Sparse Kernelized Vector Quantization with Local Dependencies

Frank-Michael Schleif

***Abstract*— Clustering approaches are very important methods to analyze data sets in an initial unsupervised setting. Traditionally many clustering approaches assume data points to be independent. Here we present a method to make use of local dependencies to improve clustering under guaranteed distortions. Such local dependencies are very common for data generated by imaging technologies with an underlying topographic support of the measured data. We provide experimental results on artificial and real world data of clustering tasks.**

I. INTRODUCTION

The dramatic growth in data generating applications and measurement techniques has created many high-volume and multi-dimensional data sets. Most of them are stored digitally and need to be efficiently analyzed to be of use. Clustering methods are very important in this setting and have been extensively studied in the last decades [6]. For most of these approaches you need to specify the number of clusters in advance and the obtained models are hard to interpret. Additionally they ignore potential available meta-knowledge like some kind of data dependencies. Novel imaging technologies for high dimensional measurements like in satellite remote sensing of life science imaging generate such large sets of data. Additionally for such type of data the labeling of individual measurement points is costly. Clusterings are therefore used in such settings to obtain full labellings of the data sets based on the cluster assignments. Traditionally approaches like single-linkage clustering or k-means are employed but also more novel hierarchical methods adapted for large data processing are used [16]. Most of these approaches consider the data points to be independent and do not or only minor integrate meta information of the data or the underlying topographic grid structure of these high-dimensional measurements. This information can improve the clustering because we expect that neighbored points on the grid are dependent and hence potentially similar in the data-space. Beside of the grid dependencies also other dependencies between datapoints are interesting, because clustering of high dimensional data is complex and useful constraints are desired.

Recently in [18] a supervised learning method has been proposed, to efficiently process data with local dependencies. Here we consider an unsupervised setting with local dependencies where potentially only some of the data points are labeled. Data of such type can be observed in the field of satellite remote sensing [15], medical imaging [18], [3] or in a simpler setting for standard imaging. Beside a good clustering also compact, sparse cluster models are of interest.

In this paper, we propose a new algorithm based on a kernelized vector quantizer algorithm [17] (KVQ) which will be adapted and extended to provide sparse models and to deal with local data dependencies. The idea is to modify the original problem definition by additional constraints and to replace the cost function by a more appropriate setting. Our model is formed by a number of prototypical vectors identified from the dataspace.

The rest of the paper is organized as follows. In section II we give preliminary settings used throughout the paper and review some kernel based prototype clustering approaches. Subsequently we introduce our kernel vector quantizer with local dependencies (LKVQ). In section IV we show the efficiency of the novel approach by experiments on artificial and real life data. Finally, we conclude with a discussion of the results and open issues.

II. PRELIMINARIES

We consider a clustering method C providing a cluster model $\hat{f}$ with a number of so called codebook vectors or prototypes summarized in a codebook W and meta-parameters ζ. Prototypes are members of the original dataspace and allow easier interpretation and analysis of the obtained model than by use of alternative clustering approaches.

We consider a dataset D with vectors $\vec{v} \in V$ with $V \subset \mathbb{R}^d$ and d the number of dimensions. The number of samples is given as $N = |V|$. Further we introduce prototypes $\vec{w}$ living in the same space as the dataspace. Prototypes are typically indexed by i and data-points by j. KVQ and LKVQ employ a kernel based distance measure ϕ mapping the data similarities of some distance measure D to $\{0, 1\}$ as detailed subsequently.

There are some prototype methods which are kernel based, now briefly reviewed for later comparison with KVQ and our method.

A. Kernel Neural Gas

Kernel neural gas (KNG) as introduced in [13] is a kernelized variant of the classical Neural Gas (NG) algorithm [9] with an optimization scheme employing neighborhood cooperation. Further KNG assumes the availability of a similarity matrix or Gram matrix S with entries s_{ij} characterizing the similarity of points numbered i and j. This should be positive semi-definite to allow an interpretation by means of an embedding in an appropriate Hilbert space, i.e. $s_{ij} = \Phi(\vec{v}_i)^t \Phi(\vec{v}_j)$ for some feature function Φ. It can, however,

