\mathbf{w}_l$. Hence, this neighborhood is dynamic during the learning process but becomes static in the vicinity of the equilibrium of the learning process. This allows a redefinition of the local costs (17) based on the function $h_{\sigma}^{NG}\left(k, l\right)$ as

$$lc_{\sigma}^{NG}\left(i, k\right) = \sum_{l=1}^{C} h_{\sigma}^{NG}\left(k, l\right) \cdot \left(d_{i,l}\right)^2. \tag{28}$$

Plugging these NG-based local costs $lc_{\sigma}^{NG}\left(i, k\right)$ into the most general fuzzy clustering scheme FPCM (13) we get the Fuzzy Neural Gas algorithm (FNG) with the cost function

$$E_{FNG}\left(\mathbf{U}, V, W, \delta\right) = \sum_{k=1}^{C}\sum_{i=1}^{N}\left(a \cdot u_{i,k}^{m} + b \cdot t_{i,k}^{\eta}\right) lc_{\sigma}^{NG}\left(i, k\right) + \sum_{k=1}^{C}\left(\delta_k \sum_{i=1}^{N}\left(t_{i,k} - 1\right)^{\eta}\right) \tag{29}$$

and the prototype adaptation

$$\mathbf{w}_k = \frac{\sum_{i=1}^{N}\sum_{l=1}^{C}\left(a\cdot u_{i,l}^{m}+b\cdot t_{i,l}^{\eta}\right)\cdot h_{\sigma}^{NG}(k,l)\cdot\mathbf{v}_i}{\sum_{i=1}^{N}\sum_{l=1}^{C}\left(a\cdot u_{i,l}^{m}+b\cdot t_{i,l}^{\eta}\right)\cdot h_{\sigma}^{NG}(k,l)} \tag{30}$$

if using the Euclidean distance. The adaptation of the fuzzy assignments $u_{i,l}^{m}$ and typicality assignments $t_{i,l}^{m}$ in FNG are as before defined for FCM in (1) and for FPCM in (12), respectively, but replacing there the dissimilarity measure $(d_{i,k})^2$ by the local costs $lc_{\sigma}^{NG}(i,k)$ as in FSOM:

$$u_{i,k} = \frac{1}{\sum_{l=1}^{C}\left(\frac{lc_{\sigma}^{NG}(i,k)}{lc_{\sigma}^{NG}(i,l)}\right)^{\frac{1}{m-1}}} \tag{31}$$

and

$$t_{i,k} = \frac{1}{\sum_{j=1}^{N}\left(\frac{lc_{\sigma}^{NG}(i,k)}{lc_{\sigma}^{NG}(j,k)}\right)^{\frac{1}{\eta-1}}}, \tag{32}$$

Thereby, the convergence is ensured due to the above mentioned fact that the neighborhood $h_{\sigma}^{NG}(k,l)$ becomes fixed in the convergence phase, which can be interpreted as an external grid structure as given in SOMs. However, it can be assumed that this dynamic neighborhood offers more flexibility and therefore will yield in better accuracy than a SOM-based scheme.

Yet, for the validation of this FNG it remains to show that the NG using the modified neighborhood function $h_{\sigma}^{NG}(k,l)$ from (27) instead of the original neighborhood $\widehat{h}_{\sigma}^{NG}(k|\mathbf{v})$ from (26) has a prototype update of the form

$$\triangle\mathbf{w}_i \sim -h_{\sigma}^{NG}(s(\mathbf{v}),i)\,\frac{\partial d(\mathbf{v},\mathbf{w}_i)}{\partial\mathbf{w}_i} \tag{33}$$

obtained from stochastic gradient of the underlying NG cost function. This is shown in the Appendix.

Analogously, the NG-based local costs $lc_{\sigma}^{NG}(i,k)$ could also be used in the STVQ model resulting in an Annealed Neural Gas (ANG). In fact, this ANG is similar to the annealed NG proposed in [19]. In the limit of low temperatures $T=\frac{1}{\beta}$ a crisp clustering is obtained as in STVQ depending on the neighborhood range σ. This range should be decreased adiabatically in an outer loop for optimum performance. However, it has to be mentioned here that this adiabatic decreasing of the neighborhood range σ is not equivalent to an annealing approach based on a free energy depending on this parameter σ [7].

Finally, it should be noticed at this point that these algorithms are not restricted to the Euclidean distance for the inner product norms $d_{i,k}$. More general dissimilarity measures are obviously applicable like, for example, the scaled Euclidean distance [15] or generalizations thereof, (kernelized) divergences [1] or other generalized dissimilarity measures [11].

4 Sparsity and Separation in FSOM and FNG

In FCM the fuzziness parameter m of the cost function E_{FCM} (2) implicitly controls the sparseness of the fuzzy assignments $u_{i,k}$. For $m \searrow 1$ the assignments converge to crisp decisions (maximum sparseness), whereas $m \gg 1$ forces equally distributed assignment values (in the limit $m \longrightarrow \infty$) [36]. Analogously, FNG and FSOM reduce to NG and SOM for $m \searrow 1$. Hence, controlling the factor $m = m(t)$ would be a possibility to govern the sparsity.

Otherwise, sparsity in prototype based vector quantization is closely related to information transfer optimization [6, 4]. Respective approaches for improved fuzzy assignments FCM using information theoretic concepts were presented in [10, 8, 12]. These approaches either enforce maximum Shannon entropy [42]

$$H_i^S(\mathbf{U}) = -\sum_{k=1}^{C} u_{i,k} \cdot \ln(u_{i,k}) \tag{34}$$

as the main optimization goal with the minimum description error taken as a constraint [12], or add a Shannon entropy term to the cost function of FCM (used with $m = 1$) to avoid crisp assignments [10]. Instead of the Shannon entropy also the fuzzy entropy [34]

$$H_i^{FS}(\mathbf{U}) = -\sum_{k=1}^{C} (u_{i,k} \cdot \ln(u_{i,k}) - (1 - u_{i,k}) \cdot \ln(1 - u_{i,k})) \tag{35}$$

was investigated [8]. But in this approach, the fuzzy entropy $HF_i(\mathbf{U})$ was taken as an additional regularization term for the Lagrange optimization. Another possibility is to take the Kullback-Leibler divergence [41] of the distribution of the fuzzy assignments $u_{i,k}$ compared with a prior distribution π_k reflecting the ratio of the data contributing to the kth fuzzy cluster (prototype) as regularization [14].

Here, we investigate entropies regarding their usefulness in FNG/FSOM sparsity control. For this purpose, we consider different entropies as an additional term in the cost functions of FSOM and FNG, whereby the sparsity parameter m is set to $m = 1$. As mentioned above, $m = 1$ yields crisp decisions. The add of scaled negative entropies to the cost function enforces fuzziness and, therefore, implicitly controls sparseness. Thus, the new cost function based on the original cost function E_{FNG} (22) of FNG with $m = 1$ becomes

$$E_{FNG}(\mathbf{U}, V, W) = \sum_{k=1}^{C} \sum_{i=1}^{N} \left(lc_\sigma^{NG}(i,k)\right) - \gamma(t) \cdot H_i(\{u_{i,l}\}) \tag{36}$$

with an entropy $H_i(\{u_{i,k}\})$ judging the fuzziness or separation degree and the local costs $lc_\sigma^{NG}(i,k)$ from (28). The factor $\gamma(t) > 0$ explicitly controls the fuzziness at time t. The regularization conditions $\sum_k u_{i,k} = 1$ for the assignments $u_{i,k}$ from FCM

should preserved in this model, such that the resulting Lagrange function is

$$L_{FNG}\left(\mathbf{U},V,W\right)=E_{FNG}\left(\mathbf{U},V,W\right)+\sum_{i=1}^{N}\varsigma_i\left(1-\sum_{k=1}^{C}\left(u_{i,k}\right)\right) \tag{37}$$

depending on the used entropy. The FSOM scheme can be handled analogously to deal with sparsity. In this manner, information theoretic concepts are used for sparsity/fuzzyness control instead of optimization of the fuzziness parameter m.