Frank-Michael Schleif is with the Department of Technology, University of Bielefeld, Universitätsstrasse 21-23, 33615 Bielefeld, Germany, (email: fschleif@techfak.uni-bielefeld.de).

algorithmically be applied to general similarity matrices. The key idea is to identify prototypes with linear combinations in the high dimensional feature space

$$\vec{w}_j = \sum_l \alpha_{jl} \Phi(\vec{v}_l) \tag{1}$$

with α_{jl} as scaling coefficients. Then, squared distances can be computed based on the Gram matrix as follows:

$$d(\Phi(\vec{v}_i), \vec{w}_j) = s_{ii} - 2\sum_l \alpha_{jl} s_{il} + \sum_{ll'} \alpha_{jl}\alpha_{jl'} s_{ll'} \tag{2}$$

In [13] this approach is used to kernelize the original NG within a gradient descent optimization, employing the kernel trick.

B. Relational Neural Gas

Relational neural gas (RNG) as introduced in [5] assumes that a symmetric dissimilarity matrix D with entries d_{ij} describing pairwise dissimilarities of data is available. In principle, it is very similar to KNG with respect to the general idea. There are two differences: RNG is based on dissimilarities rather than similarities, and it solves the resulting cost function using a batch optimization with quadratic convergence as compared to a stochastic gradient descent.

As shown in [12], there always exists a so-called pseudo-Euclidean embedding of a given set of points characterized by pairwise symmetric dissimilarities by means of a mapping Φ, i.e. a real vector space and a symmetric bi-linear form (with probably negative eigenvalues) such that the dissimilarities are obtained by means of this bi-linear form. As before, prototypes are restricted to linear combinations

$$\vec{w}_j = \sum_l \alpha_{jl} \Phi(\vec{v}_l) \text{ with } \sum_l \alpha_{jl} = 1 \tag{3}$$

We can put the restriction that the sum always leads to one. Under this constraint, one can see that dissimilarities can be computed by means of the formula

$$d(\Phi(\vec{v}_i), \vec{w}_j) = [D^t \alpha_j]_i - \frac{1}{2} \cdot \alpha_j^t \tag{4}$$

where $[\cdot]_i$ refers to component i of the vector. This allows a direct transfer of batch NG to general dissimilarities by the following iterations

$$\text{determine } k_{ij} \text{ based on } d(\Phi(\vec{v}_i), \vec{w}_j) \tag{5}$$

$$\alpha_{jl} := h_\sigma(k_{lj}) / \sum_l h_\sigma(k_{lj}) \tag{6}$$

where $h_\sigma(t) = \exp(-t/\sigma)$ is a neighborhood function which exponentially scales the neighborhood range, and k_{ij} denotes the rank of prototype $\vec{w}_j$ with respect to $\vec{v}_i$, i.e. the number of prototypes $\vec{w}_k$ with $k \neq j$ which are closer to $\vec{v}_i$ as measured by some distance measure D. This algorithm can be interpreted as neural gas in pseudo Euclidean space for every symmetric dissimilarity matrix D. If negative eigenvalues are present, however, convergence is not always guaranteed, all though it can mostly be observed in practice [5].

III. MODEL

Our objective is to summarize a larger set of data by a vector quantization (VQ) approach [4]. We represent a set of data vectors $\{\vec{v}_1, \ldots, \vec{v}_l\} \in V$ by a reduced number of m prototypes or codebook vectors $\{\vec{w_1}, \ldots, \vec{w_m}\} \in W$ with the same dimensionality as the original dataspace V and the average distortion is minimized according to a winner takes all (wta) scheme [7]. For an l_2 metric we consider:

$$E_{VQ} = \sum_{i=1}^{l} ||\vec{v_i} - \vec{w_j}||^2 \tag{7}$$

with $\vec{w_j} = \underset{\vec{w_k}}{argmin} ||\vec{v_i} - \vec{w_k}||^2 = wta(\vec{v_i})$. Following the concept of [17] the goal is to find a minimal codebook minimizing (7) with the constrained that the prototypes are part of V and the distance of the prototypes to datapoints limited by a radius parameter $\mathcal{R}$, guaranteeing a maximal level of distortion. The problem specific radius parameter $\mathcal{R}$ has to be provided by the user. Roughly, $\mathcal{R}$ defines the number of cluster in the data, but is typically more easier accessible from knowledge of typical data similarities than the number of clusters. This can be efficiently formulated as a linear programing problem. While this approach is very promising the solutions of the approach in [17] are not very sparse and limited to rather small sets V. As a further criticism of KVQ the prototypes are not necessary representative. Actually the prototype solutions of KVQ may still contain points which are borderline points of the solution, they are valid but not prototypical and the solution may still not be very sparse due to the considered linear, real weighted optimization problem. To overcome this we modify the considered cost function and formulate the problem as a binary optimization problem. Further, taking the local topology into account, we are able to provide approximate solutions also for very large sets V in reasonable time and can potentially improve the labeling of unknown points.

A. Clustering via binary linear programming

Suppose we are given data

$$\{\vec{v_1}, \ldots, \vec{v_N}\} \in V$$

where V is our data space equipped with an additional dependency relation $c_{i,k} \in [0, 1]$ for each pair $(\vec{v_i}, \vec{v_k})$ with $c_{i,k} \in C^{N \times N}$ and a distance measure $\mathcal{D}$ defined over V and potentially a further distance measure $\mathcal{D}'$ defined on V generating C. Following [17] we consider a kernel k:

$$k : V \times V \to \mathcal{R}$$

in particular

$$k(\vec{v}, \vec{v'}) = 1_{(\vec{v},\vec{v'}) \in V \times V : \mathcal{D}(\vec{v},\vec{v'}) \leq \mathcal{R}} \tag{8}$$

we consider also the empirical kernel map

$$\phi_l(\vec{v}) = (k(\vec{v_1}, \vec{v}), \ldots, k(\vec{v_l}, \vec{v}))$$

We are now looking for a vector $\vec{x} \in \mathbb{R}^l$ such that:

$$x^\top \phi_l(\vec{v}) > 0 \tag{9}$$

is true $\forall\, i = 1, \ldots, l$. Then each point $\vec{v_i}$ is within a distance $\mathcal{R}$ of some prototype $\vec{w_j}$ with a positive weight $x_j > 0$ [17]. The obtained $\vec{w_j}$ define a codebook providing an approximation of V up-to an error $\mathcal{R}$ measured in $\mathbb{R}^d$. To avoid trivial solutions the optimization problem is reformulated in [17] to:

$$\min_{x \in \mathbb{R}} \quad ||x||_1 \tag{10}$$

$$s.t. \quad x^\top \phi_l(\vec{v_i}) \geq 1 \tag{11}$$

The original problem in Eq. (10) considers all points $\vec{v}$ equally as long as the constraints are fulfilled. Hence for equivalent $\{\vec{v_k}, \ldots, \vec{v_m}\}$ the weights with $\{x_k, \ldots, x_m\}$ are not necessary sparse and the final prototype becomes arbitrary. To overcome this problem we reformulate the optimization problem as follows:

$$\min_{x \in \mathbb{R}} \quad ||f^\top x||_1 \tag{12}$$

$$s.t. \quad x^\top \phi_l(\vec{v_i}) \geq 1 \tag{13}$$

$$x \in [0, 1] \tag{14}$$

leading to a binary, integer optimization problem[1] and a weighting of each $\vec{v_k}$ by a factor (cost) f_k. To calculate f_k we consider the dependencies $C(k, \cdot)$ using radius R' and calculate the median distance of $\vec{v_k}$ to all points $\vec{v_m}$ with $C(k, m) \geq 0$ with a fixed offset of 1 and a self dependence of 1. If no dependencies are given we may still able to provide some meaningful f by choosing f as e.g. the median distance with respect to the data or a randomly sampled subset thereof.

This definition ensures on the one hand-side maximal sparsity of the model but also that the prototypes are somehow typical (measured by the median) with respect to dependent points measured with distance $\mathcal{D}'$.