Obviously, the adaptation scheme for the prototypes is not influenced by the entropy term, because the entropies H_i do not depend on the prototypes and, therefore, $\frac{\partial H_i(\{u_{i,k}\})}{\partial \mathbf{w}_j}=0$ is valid. Hence, only the adaptation of the fuzzy assignments is affected by this sparsity term. In the following we consider different types of entropies: the Shannon, the Rényi, the Tsallis and the Burg entropy as well as their fuzzy counterparts. It turns out that only Shannon entropy, Tsallis entropy and Fuzzy Tsallis entropy are applicable.

4.1 Shannon entropy and Fuzzy Shannon entropy

For the Shannon entropy (34) the derivative of the Lagrangian (37) is

$$\frac{\partial L_{FNG}^{S}\left(\mathbf{U},V,W\right)}{\partial u_{j,r}}=lc_{\sigma}^{NG}\left(j,r\right)+\gamma\left(t\right)\cdot\left(\ln\left(u_{j,r}\right)+1\right)-\varsigma_j \tag{38}$$

with the necessary condition $\frac{\partial L_{FNG}^{S}(\mathbf{U},V,W)}{\partial u_{j,r}}=0$ for an extremum. For the case $m=1$ it can be explicitly solved such that

$$u_{j,r}=\exp\left(-\frac{lc_{\sigma}^{NG}\left(j,r\right)}{\gamma\left(t\right)}+\left(\frac{\varsigma_j}{\gamma\left(t\right)}-1\right)\right). \tag{39}$$

Summing up over all prototypes, we get

$$1=\sum_{k=1}^{C}\left(\exp\left(-\left(lc_{\sigma}^{NG}\left(j,k\right)\right)+\left(\frac{\varsigma_j}{\gamma\left(t\right)}-1\right)\right)\right) \tag{40}$$

and, consequently,

$$\frac{\varsigma_j}{\gamma\left(t\right)}-1=-\ln\left(\sum_{k=1}^{C}\exp\left(-\frac{lc_{\sigma}^{NG}\left(j,k\right)}{\gamma\left(t\right)}\right)\right). \tag{41}$$

Replacing this in (39) we finally obtain

$$u_{j,r}=\frac{\exp\left(-\frac{lc_{\sigma}^{NG}(j,r)}{\gamma(t)}\right)}{\sum_{k=1}^{C}\exp\left(-\frac{lc_{\sigma}^{NG}(j,k)}{\gamma(t)}\right)} \tag{42}$$

for fuzzy assignment adaptation in FNG. As we can see the adaptaion is very sensitive according to the $\gamma(t)$ value.

For the Fuzzy Shannon entropy (35) the derivative of the respective Lagrangian is

$$\frac{\partial L_{FNG}^{FS}(\mathbf{U},V,W)}{\partial u_{j,r}} = lc_{\sigma}^{NG}(j,r) + \gamma(t) \cdot \ln\left(\frac{u_{j,r}}{1-u_{j,r}}\right) - \varsigma_j \tag{43}$$

with the necessary condition $\frac{\partial L_{FNG}^{FS}(\mathbf{U},V,W)}{\partial u_{j,r}} = 0$ for an extremum. Thus we get

$$\frac{u_{j,r}}{1-u_{j,r}} = \exp\left(-\frac{lc_{\sigma}^{NG}(j,r)}{\gamma(t)} + \frac{\varsigma_j}{\gamma(t)}\right) \tag{44}$$

or equivalently

$$u_{j,r} = \frac{1}{\left(1+\exp\left(-\frac{lc_{\sigma}^{NG}(j,r)}{\gamma(t)} + \frac{\varsigma_j}{\gamma(t)}\right)\right)} \tag{45}$$

Summing up over all prototypes, we get

$$1 = \sum_{k=1}^{C} \frac{1}{\left(1+\exp\left(-\frac{lc_{\sigma}^{NG}(j,k)}{\gamma(t)} + \frac{\varsigma_j}{\gamma(t)}\right)\right)} \tag{46}$$

which cannot be explicitly solved for the Lagrange variables ς_j. Hence, the Fuzzy Shannon entropy (35) cannot be applied automatically. Therefore, an explicit normalization of the fuzzy assigments (45) was introduced in [8], which still contains the Lagrange parameters and, hence, is being unsatisfactory. For solution of this problem a deterministic as well as simulated annealing procedure were proposed in [8].

4.2 Rényi entropy and Fuzzy Rényi entropy

Instead of Shannon entropies, Rényi entropies can be applied alternatively [40]. They seem to be to less sensitive [4] and are defined as

$$H_i^R(\alpha,\{u_{i,k}\}) = \frac{1}{1-\alpha}\ln\left(\sum_{k=1}^{C}(u_{i,k})^{\alpha}\right) \tag{47}$$

with the derivatives

$$\frac{\partial H_i^R(\alpha,\{u_{i,k}\})}{\partial u_{j,r}} = \frac{\alpha}{1-\alpha}\frac{(u_{j,r})^{\alpha-1}}{\sum_{k=1}^{C}(u_{j,k})^{\alpha}}. \tag{48}$$

Analogously, the Fuzzy Rényi entropies can be defined as

$$H_i^{FR}(\alpha,\{u_{i,k}\}) = \frac{1}{1-\alpha}\ln\left(\sum_{k=1}^{C}((u_{i,k})^{\alpha}+(1-u_{i,k})^{\alpha})\right) \tag{49}$$

with the derivative

$$\frac{\partial H_i^{FR}\left(\alpha,\{u_{i,k}\}\right)}{\partial u_{j,r}} = \frac{\alpha}{1-\alpha}\frac{\left(u_{j,r}\right)^{\alpha-1}-\left(1-u_{j,r}\right)^{\alpha-1}}{\sum_{k=1}^{C}\left(\left(u_{i,k}\right)^{\alpha}+\left(1-u_{i,k}\right)^{\alpha}\right)} \tag{50}$$

According to the stability, the case $\alpha = 2$ is of special interest [16]. The derivative of the respective Lagrangian for the Rényi entropy (47) has the form

$$\frac{\partial L_{FNG}^{R}\left(\mathbf{U},V,W\right)}{\partial u_{j,r}} = lc_{\sigma}^{NG}\left(j,r\right)+\gamma\left(t\right)\frac{\alpha}{1-\alpha}\frac{\left(u_{j,r}\right)^{\alpha-1}}{\sum_{k=1}^{C}\left(u_{j,k}\right)^{\alpha}}-\varsigma_j \tag{51}$$

which reduces for $\alpha = 2$

$$\frac{\partial L_{FNG}^{R}\left(\mathbf{U},V,W\right)}{\partial u_{j,r}} = lc_{\sigma}^{NG}\left(j,r\right)-\frac{2\gamma\left(t\right)u_{j,r}}{\sum_{k=1}^{C}\left(u_{j,k}\right)^{2}}-\varsigma_j. \tag{52}$$

For $\frac{\partial L_{FNG}^{R}(\mathbf{U},V,W)}{\partial u_{j,r}} = 0$ we obtain

$$u_{j,r} = \frac{lc_{\sigma}^{NG}\left(j,r\right)-\varsigma_j}{2\gamma\left(t\right)}\left(\sum_{k=1}^{C}\left(u_{j,k}\right)^{2}\right) \tag{53}$$

which cannot be explicitly solved. Hence, although the quadratic Rényi entropy is frequently recommended because of its stability property, it is not applicable here.

For the Fuzzy Rényi entropy (49) considering again $\alpha = 2$ we analogously obtain

$$\frac{\partial L_{FNG}^{FR}\left(\mathbf{U},V,W\right)}{\partial u_{j,r}} = lc_{\sigma}^{NG}\left(j,r\right)-\frac{2\gamma\left(t\right)\left(2u_{j,r}-1\right)}{\sum_{k=1}^{C}\left(\left(u_{j,k}\right)^{2}+\left(1-u_{i,k}\right)^{2}\right)}-\varsigma_j, \tag{54}$$

which causes similar difficulties and, therefore, is also not feasible.