For our experiments we assume that all data are located or measured on a grid. In this case the local dependencies C can be defined e.g. on this given grid structure. Further, using this additional information we can split the optimization problems into patches such that Eq. (12) is calculated only on a part of the data, e.g. a band of up-to some 1000 points of V defined by the grid[2]. This leads to a very efficient optimization problem also in case of very large data sets. The combination of all these local solutions to the final solution is still optimal but maybe slightly over-defined. The solution could be more sparse but for our experiments we found that this is no significant problem. The final model can be used to assign new points $\vec{v_k}$ to its nearest prototypes according to the wta scheme. If a labeling is available the datapoints $\vec{v_k}$ can be labeled according to the labeling of the prototypes they belong to. Alternatively and similar as within [18] we may take the local dependencies into account to re-weight the original labeling. If the closest prototype of $\vec{v_k}$ is labeled

[1]Linear programming is solvable in polynomial time, which is not sure for integer programming, so it is more demanding, but in our studies the calculation times were comparable short.

[2]In case of grid structured data it would of course also be possible to split the dataset in advance and combine the solutions manually.

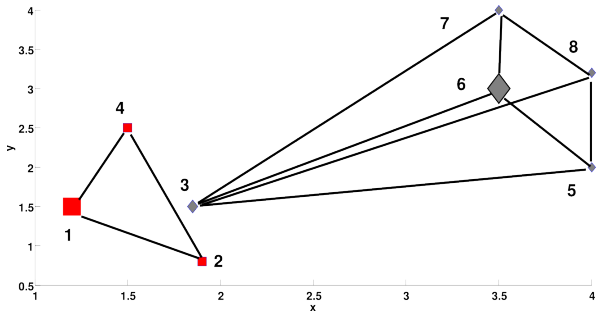

Fig. 1. The plot shows an output of LKVQ ($R = 1.5, R' = 0.5$) for a simple example explaining the dependency concept. Two clusters of rectangles and diamonds are shown with lines connecting the dependent entries. Using standard KVQ the prototypes are identified as the points $\{(1.2, 1.5); (1.8, 1.5)\}$ and $\{(3.5, 3); (4, 3.2)\}$. Using LKVQ the prototypes are $(1.2, 1.5)$ and $(3.5, 3)$ - highlighted by larger symbols. Additionally using $\theta = 0.7$ LKVQ could correct the label of point $(1.8, 1.5)$ to diamond.

with L_j, we label $\vec{v_k}$ in accordance to the labeling scheme $\mathcal{L}$:

$$\mathcal{L}(\vec{w_j}) = L_j \quad \text{for prototypes} \tag{15}$$

$$\mathcal{L}(\vec{v_k}) = \theta \cdot L_j + (1 - \theta) \cdot \sum_t \frac{\mathcal{L}(wta(\vec{v_t}))}{C(k, t)^{-1}} \tag{16}$$

and θ a parameter typically $\theta = 0.7$. Using this approach a *wrong* labeling of a point can be potentially corrected by taking the labeling of the dependent neighbors into account. It should be noted again that by the modification of the cost function this dependency relation is also included on the optimization process. The effect of the local dependency optimization is schematically shown in Figure 1. Exemplary we consider two clusters shown in a 2D data space. The data have additionally a dependency indicated by lines between the points. Now assume that points $\{3, 5, 6, 7, 8\}$ are dependent, e.g. are neighbored on the grid, and also the points $\{1, 2, 4\}$ are dependent. In the dataspace these two sets are however not perfectly disjunct because point 3 is close to the cluster of the points $\{1, 2, 4\}$. Without dependencies the point 3 would be assigned to the left (rectangle symbols) cluster in Figure 1 or the point itself would become the cluster center. Using the dependencies in the clustering an alternative prototype has been identified for the cluster of the points $\{1, 2, 4\}$ and the point 3 got the labeling of cluster 2 (diamond), although it has been assigned to cluster 1 due to its data space proximity.

IV. EXPERIMENTS

A. Artificial data

Initially we repeated the experiment similar as given in [17] for data spread on a rectangular $2D$ grid using the Euclidean distance and a radius $R = 0.2$. Both approaches, the KVQ from [17] and our model lead to similar results, see Figure 2. Here for LKVQ we have split the problem into patches using the locality of the data and we observe slightly more prototypes than for KVQ as expected. It should be mentioned that for KVQ the implementation given in *Spider*

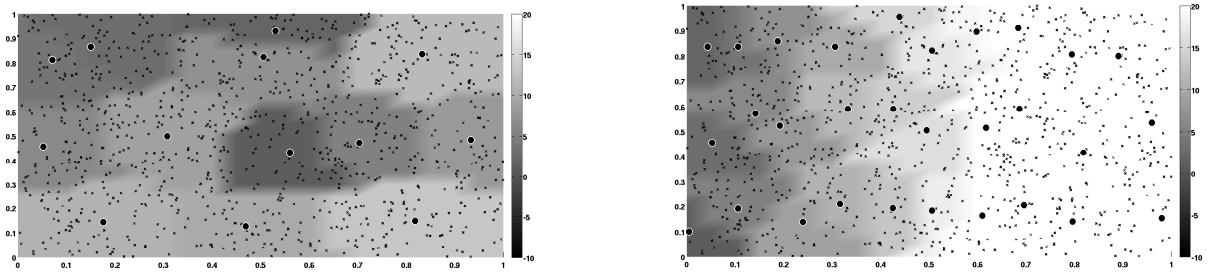

Fig. 2. Visualization of the prototype models as obtained by KVQ (left) and LKVQ (right). The data are uniform Gaussian in $[0,1] \times [0,1]$ and a radius $R = 0.2$ was chosen. Both solutions are valid but the solution of LKVQ is over-defined due to the used patch concept. Prototypes are given as bold dots with a white circle. The receptive fields of the prototypes are indicated by different background shades.

[19] has in parts problems to find a valid solution for the optimization problem using different values for R, whereas LKVQ always finds a sparse solution for each patch. If the data set is not to large we can process the data as one single block using LKVQ. In Figure 3 the solutions for a checker board dataset with 16 clusters is depicted. Here we used $R = 30$ for the data and grid distances. We observe that the solution of KVQ is valid but not maximum sparse whereas for LKVQ the smallest valid solution with 16 prototypes, one per cluster, has been identified due to the binary linear problem. We also observe that the prototypes for LKVQ are closer to the center of the data clouds as expected due to the alternative cost function.

B. Influence of the radius R

The radius R has a direct impact on the number of generated clusters but also on the efficiency of the clustering algorithm. For very small R the number of clusters is increasing whereas for large R the number of clusters (NC) decreases until $NC = 1$. Beside of this effect also the computation time is sometimes effected. If R is chosen too large and the costs f are not equalized, the clusters may strongly overlap and the optimization problem becomes more and more complex.

To show this we consider a small example (without local dependencies) taken from [11] (subset n15,a45), known as the chicken data set. The task is to classify 446 silhouettes of chicken pieces into 5 categories (wing, back, drumstick, thigh and back, breast). The data silhouettes are represented as a string of the angles of consecutive tangential line pieces of length 20 including appropriate scaling. The strings are then compared using a (rotation invariant) edit distance. The results for a run of the chicken data with different R are shown in Figure 4. One clearly observe the mentioned effect and also that for some large R the runtime is significantly increased. This is caused because there exist multiple solutions with very similar costs. To overcome this problem one can add uniform noise $N(0,1)$ to the similarity matrix with exception of the main diagonal. Thereafter the effect is significantly less pronounced, in our experiments it could be reduced by a factor of 10 such that almost the original performance has been obtained, with the same accuracy as before. The chicken example also demonstrates the capability of LKVQ to deal with non-euclidean data.

C. Image encoding

As the second example we consider an image encoding approach using a standard image, because image coding provides an easily visualized task. We consider different radii leading to a different number of prototypes and compare the image quality by the standard PSNR value. Additional to KVQ we compare with the LBG algorithm. The results are shown in Figure 5 and 6. The original image consists of 256×256 monochrome pixels and has been preprocessed to 32×32 pixel with 64 dimensions each in the same manner as in [17]. We found that all approaches are able to achieve $PSNR > 30$ but the visual quality is different and also the number of necessary prototypes is significantly larger for LBG and KVQ. In the second plot LBG and LKVQ are compared with 529 prototypes. Again we observe a very good reconstruction quality for LBG and LKVQ but the PSNR for LKVQ is significantly better and artifacts in the face region of the LBG reconstruction are not present for LKVQ. KVQ shows the same artefact's in some image blocks as in the previous publication.