4.3 Tsallis entropy and Fuzzy Tsallis entropy

The Tsallis entropy [33] is closely related to the Rényi-entropy [5, 2]. It does not include a logarithm and is given as

$$H_i^{T}\left(\{u_{i,k}\}\right) = \frac{1}{1-\alpha}\left(1-\sum_{k=1}^{C}\left(u_{i,k}\right)^{\alpha}\right) \tag{55}$$

whereas its fuzzy counterpart may be defined as

$$H_i^{FT}\left(\{u_{i,k}\}\right) = \frac{1}{1-\alpha}\left(1-\sum_{k=1}^{C}\left(\left(u_{i,k}\right)^{\alpha}+\left(1-u_{i,k}\right)^{\alpha}\right)\right) \tag{56}$$

As the Rényi entropy, the Tsallis entropy is numerically stable also for small values $u_{i,k}$ and becomes easy in calculation for $\alpha = 2$. We get for the respective Lagrangian

$$\frac{\partial L_{FNG}^{T}\left(\mathbf{U},V,W\right)}{\partial u_{j,r}} = lc_{\sigma}^{NG}\left(j,r\right) - 2\gamma\left(t\right)u_{j,r} - \varsigma_{j} \tag{57}$$

and, therefore, we get for $\frac{\partial L_{FNG}^{T}(\mathbf{U},V,W)}{\partial u_{j,r}} = 0$:

$$u_{j,r} = \frac{lc_{\sigma}^{NG}\left(j,r\right) - \varsigma_{j}}{2\gamma\left(t\right)}. \tag{58}$$

Summing up we obtain

$$1 = \sum_{k=1}^{C} \frac{lc_{\sigma}^{NG}\left(j,k\right) - \varsigma_{j}}{2\gamma\left(t\right)} \tag{59}$$

such that

$$\varsigma_{j} = \frac{\left(\sum_{k=1}^{C} lc_{\sigma}^{NG}\left(j,k\right)\right) - 2\gamma\left(t\right)}{C}. \tag{60}$$

This finally leads to

$$u_{j,r} = \frac{C \cdot lc_{\sigma}^{NG}\left(j,r\right) - \left(\sum_{k=1}^{C} lc_{\sigma}^{NG}\left(j,k\right)\right) - 2\gamma\left(t\right)}{2C \cdot \gamma\left(t\right)} \tag{61}$$

for the fuzzy assignment adaptation in case of the quadratic Tsallis entropy.

Looking now at the fuzzy variant (56) we have

$$\frac{\partial L_{FNG}^{FT}\left(\mathbf{U},V,W\right)}{\partial u_{j,r}} = lc_{\sigma}^{NG}\left(j,r\right) - 2\gamma\left(t\right)\left(2u_{j,r} - 1\right) - \varsigma_{j} \tag{62}$$

yielding

$$u_{j,r} = \frac{lc_{\sigma}^{NG}\left(j,r\right) - \varsigma_{j}}{4\gamma\left(t\right)} + \frac{1}{2} \tag{63}$$

for $\frac{\partial L_{FNG}^{FT}(\mathbf{U},V,W)}{\partial u_{j,r}} = 0$. Summing up again we obtain

$$\varsigma_{j} = \frac{\left(\sum_{k=1}^{C} lc_{\sigma}^{NG}\left(j,k\right)\right) - \gamma\left(t\right) \cdot \left(2 - C\right)}{C} \tag{64}$$

which finally leads to

$$u_{j,r} = \frac{C \cdot lc_{\sigma}^{NG}\left(j,r\right) - \left(\sum_{k=1}^{C} lc_{\sigma}^{NG}\left(j,k\right)\right) - \gamma\left(t\right) \cdot \left(2 - C\right)}{4C \cdot \gamma\left(t\right)} + \frac{1}{2} \tag{65}$$

applicable in FNG/FSOM.

4.4 The Burg entropy

The last entropy considered here is Burg entropy frequently applied in time series analysis [5, 2]. This entropy is similar to the Shannon entropy:

$$H_i^B(\mathbf{U}) = -\sum_{k=1}^{C} \ln(u_{i,k}) \tag{66}$$

and the Fuzzy Burg entropy we define as

$$H_i^{FB}(\mathbf{U}) = -\sum_{k=1}^{C} \ln(u_{i,k}) \cdot \ln(1-u_{i,k}) \tag{67}$$

in complete analogy. The derivative of the Lagrangian becomes

$$\frac{\partial L_{FNG}^B(\mathbf{U},V,W)}{\partial u_{j,r}} = lc_\sigma^{NG}(j,r) + \frac{\gamma(t)}{u_{j,r}} - \varsigma_j \tag{68}$$

resulting for $\frac{\partial L_{FNG}^B(\mathbf{U},V,W)}{\partial u_{j,r}} = 0$ in

$$u_{j,r} = \frac{\gamma(t)}{\varsigma_j - lc_\sigma^{NG}(j,r)}. \tag{69}$$

After summation we have

$$1 = \sum_{k=1}^{C} \frac{\gamma(t)}{\varsigma_j - lc_\sigma^{NG}(j,k)} \tag{70}$$

which is similar to the result obtained in (46) for the Fuzzy Shannon entropy. Hence, the same arguments are valid: The Burg entropy is applicable in principle using the fuzzy assignments according to (69) but with necessary renormalization thereafter and the remaining problem of non-vanishing Lagrange variables ς_j. A simulated/deterministic annealing strategy according to [8] could offer a solution for that but not investigated so far.

For the Fuzzy Burg entropy we get the Lagrangian

$$\frac{\partial L_{FNG}^B(\mathbf{U},V,W)}{\partial u_{j,r}} = lc_\sigma^{NG}(j,r) + \frac{\gamma(t)}{u_{j,r}} \frac{2u_{j,r}-1}{u_{j,r}-1} - \varsigma_j \tag{71}$$

to be set to zero. This leads to a quadratic equation in $u_{j,r}$ with the solution

$$u_{j,r} = \frac{1}{2(\varsigma_j - lc_\sigma^{NG}(j,r))} \left(\varsigma_j - 2\gamma(t) - lc_\sigma^{NG}(j,r) \pm \sqrt{(\varsigma_j - lc_\sigma^{NG}(j,r))^2 + (2\gamma(t))^2} \right) \tag{72}$$

which has to be considered further again resulting in the above discussed difficulties.

5 Conclusion

In this article we provide the theoretical framework for the combination of the fuzzy-c-means algorithm and its variants with the neighborhood cooperativeness learning as known from the neural gas quantizer. Using a redefinition of the NG neighborhood it can easily applied to FCM. This modification is carried out in complete analogy to the incorporation of the SOM-neighborhood concept into FCM known as FSOM. Yet, the resulting FNG offers a greater flexibility compared to FSOM, because the latter one depends on a lattice structure, which has to be chosen apriori. This behavior is also observable comparing SOM and NG for crisp vector quantization.

Further, we focus on sparsity enforcement for fuzzy assignments using information theoretic concepts while keeping the regularization condition from FCM. This latter property distinguishs these methodology from other approaches. It turns out that amoung the several entropies only Shannon, Tsallis as well as Fuzzy Tsallis entropy are applicable for control.

A

ppendix - Convergence proof of the modified neural gas

In this section we consider the original NG but with the neighborhood function $h_\sigma^{NG}(k,l)$ defined in (27). Supposing data points $\mathbf{v} \in V \subset \mathbb{R}^n$ with the data density $P(\mathbf{v})$ and prototypes $\mathbf{w}_j \in \mathbb{R}^n$, $j = 1 \ldots N$, the cost function to be minimized by the NG now is

$$E_{\mathrm{NG}} = \frac{1}{2} \sum_j \int P(\mathbf{v})\, h_\sigma^{NG}(s(\mathbf{v}), j)\, (d(\mathbf{v}, \mathbf{w}_j))^2\, d\mathbf{v} \tag{73}$$

whereby the new NG-neighborhood function (27) is used instead of the original one (26). Further, we explicitly require the obvious separability assumption that $\mathbf{w}_j \neq \mathbf{w}_i$ for $j \neq i$. We have to show that the averaged change $\langle \mathbf{w}_i \rangle$ is the gradient of E_{NG}.