In Table I we show the results for different radii and compare the PSNR values. Due to the rather small number of sample points for the considered complex scenery the setting of R' has no significant effect for LKVQ and has been skipped.

Additionally we compare the results using the KNG algorithm and the RNG algorithm, both with Euclidean distance in Table II. It should be noted that for KNG and RNG the prototypes are not restricted to points from the dataspace.

D. Local dependencies - remote sensing application

The main subject of this research is to provide an efficient clustering approach taking local dependencies into account. Such problems occur more and more often e.g. in satellite remote sensing or for medical imaging technologies. In a real life example we consider a satellite remote sensing data set from the Colorado region (see Figure 7).

Airborne and satellite-borne remote sensing spectral images consist of an array of multi-dimensional vectors (spectra) assigned to particular spatial regions (pixel locations)

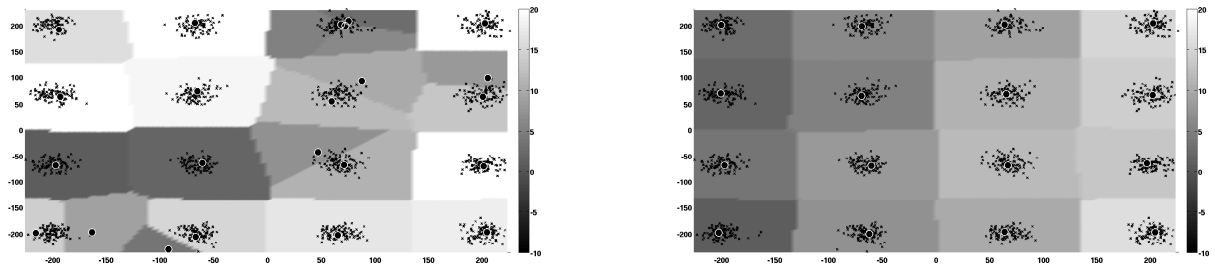

Fig. 3. Visualization of the prototype models as obtained by KVQ (left) and LKVQ (right) for a 4×4 checkerboard. The parameter $R = 30$ was chosen. Both solutions are valid but the solution of LKVQ has maximum sparseness and the fidelity of the prototype positions is better than for KVQ. Prototypes are given as bold dots with a white circle. The receptive fields of the prototypes are indicated by different background shades.

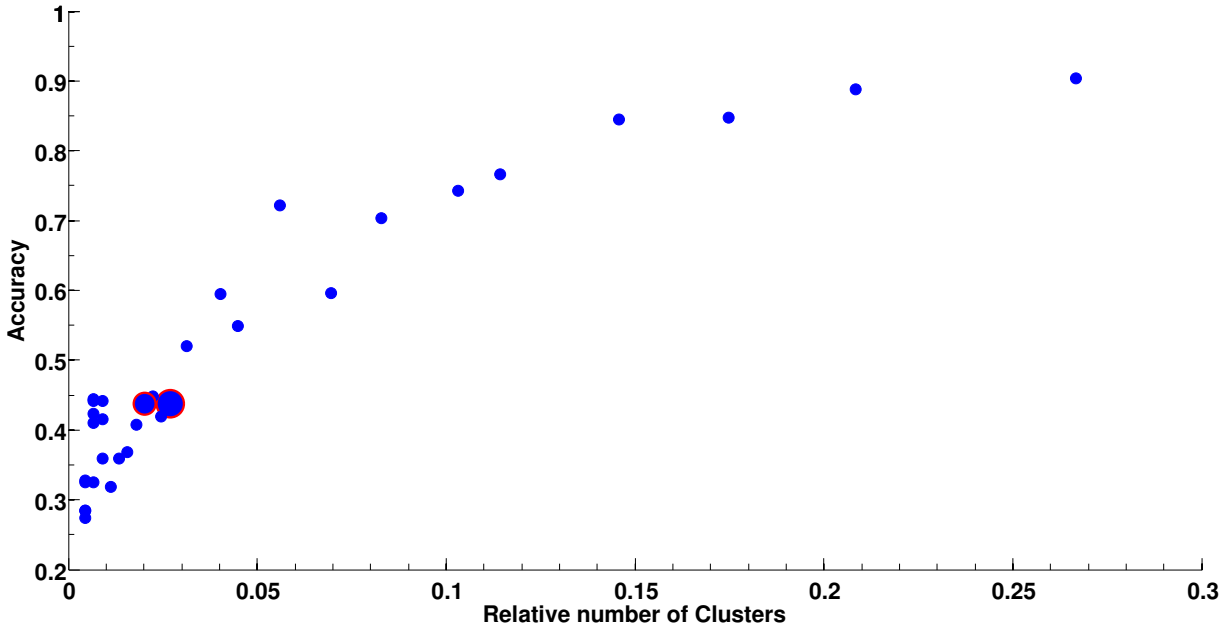

Fig. 4. The plot shows results of LKVQ varying R in a range of $0.1, 0.6 \ldots, 29.6$ for the chicken data set with a mean distance of 23. The x-axis depicts the relative number of cluster ($1.0\% \equiv 446$ points) and the y-axis the obtain recognition accuracy using a given labeling. The runtime of the LKVQ optimization is indicated by the size of the dots (great dots indicate long runtimes).

Image	Size	R	Prototypes	PSNR
Original	full	0	65536	-
LBG-Reconstr.	small	-	184	25
LBG-Reconstr.	medium	-	257	27
LBG-Reconstr.	large	-	529	32
KVQ-Reconstr.	small	300	891*	23
KVQ-Reconstr.	medium	200	204*	22
KVQ-Reconstr.	large	50	495	33
LKVQ-Reconstr.	small	300	184	27
LKVQ-Reconstr.	medium	200	257	30
LKVQ-Reconstr.	large	50	529	41

TABLE I

COMPRESSION MODELS USING LBG, KVQ AND LKVQ FOR DIFFERENT MODEL SIZES. FOR ENTRIES WITH * THE APPROACH DID NOT CONVERGE AND HAD BEEN STOPPED AFTER 10 ITERATIONS. ONE CAN CLEARLY OBSERVE THE LKVQ PERFORMED BEST FOR ALL MODEL SIZES.

reflecting the response of a spectral sensor at various wavelengths. A spectrum is a characteristic pattern that provides a clue to the surface material within the respective surface element. The utilization of these spectra includes areas

Fig. 5. Visualization of the image reconstruction LBG (left), KVQ (middle) and LKVQ (right). All reconstructions have a PSNR of $30 - 32$ but LBG requires 529 prototypes, KVQ 495 and LKVQ the lowest number of prototypes 257. The radius of LKVQ is $R = 200$ and for KVQ $R = 50$.

Fig. 6. Visualization of the image reconstruction LBG (left), LKVQ (right) each with 529 prototypes. The LBG reconstruction has a PSNR of 32 whereas the LKVQ reconstruction provides an almost perfect reconstruction with a PSNR of 41 and a radius $R = 50$.

Method	Prototypes	PSNR	LKVQ-PSNR
RNG	529	30.25	41
	257	25.29	30
	184	24.12	27
KNG	529	20.2	41
	257	19.9	30
	184	19.8	27

TABLE II

PSNR VALUES OF THE *man*-IMAGE-RECONSTRUCTIONS WITH DIFFERENT #PROTOTYPES FOR RNG AND KNG COMPARED TO LKVQ.

such as mineral exploration, land use, forestry, ecosystem management, assessment of natural hazards, water resources, environmental contamination, biomass and productivity; and many other activities of economic relevance [14].

Spectral images can formally be described as a matrix $\mathbf{S} = \vec{v}^{(x,y)}$, where $\vec{v}^{(x,y)} \in \mathbb{R}^{D_\mathcal{V}}$ is the vector (spectrum) at pixel location (x, y) here with $D_\mathcal{V} = 6$. The elements $v_i^{(x,y)}$, $i = 1 \ldots D_\mathcal{V}$ of spectrum $\vec{v}^{(x,y)}$ reflect the responses of a spectral sensor at a suite of wavelengths [1]. The spectrum is a characteristic fingerprint pattern that identifies the averaged content of the surface material within the area defined by pixel (x, y). Some sample spectra for different data classes are depicted in Figure 8. The data density $\mathcal{P}(\mathcal{V})$ may vary strongly within the data. Sections of the data space can be very densely populated while other parts may be extremely sparse, depending on the materials in the scene and on the spectral band-passes of the sensor. Therefore with standard clusterings it may easily happen that sparse data regions are omitted from the model leading to large errors for rare classes.