Following the work of MARTINETZ ET AL. [28] we have to investigate

$$\frac{\partial E_{\mathrm{NG}}}{\partial \mathbf{w}_i} = R_i + \int P(\mathbf{v})\, h_\sigma^{NG}(s(\mathbf{v}), i)\, \frac{\partial d(\mathbf{v}, \mathbf{w}_i)}{\partial \mathbf{w}_i} d\mathbf{v} \tag{74}$$

with the winner-take-all mapping rule

$$s(\mathbf{v}_i) = \mathrm{argmin}_j\, (d(\mathbf{v}_i, \mathbf{w}_j)) \tag{75}$$

determining the best matching prototype $\mathbf{w}_s$. The second term in (73) is the desired averaged change $\langle \mathbf{w}_i \rangle$, which is equivalent to the stochastic gradient descent learning rule (33). Hence, it remains to show that R_i is vanishing.

The term R_i is obtained as

$$R_i = \frac{1}{2} \sum_j \int P(\mathbf{v})\, \frac{\partial h_\sigma^{NG}(s(\mathbf{v}), j)}{\partial \mathbf{w}_i}\, (d(\mathbf{v}, \mathbf{w}_j))^2\, d\mathbf{v} \tag{76}$$

with

$$\frac{\partial h_\sigma^{NG}(s(\mathbf{v}), j)}{\partial \mathbf{w}_i} = \left[h_\sigma^{NG}\right]'(s(\mathbf{v}), j) \cdot \frac{\partial rk_j(\mathbf{w}_s, W)}{\partial \mathbf{w}_i}$$

and $\left[h_\sigma^{NG}\right]'(\bullet)$ denoting the derivative of $h_\sigma^{NG}(\bullet)$. Using the definition of the rank function (24) we get

$$\frac{\partial rk_j(\mathbf{w}_s, W)}{\partial \mathbf{w}_i} = \frac{\partial}{\partial \mathbf{w}_i} \left[\sum_{l=1}^{N} \Theta(d(\mathbf{w}_s, \mathbf{w}_j) - d(\mathbf{w}_s, \mathbf{w}_l)) \right] \tag{77}$$

which allows a decomposition of R_i into $R_i = R_{i,1} + R_{i,2}$ such that

$$R_{i,1} = \int P(\mathbf{v}) \left[h_\sigma^{NG}\right]'(rk_i(\mathbf{w}_s, W)) \cdot (d(\mathbf{v}, \mathbf{w}_i))^2\, \frac{\partial d(\mathbf{w}_s, \mathbf{w}_i)}{\partial \mathbf{w}_i} \sum_l \theta(\triangle_{il})\, d\mathbf{v}$$

and

$$-R_{i,2} = \sum_j \int P(\mathbf{v}) \left[h_\sigma^{NG}\right]' (rk_j(\mathbf{w}_s, W)) \cdot (d(\mathbf{v}, \mathbf{w}_j))^2 \cdot \frac{\partial d(\mathbf{w}_s, \mathbf{w}_i)}{\partial \mathbf{w}_i} \cdot \theta(\triangle_{ji})\, d\mathbf{v}$$

with $\triangle_{mk} = d(\mathbf{w}_s, \mathbf{w}_m) - d(\mathbf{w}_s, \mathbf{w}_k)$. Thereby we have used the fact that the derivative of the Heaviside function $\Theta(x)$ from (25) is the Dirac distribution $\theta(x)$, which is zero iff $x \neq 0$ and $\int \theta(x)\, dx = 1$. Further, the relation $\theta(x) = \theta(-x)$ holds. Hence, taking into account the above assumption about the separability of the prototypes, we can conclude that $\theta(\triangle_{ji}) \neq 0$ holds iff $d(\mathbf{w}_s, \mathbf{w}_i) = d(\mathbf{w}_s, \mathbf{w}_j)$ and, therefore, we can replace $(d(\mathbf{v}, \mathbf{w}_j))^2$ by $(d(\mathbf{v}, \mathbf{w}_i))^2$ for those non-vanishing cases of $\theta(\triangle_{ji})$. Thus we have in this situation

$$-R_{i,2} = \int P(\mathbf{v}) \left[h_\sigma^{NG}\right]' (rk_i(\mathbf{w}_s, W)) \cdot (d(\mathbf{v}, \mathbf{w}_i))^2 \cdot \frac{\partial d(\mathbf{w}_s, \mathbf{w}_i)}{\partial \mathbf{w}_i} \cdot \sum_j \theta(\triangle_{ji})\, d\mathbf{v} \tag{78}$$

using the same arguments as in [28]. This immediately implies $R_{i,1} = -R_{i,2}$, which completes the proof.

References

[1] T. Villmann and S. Haase. Divergence based vector quantization. *Neural Computation*, 23(5):1343–1392, 2011.

[2] A. Cichocki and S.-I. Amari. Families of alpha- beta- and gamma- divergences: Flexible and robust measures of similarities. *Entropy*, 12:1532–1568, 2010.

[3] T. Geweniger, D. Zühlke, B. Hammer, and T. Villmann. Median fuzzy c-means for clustering dissimilarity data. *Neurocomputing*, 73(7–9):1109–1116, 2010.

[4] J.C. Principe. *Information Theoretic Learning*. Springer, Heidelberg, 2010.

[5] A. Cichocki, R. Zdunek, A.H. Phan, and S.-I. Amari. *Nonnegative Matrix and Tensor Factorizations*. Wiley, Chichester, 2009.

[6] K. Labusch, E. Barth, and T. Martinetz. Sparse coding neural gas: Learning of overcomplete data representations. *Neuro*, 72(7-9):1547–1555, 2009.

[7] T. Villmann, B. Hammer, and M. Biehl. Some theoretical aspects of the neural gas vector quantizer. In M. Biehl, B. Hammer, M. Verleysen, and T. Villmann, editors, *Similarity-based Clustering*, volume 5400 of *LNAI*, pages 23–34. Springer, Berlin, 2009.

[8] M. Yasuda and T. Furuhashi. Fuzzy entropy based fuzzy-c-means clustering with deterministic and simulated annealing methods. *IEICE Transactions on Information and Systems*, E92-D:1232–1239, 2009.

[9] R. Inokuchi and S. Miyamoto. Fuzzy c-means algorithms using Kullback-Leibler divergence and Hellinger distance based on multinomial manifold. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 12(5):443–447, 2008.

[10] S. Miyamoto, H. Ichihashi, and K. Honda. *Algorithms for Fuzzy Clustering*, volume 229 of *Studies in Fuzziness and Soft Computing*. Springer, 2008.

[11] E. Pekalska and R.P.W. Duin. *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications*. World Scientific, 2006.

[12] M. Ghorbani. Maximum entropy-based fuzzy clustering by using L_1 norm space. *Turkish Journal of Mathematics*, 29(4):431–438, 2005.

[13] N.R. Pal, K. Pal, J.M. Keller, and J.C. Bezdek. A possibilistic fuzzy c-means clustering algorithm. *IEEE Transactions on Fuzzy Systems*, 13(4):517–530, 2005.

[14] H. Ichihashi and K. Honda. Application of kernel trick to fuzzy c-means with regularization by K-L information. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 8(6):566–572, 2004.

[15] B. Hammer and Th. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.

[16] J. C. Principe, J.W. Fischer III, and D. Xu. Information theoretic learning. In S. Haykin, editor, *Unsupervised Adaptive Filtering*. Wiley, New York, NY, 2000.

[17] T. Graepel, M. Burger, and K. Obermayer. Self-organizing maps: generalizations and new optimization techniques. *Neurocomputing*, 21(1–3):173–90, 1998.

[18] T. Graepel, M. Burger, and K. Obermayer. Phase transitions in stochastic self-organizing maps. *Physical Review E [Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics]*, 56(4):3876–90, 1997.

[19] T. Hofmann and J.M. Buhmann. An annealed "Neural Gas" network for robust vector quantization. In C. v. d. Malsburg, W. v. Seelen, J.C. Vorbrüggen, and B. Sendhoff, editors, *Proc. Int. Conference on Artificial Neural Networks (ICANN)*, volume 1112 of *LNCS*, pages 151–156, Berlin, 1996. Springer.

[20] R. Krishnapuram and J. Keller. The possibilistic c-means algorithm: insights and recommendations. *IEEE Transactions on Fuzzy Systems*, 4(3):385–393, 1996.

[21] N. R. Pal, J. C. Bezdek, and E. C. K. Tsao. Errata to Generalized clustering networks and Kohonen's self-organizing scheme. *IEEE Transactions on Neural Networks*, 6(2):521–521, March 1995.

[22] James C. Bezdek and Nikhil R. Pal. A note on self-organizing semantic maps. *IEEE Transactions on Neural Networks*, 6(5):1029–1036, 1995.

[23] James C. Bezdek and Nikhil R. Pal. Two soft relatives of learning vector quantization. *Neural Networks*, 8(5):729–743, 1995.

[24] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).

[25] S. P. Luttrell. A Bayesian analysis of self-organising maps. *Neural Computation*, 6(5):767–794, 1994.

[26] E.C. Tsao, J.C. Bezdek, and N.R. Pal. Fuzzy Kohonen clustering networks. *Pattern Recognition*, 27(5):757–764, 1994.

[27] R. Krishnapuram and J. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(4):98–110, 1993.

[28] Thomas M. Martinetz, Stanislav G. Berkovich, and Klaus J. Schulten. 'Neural-gas' network for vector quantization and its application to time-series prediction. *IEEE Trans. on Neural Networks*, 4(4):558–569, 1993.

[29] Nikhil R Pal, James C Bezdek, and Eric C K Tsao. Generalized clustering networks and Kohonen's self-organizing scheme. *IEEE Transactions on Neural Networks*, 4(4):549–557, 1993.

[30] J. C. Bezdek, E. C. K. Tsao, and N. R. Pal. Fuzzy Kohonen clustering networks. In *Proc. IEEE International Conference on Fuzzy Systems*, pages 1035–1043, Piscataway, NJ, 1992. IEEE Service Center.

[31] K. Rose, E. Gurewitz, and G.C. Fox. Vector quantization by deterministic annealing. *IEEE Transactions on Information Theory*, 38(4):1249–1257, 1992.

[32] J.C. Bezdek, R.J. Hathaway, and M.P. Windham. Numerical comparison of RFCM and AP algorithms for clustering relational data. *Pattern recognition*, 24:783–791, 1991.

[33] C. Tsallis. Possible generalization of Bolzmann-Gibbs statistics. *Journal oft Mathematical Physics*, 52:479–487, 1988.

[34] B. Kosko. Fuzzy entropy and conditioning. *Information Sciences*, 40:165–174, 1986.

[35] J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, 1981.

[36] J.C. Bezdek. A convergence theorem for the fuzyy ISODATA clustering algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):1–8, 1980.

[37] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95, 1980.

[38] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[39] J.C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.

[40] A. Renyi. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1961.

[41] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[42] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–432, 1948.

Relational Extensions of Learning Vector Quantization

X. Zhu, F.-M. Schleif, B. Hammer[1,2]

Abstract

Prototype-based learning algorithms offer an intuitive interface to given data sets by means of an inspection of the prototypes. Supervised classification can be achieved by popular techniques such as learning vector quantization (LVQ) and extensions derived from cost functions such as generalized LVQ (GLVQ) and robust soft LVQ (RSLVQ). These methods, however, are restricted to Euclidean vectors. Thus they are unsuitable for complex or heterogeneous data sets where input dimensions have different relevance or a high dimensionality yields to accumulated noise which disrupts the classifications. Although this problem can partially be avoided by appropriate metric learning, or by kernel variants, however, if data are inherently non-Euclidean, the techniques cannot be applied. In modern applications, data are often addressed using dedicated non-Euclidean dissimilarities such as dynamic time warping for time series, alignment for symbolic strings, the compression distance to compare sequences based on an information theoretic ground, and similar. These settings do not allow an Euclidean representation of data at all, rather, data are given implicitly in terms of pairwise dissimilarities or relations. In this contribution, we propose relational extensions of GLVQ and RSLVQ, which can directly be applied to relational data sets which are characterized in terms of a symmetric dissimilarity matrix only. The optimization can take place using gradient techniques. We test these techniques on several benchmarks, leading to results comparable to SVM while providing prototype based presentations.

[1]E-mail: xzhu|fschleif|bhammer@techfak.uni-bielefeld.de
[2]CITEC Centre of excellence,
Technical Department
Bielefeld University
33615 Bielefeld, Germany

1 Supervised prototype-based learning: GLVQ, RSLVQ

In the classical vectorial space, data $\mathbf{x}^i \in \mathbb{R}^n, i = 1, \ldots, m$, are given. Prototypes are vectors $\mathbf{w}^j \in \mathbb{R}^n, j = 1, \ldots, k$ in the same space, and they divide data into into receptive fields

$$R(\mathbf{w}^j) := \{\mathbf{x}^i \, : \, \forall k \; d(\mathbf{x}^i, \mathbf{w}^j) \leq d(\mathbf{x}^i, \mathbf{w}^k)\}$$

based on the squared Euclidean distance

$$d(\mathbf{x}^i, \mathbf{w}^j) = \|\mathbf{x}^i - \mathbf{w}^j\|^2 .$$

The goal of prototype-base learning techniques is to find prototypes which represent a given data set as accurately as possible. Unsupervised optimized the quantization error

$$E_{\mathrm{qe}} = \frac{1}{2} \sum_{i,j} \chi_i(\mathbf{x}^i) d(\mathbf{x}^i, \mathbf{w}^j) \tag{1}$$

where

$$\chi_j(\mathbf{x}) = \begin{cases} 1 & \text{if } d(\mathbf{x}, \mathbf{w}^j) \leq d(\mathbf{x}, \mathbf{w}^k) \text{ for all } k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

indicate whether $\mathbf{w}^j$ is prototype of data $\mathbf{x}$. Although the quantization error is one of the most popular measures to evaluate unsupervised clustering, it is often not sufficient in practical applications due to several aspects: it suffers from numerical problems due to the multi modality of the cost function and its sensitivity to noise and outliers. In addition, further functionalities are often required in application scenarios such as the possibility to visualize the prototypes and to inspect relations in between prototypes. Both problems are addressed by topographic mapping, such as SOM. NG, and GTM.

For supervised learning, data $\mathbf{x}^i$ are equipped with class labels $c(\mathbf{x}^i) \in \{1, \ldots, L\}$. Similarly, every prototype is equipped with a priorly fixed label $c(\mathbf{w}^j)$. A data point is mapped to the closest prototype and classified according to the class of the prototype. The classification error of this mapping is given by the term

$$\sum_j \sum_{\mathbf{x}^i \in R(\mathbf{w}^j)} \delta(c(\mathbf{x}^i) \neq c(\mathbf{w}^j))$$

with the delta function δ. This cost function cannot easily be optimized explicitly due to vanishing gradients and discontinuities. Therefore, LVQ relies on a reasonable heuristic by performing Hebbian updates of the prototypes, given a data point [12]. Recent alternatives derive similar update rules from explicit cost functions which are related to the classification error, but display better numerical properties such that efficient optimization algorithms can be derived thereof [3, 15, 11].

Generalized LVQ (GLVQ) has been proposed in the approach [15]. It is derived from a cost function which can be related to the generalization ability of LVQ classifiers [3].

The cost function of GLVQ is given as

$$E_{\mathrm{GLVQ}} = \sum_i \Phi\left(\frac{d(\mathbf{x}^i, \mathbf{w}^+(\mathbf{x}^i)) - d(\mathbf{x}^i, \mathbf{w}^-(\mathbf{x}^i))}{d(\mathbf{x}^i, \mathbf{w}^+(\mathbf{x}^i)) + d(\mathbf{x}^i, \mathbf{w}^-(\mathbf{x}^i))}\right)$$

where Φ is a differentiable monotonic function such as the hyperbolic tangent, and $\mathbf{w}^+(\mathbf{x}^i)$ refers to the prototype closest to $\mathbf{x}^i$ with the same label as $\mathbf{x}^i$, $\mathbf{w}^-(\mathbf{x}^i)$ refers to the closest prototype with a different label. This way, for every data point, its contribution to the cost function is small if and only if the distance to the closest prototype with a correct label is smaller than the distance to a wrongly labeled prototype, resulting in a correct classification of the point and, at the same time, by optimizing this so-called hypothesis margin of the classifier, aiming at a good generalization ability.

A learning algorithm can be derived thereof by means of a stochastic gradient descent. After a random initialization of prototypes, data $\mathbf{x}^i$ are presented in random order. Adaptation of the closest correct and wrong prototype takes place by means of the update rules

$$\begin{aligned}
\Delta\mathbf{w}^+(\mathbf{x}^i) &\sim -\Phi'(\mu(\mathbf{x}^i)) \cdot \mu^+(\mathbf{x}^i) \cdot \nabla_{\mathbf{w}^+(\mathbf{x}^i)} d(\mathbf{x}^i, \mathbf{w}^+(\mathbf{x}^i)) \\
\Delta\mathbf{w}^-(\mathbf{x}^i) &\sim \Phi'(\mu(\mathbf{x}^i)) \cdot \mu^-(\mathbf{x}^i) \cdot \nabla_{\mathbf{w}^-(\mathbf{x}^i)} d(\mathbf{x}^i, \mathbf{w}^-(\mathbf{x}^i))
\end{aligned}$$

where

$$\mu(\mathbf{x}^i) = \frac{d(\mathbf{x}^i, \mathbf{w}^+(\mathbf{x}^i)) - d(\mathbf{x}^i, \mathbf{w}^-(\mathbf{x}^i))}{d(\mathbf{x}^i, \mathbf{w}^+(\mathbf{x}^i)) + d(\mathbf{x}^i, \mathbf{w}^-(\mathbf{x}^i))},$$

$$\mu^+(\mathbf{x}^i) = \frac{2 \cdot d(\mathbf{x}^i, \mathbf{w}^-(\mathbf{x}^i))}{(d(\mathbf{x}^i, \mathbf{w}^+(\mathbf{x}^i)) + d(\mathbf{x}^i, \mathbf{w}^-(\mathbf{x}^i))^2},$$

and

$$\mu^-(\mathbf{x}^i) = \frac{2 \cdot d(\mathbf{x}^i, \mathbf{w}^+(\mathbf{x}^i))}{(d(\mathbf{x}^i, \mathbf{w}^+(\mathbf{x}^i)) + d(\mathbf{x}^i, \mathbf{w}^-(\mathbf{x}^i))^2}.$$

For the squared Euclidean norm, the derivative yields

$$\nabla_{\mathbf{w}^j} d(\mathbf{x}^i, \mathbf{w}^j) = -2(\mathbf{x}^i - \mathbf{w}^j),$$

leading to Hebbian update rules of the prototypes which take into account the priorly known class information, i.e. they adapt the closest prototypes towards / away from a given data point depending on their labels. GLVQ constitutes one particularly efficient method to adapt the prototypes according to a given labeled data sets.

Robust soft LVQ (RSLVQ) as proposed in [11] is an alternative approach which is based on a statistical model of the data. In the limit of small bandwidth, update rules which are very similar to LVQ result. For non-vanishing bandwidth, soft assignments of data points to prototypes take place. Every prototype induces a probability induced by Gaussians, for example, i.e. $p(\mathbf{x}^i|\mathbf{w}^j) = K \cdot \exp(-d(\mathbf{x}^i, \mathbf{w}^j)/2\sigma^2)$ with parameter

$\sigma \in \mathbb{R}$ and normalization constant $K = (2\pi\sigma^2)^{-n/2}$. Assuming that every prototype has the same prior, we obtain the overall probability of a data point

$$p(\mathbf{x}^i) = \sum_{\mathbf{w}^j} p(\mathbf{x}^i|\mathbf{w}^j)/k$$

and the probability of a point and its corresponding class

$$p(\mathbf{x}^i, c(\mathbf{x}^i)) = \sum_{\mathbf{w}^j : c(\mathbf{w}^j)=c(\mathbf{x}^i)} p(\mathbf{x}^i|\mathbf{w}^j)/k\,.$$

The cost function of RSLVQ is given by the quotient

$$E_{\mathrm{RSLVQ}} = \log \prod_i \frac{p(\mathbf{x}^i, c(\mathbf{x}^i))}{p(\mathbf{x}^i)} = \sum_i \log \frac{p(\mathbf{x}^i, c(\mathbf{x}^i))}{p(\mathbf{x}^i)}$$

Considering gradients, we obtain the adaptation rule for every prototype $\mathbf{w}^j$ given a training point $\mathbf{x}^i$

$$\Delta\mathbf{w}^j \sim -\frac{1}{2\sigma^2} \cdot \left(\frac{p(\mathbf{x}^i|\mathbf{w}^j)}{\sum_{j:c(\mathbf{w}^j)=c(\mathbf{x}^i)} p(\mathbf{x}^i|\mathbf{w}^j)} - \frac{p(\mathbf{x}^i|\mathbf{w}^j)}{\sum_j p(\mathbf{x}^i|\mathbf{w}^j)} \right) \cdot \nabla_{\mathbf{w}^j} d(\mathbf{x}^i, \mathbf{w}^j)$$

if $c(\mathbf{x}^i) = c(\mathbf{w}^j)$ and

$$\Delta\mathbf{w}^j \sim \frac{1}{2\sigma^2} \cdot \frac{p(\mathbf{x}^i|\mathbf{w}^j)}{\sum_j p(\mathbf{x}^i|\mathbf{w}^j)} \cdot \nabla_{\mathbf{w}^j} d(\mathbf{x}^i, \mathbf{w}^j)$$

if $c(\mathbf{x}^i) \neq c(\mathbf{w}^j)$. Obviously, the scaling factors can be interpreted as soft assignments of the data to corrsponding prototypes. The choice of an appropriate parameter σ can critically influence the overall behavior and the quality of the technique, see e.g. [4, 8] for comparisons of GLVQ and RSLVQ and ways to automatically determine σ based on given data.

2 Dissimilarity data

In many application domains, data are becoming more and more complex, and data are often addressed by a dedicated dissimilarity measure which respects the structural form of the data and describes intrinsic relations between them. Prototype-based algorithms such as GLVQ and RSLVQ are restricted to Euclidean space, so that they are not suitable for this kind of more general data formats. Thus, here we extend GLVQ and RSLVQ to relational variants by means of an implicit reference to a pseudo-Euclidean space of data.

Assume that data $\mathbf{x}^i$ are given as pairwise dissimilarities $d_{ij} = d(\mathbf{x}^i, \mathbf{x}^j)$. D refers to the corresponding dissimilarity matrix. We assume D is symmetric and diagonal is

zero, i.e. $d_{ij} = d_{ji}$, $d_{ii} = 0$. However, we do not require that d refers to a Euclidean data space, i.e. D does not need to be embeddable in Euclidean space, nor does it need to fulfill the conditions of a metric.

As argued in [9, 1], every such set of data points can be embedded in a so-called pseudo-Euclidean vector space the dimensionality of which is limited by the number of given points. A pseudo-Euclidean vector space is a real-vector space equipped with the bilinear form $\langle \mathbf{x}, \mathbf{y} \rangle_{p,q} = \mathbf{x}_{p,q}^t \mathbf{y}$ where $I_{p,q}$ is a diagonal matrix with p entries 1 and q entries -1. The tuple (p, q) is also referred to as the signature of the space, and the value q determines in how far the standard Euclidean norm has to be corrected by negative eigenvalues to arrive at the given dissimilarity measure. The data set is Euclidean if and only if $q = 0$. For a given matrix D, the corresponding pseudo-Euclidean embedding can be computed by means of an eigenvalue decomposition of the related Gram matrix, which is an $\mathcal{O}(N^3)$ operation. It yields explicit vectors $\mathbf{x}^i$ such that $d_{ij} = \langle \mathbf{x}^i - \mathbf{x}^j, \mathbf{x}^i - \mathbf{x}^j \rangle_{p,q}$ holds for every pair of data points.

Note that vector operations can be naturally transferred to pseudo-Euclidean space, i.e. we can define prototypes as linear combinations of data in this space. Hence we can perform techniques such as GLVQ explicitly in pseudo-Euclidean space since it relies on vector operations only. One problem of this explicit transfer is given by the computational complexity of the initial embedding, on the one hand, and the fact that out-of-sample extensions to new data points characterized by pairwise dissimilarities are not immediate.

Because of this fact, we are interested in efficient techniques which implicitly refer to such embeddings only. As a side product, such algorithms are invariant to coordinate transforms in pseudo-Euclidean space, rather they depend on the pairwise dissimilarities only instead of the chosen embedding. The key assumption is to restrict prototype positions to linear combination of data points of the form

$$\mathbf{w}^j = \sum_i \alpha_{ji} \mathbf{x}^i \text{ with } \sum_i \alpha_{ji} = 1 \,.$$

Since prototypes are located at representative points in the data space, it is a reasonable assumption to restrict prototypes to the affine subspace spanned by the given data points. In this case, dissimilarities can be computed implicitly by means of the formula

$$d(\mathbf{x}^i, \mathbf{w}^j) = [D \cdot \alpha_j]_i - \frac{1}{2} \cdot \alpha_j^t D \alpha_j$$

where $\alpha_j = (\alpha_{j1}, \ldots, \alpha_{jn})$ refers to the vector of coefficients describing the prototype $\mathbf{w}^j$ implicitly, as shown in [1].

This observation constitutes the key to transfer GLVQ and RSLVQ to relational data without an explicit embedding in pseudo-Euclidean space. Prototype $\mathbf{w}^j$ is represented implicitly by means of the coefficient vectors α_j. Then, we can use the equivalent characterization of distances in the GLVQ and RSVLQ cost function leading to the

costs of relational GLVQ (RGLVQ) and relational RSLVG (RSLVQ), respectively:

$$E_{\mathrm{RGLVQ}} = \sum_i \Phi \left(\frac{[D\alpha^+]_i - \frac{1}{2} \cdot (\alpha^+)^t D \alpha^+ - [D\alpha^-]_i + \frac{1}{2} \cdot (\alpha^-)^t D \alpha^-}{[D\alpha^+]_i - \frac{1}{2} \cdot (\alpha^+)^t D \alpha^+ + [D\alpha^-]_i - \frac{1}{2} \cdot (\alpha^-)^t D \alpha^-} \right) ,$$

where as before the closest correct and wrong prototype are referred to, corresponding to the coefficients α^+ and α^-, respectively. A stochastic gradient descent leads to adaptation rules for the coefficients α^+ and α^- in relational GLVQ: component k of these vectors is adapted as

$$\begin{aligned} \Delta\alpha_k^+ &\sim -\Phi'(\mu(\mathbf{x}^i)) \cdot \mu^+(\mathbf{x}^i) \cdot \frac{\partial \left([D\alpha^+]_i - \frac{1}{2} \cdot (\alpha^+)^t D \alpha^+ \right)}{\partial \alpha_k^+} \\ \Delta\alpha_k^- &\sim \Phi'(\mu(\mathbf{x}^i)) \cdot \mu^-(\mathbf{x}^i) \cdot \frac{\partial \left([D\alpha^-]_i - \frac{1}{2} \cdot (\alpha^-)^t D \alpha^- \right)}{\partial \alpha_k^-} \end{aligned}$$

where $\mu(\mathbf{x}^i)$, $\mu^+(\mathbf{x}^i)$, and $\mu^-(\mathbf{x}^i)$ are as above. The partial derivative yields

$$\frac{\partial \left([D\alpha_j]_i - \frac{1}{2} \cdot \alpha_j^t D \alpha_j \right)}{\partial \alpha_{jk}} = d_{ik} - \sum_l d_{lk} \alpha_{jl}$$

Similarly,

$$E_{\mathrm{RRSLVQ}} = \sum_i \log \frac{\sum_{\alpha_j : c(\alpha_j) = c(\mathbf{x}^i)} p(\mathbf{x}^i | \alpha_j)/k}{\sum_{\alpha_j} p(\mathbf{x}^i | \alpha_j)/k}$$

where

$$p(\mathbf{x}^i | \alpha_j) = K \cdot \exp \left(- \left([D\alpha_j]_i - \frac{1}{2} \cdot \alpha_j^t D \alpha_j \right) / 2\sigma^2 \right)$$

A stochastic gradient descent leads to the adaptation rule

$$\Delta\alpha_{jk} \sim -\frac{1}{2\sigma^2} \cdot \left(\frac{p(\mathbf{x}^i | \alpha_j)}{\sum_{j : c(\alpha_j) = c(\mathbf{x}^i)} p(\mathbf{x}^i | \alpha_j)} - \frac{p(\mathbf{x}^i | \alpha_j)}{\sum_j p(\mathbf{x}^i | \alpha_j)} \right) \cdot \frac{\partial \left([D\alpha_j]_i - \frac{1}{2} \alpha_j^t D \alpha_j \right)}{\partial \alpha_{jk}}$$

if $c(\mathbf{x}^i) = c(\alpha_j)$ and

$$\Delta\alpha_{jk} \sim \frac{1}{2\sigma^2} \cdot \frac{p(\mathbf{x}^i | \alpha_j)}{\sum_j p(\mathbf{x}^i | \alpha_j)} \cdot \frac{\partial \left([D\alpha_j]_i - \frac{1}{2} \alpha_j^t D \alpha_j \right)}{\partial \alpha_{jk}}$$

if $c(\mathbf{x}^i) \neq c(\alpha_j)$.

After every adaptation step, normalization takes place to guarantee $\sum_i \alpha_{ji} = 1$. This way, a learning algorithm which adapts prototypes in a supervised manner similar to GLVQ or RSLVQ, respectively, is given for general dissimilarity data, whereby prototypes are implicitly embedded in pseudo-Euclidean space.

The prototypes are initialized as random vectors, i.e we initialize α_{ij} with small random values such that the sum is one. It is possible to take class information into account by setting all α_{ij} to zero which do not correspond to the class of the prototype. Further, it is possible to arrive at a good initialization which avoids local optima and also a good initial guess of the number of prototypes per class by initializing the prototype positions with relational prototype vectors trained by means of an unsupervised relational vector quantization technique such as relational neural gas [1]. The prototype labels can then be determined based on their receptive fields before adapting the initial decision boundaries by means of supervised learning vector quantization.

An extension of the classification to new data is immediate based on an observation made in [1]: given a novel data point $\mathbf{x}$ characterized by its pairwise dissimilarities $D(\mathbf{x})$ to the data used for training, the dissimilarity of $\mathbf{x}$ to a prototype represented by α_j is $d(\mathbf{x}, \mathbf{w}^j) = D(\mathbf{x})^t \cdot \alpha_j - \frac{1}{2} \cdot \alpha_j^t D \alpha_j$.

3 Acceleration by Nyström

In addition, the technique depends on the full dissimilarity matrix and thus displays quadratic time and space complexity. Depending on the chosen dissimilarity, the main computational bottleneck is given by the computation of the dissimilarity matrix itself. The Nyström approximation as introduced in [13] allows an efficient approximation of a kernel matrix by a low rank matrix. This approximation can directly be transferred to dissimilarity data. The basic principle is to pick M representative landmarks from N data points which induce the rectangular sub-matrix $D_{M,N}$ of dissimilarities of data points and landmarks. This matrix is of linear size, assuming M is fixed. The full matrix can be approximated in an optimum way in the form

$$D \approx D_{M,N}^T D_{M,M}^{-1} D_{M,N}$$

where $D_{M,M}$ is the rectangular sub-matrix of D. Its computation is $\mathcal{O}(M^3)$ instead of $\mathcal{O}(N^2)$ for the full matrix D. The approximation is exact if M corresponds to the rank of D. For 10% landmarks, this leads to a speed-up factor 50, i.e. the computation of an approximated dissimilarity matrix for $11,000$ sequences can be computed in less than two hours instead of eight days. Note that the Nyström approximation can be integrated into relational GLVQ in such a way that the training complexity is linear. We refer to results obtained by a Nyström approximation by the superscript RGLVQ$^\nu$ whereby we pick 10% of the data set as landmarks per default.

4 Experiments

We evaluate the algorithms for several benchmark data sets where data are characterized by pairwise dissimilarities. On the one hand, we consider the data sets used also in [2]. The article [2] investigates the possibility to deal with similarity/dissimilarity data

which is non-Euclidean with the SVM. Since the corresponding Gram matrix is not positive semidefinite, according preprocessing steps have to be done which make the SVM well defined. These steps can change the spectrum of the Gram matrix or they can treat the dissimilarity values as feature vectors which can be processed by means of a standard kernel. Data are:

1. Amazon47 consisting of 204 data points from 47 classes, representing books and their similarity based on customer preferences.
2. Aural Sonar consists of 100 signals with two classes (target of interest/clutter), consisting of sonar signals with dissimilarity measures according to an ad hoc classification of humans.
3. Face Recognition consists of 945 samples with 139 classes, representing faces of people, compared by the cosine similarity.
4. Patrol consists of 241 data points from 8 classes, corresponding to seven patrol units (and non-existing persons, respectively). Similarities are based on clusters named by people.
5. Protein consists of 213 data from 4 classes, representing globin proteins compared by an evolutionary measure.
6. Voting contains 435 samples in 2 classes, representing categorical data compared based on the value difference metric.

As pointed out in [2], these matrices cover a diverse range of different characteristics such that they constitute a well suited test bench to evaluate the performance of algorithms for similarities/dissimilarities. In addition, we consider three data sets representing typical application scenarios:

1. The Cat Cortex data set consists of 65 data points from 5 classes. The data originate from anatomic studies of cats' brains. The dissimilarity matrix displays the connection strength between 65 cortical areas. For our purposes, a preprocessed version as presented in [10] was used.
2. The Copenhagen Chromosomes data set constitutes a benchmark from cytogenetics [16]. A set of 4,200 human chromosomes from 21 classes (the autosomal chromosomes) are represented by grey-valued images. These are transferred to strings measuring the thickness of their silhouettes. These strings are compared using edit distance [7].
3. The Vibrio data set consists of 1,100 samples of vibrio bacteria populations characterized by mass spectra. The spectra contain approx. 42,000 mass positions. The full data set consists of 49 classes of vibrio-sub-species. The mass spectra are preprocessed with a standard workflow using the BioTyper software [6]. As usual, mass spectra display strong functional characteristics due to the dependency of subsequent masses, such that problem adapted similarities such as described in [5, 6] are beneficial. In our case, similarities are calculated using a specific similarity measure as provided by the BioTyper software[6].

	RGLVQ	RGLVQ$^{\nu}$	**RRSLVQ**	**best SVM [2]**	**#Prototypes**
Amazon47	0.810(0.014)	0.814(0.011)	0.830(0.016)	0.82	94
Aural Sonar	0.884(0.016)	0.864(0.008)	0.848(0.017)	0.87	10
Face Rec.	0.964(0.002)	0.864(0.002)	0.964(0.002)	0.96	139
Patrol	0.841(0.014)	0.856(0.015)	0.850(0.011)	0.88	24
Protein	0.924(0.019)	0.558(0.028)	0.530(0.011)	0.97	20
Voting	0.946(0.005)	0.905(0.003)	0.623(0.014)	0.95	20
Cat Cortex	0.930(0.010)	0.922(0.023)	0.941(0.011)	n.d.	12
Vibrio	1.000(0.000)	0.992(0.001)	0.941(0.077)	n.d.	49
Chromosome	0.927(0.002)	0.782(0.004)	0.795(0.009)	n.d.	63

Table 1: Results of prototype based classification in comparison to SVM for diverse dissimilarity data sets. The classification accuracy obtained in a repeated cross-validation is reported, the standard deviation is given in parenthesis.

Since some of these matrices correspond to similarities rather than dissimilarities, we use standard preprocessing as presented in [14]. For every data set, a number of prototypes which mirrors the number of classes was used, representing every class by only few prototypes relating to the choices as taken in [1], see Tab. 1. The evaluation of the results is done by means of the classification accuracy as evaluated on the test set in a ten fold repeated cross-validation with ten repeats. The results are reported in Tab. 1. In addition, we report the best results obtained by SVM after diverse preprocessing techniques [2].

Interestingly, in most cases, results which are comparable to the best SVM as reported in [2] can be found, whereby making preprocessing as done in [2] superfluous. Further, unlike for SVM which is based on support vectors in the data set, solutions are represented as typical prototypes. Nyström leads to improved speed, specially sensible on Chromosomes data, and the results are reduced but in most case are comparable to RGLVQ.

5 Conclusions

We have presented an extension of prototype-based techniques to general possibly non-Euclidean data sets by means of an implicit embedding in pseudo-Euclidean data space and a corresponding extension of the cost function of GLVQ and RSLVQ to this setting. As a result, a very powerful learning algorithm can be derived which, in most cases, achieves results which are comparable to SVM but without the necessity of according preprocessing since relational LVQ can directly deal with possibly non-Euclidean data

whereas SVM requires a positive semidefinite Gram matrix. A linear acceleration technique by means of the Nyström approximation for dissimilarity data is also integrated into RGLVQ which, in most cases, achieves comparable results to original RGLVQ.

References

[1] B. Hammer and A. Hasenfuss. Topographic mapping of large dissimilarity datasets. *Neural Computation*, 22(9):2229–2284, 2010.

[2] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti. Similarity-based classification: Concepts and algorithms. *Journal of Machine Learning Research*, 10(Mar):747–776, 2009.

[3] P. Schneider, M. Biehl, and B. Hammer. Adaptive relevance matrices in adaptive relevance matrices in adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561, 2009.

[4] P. Schneider, M. Biehl, and B. Hammer. Distance learning in discriminative vector quantization. *Neural Computation*, 21:2942–2969, 2009.

[5] S. B. Barbuddhe, T. Maier, G. Schwarz, M. Kostrzewa, H. Hof, E. Domann, T. Chakraborty, and T. Hain. Rapid identification and typing of listeria species by matrix-assisted laser desorption ionization-time of flight mass spectrometry. *Applied and Environmental Microbiology*, 74(17):5402–5407, 2008.

[6] T. Maier, S. Klebel, U. Renner, and M. Kostrzewa. Fast and reliable maldi-tof ms–based microorganism identification. *Nature Methods*, (3), 2006.

[7] M. Neuhaus and H. Bunke. Edit distance based kernel functions for structural pattern classification. *Pattern Recognition*, 39(10):1852–1863, 2006.

[8] Sambu Seo and Klaus Obermayer. Dynamic hyperparameter scaling method for lvq algorithms. In *IJCNN*, pages 3196–3203, 2006.

[9] Elzbieta Pekalska and Robert P.W. Duin. *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications*. World Scientific, 2005.

[10] B. Haasdonk and C. Bahlmann. Learning with distance substitution kernels. *Pattern Recognition - Proc. of the 26th DAGM Symposium*, 2004.

[11] S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15(7):1589–1604, 2003.

[12] T. Kohonen, editor. *Self-Organizing Maps*. Springer-Verlag New York, Inc., 3rd edition, 2001.

[13] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

[14] B. Hammer. *Learning with Recurrent Neural Networks*, volume 254 of *Lecture Notes in Control and Information Sciences*. Springer, 2000.

[15] A. Sato and K. Yamada. Generalized learning vector quantization. In M. C. Mozer, D. S. Touretzky, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*, pages 423–429, Cambridge, MA, USA, 1996. MIT Press.

[16] C. Lundsteen, J-Phillip, and E. Granum. Quantitative analysis of 6985 digitized trypsin g-banded human metaphase chromosomes. *Clinical Genetics*, 18:355–370, 1980.