The considered image was taken very close to colorado springs using satellites of LANDSAT-TM type

The satellite produced pictures of the earth in 7 different

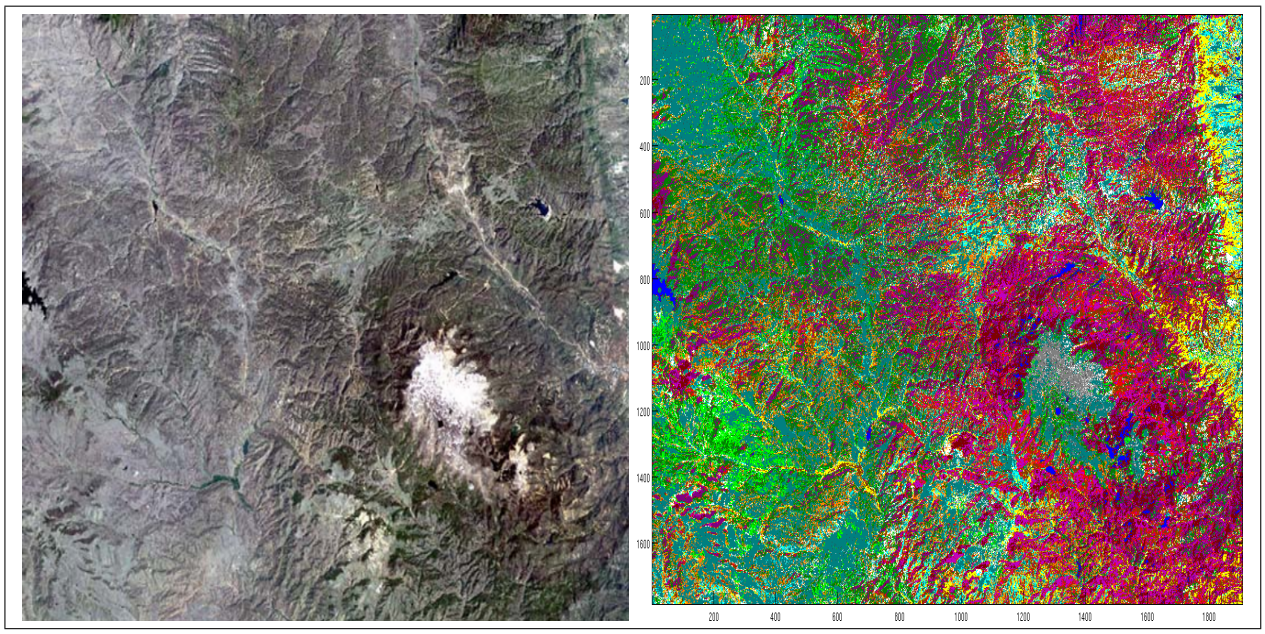

Fig. 7. (Left) a coloring in accordance to the RGB channels of the original data (data approx 1990) is shown, middle the obtained labeling by LKVQ and on the right the prototype positions as black dots. Only 2% of the data have been selected as prototypes.

Label	class	R	G	B	ground cover	#pixels
a	1	0	128	0	Scotch pine	581424
b	2	128	0	128	Douglas fir	355145
c	3	128	0	0	Pine / fir	181036
d	4	192	0	192	Mixed pines	272282
e	5	0	255	0	Mixed pines	144334
f	6	255	0	0	Aspen/Pines	208152
g	7	255	255	255	No veg.	170196
h	8	128	60	0	Aspen	277778
i	9	0	0	255	Water	16667
j	10	0	255	255	Moist meadow	97502
k	11	255	255	0	Bush land	127464
l	12	255	128	0	Pastureland	267495
m	13	0	128	128	Dry meadow	675048
n	14	128	128	128	Alpine veg.	27556
o	15	0	0	0	misclassif.	-

TABLE III

CLASSES OF THE SATELLITE IMAGE, USED SIMILARITY BASED COLORING (RGB) AND THE NUMBER OF PIXEL OF EACH CLASS.

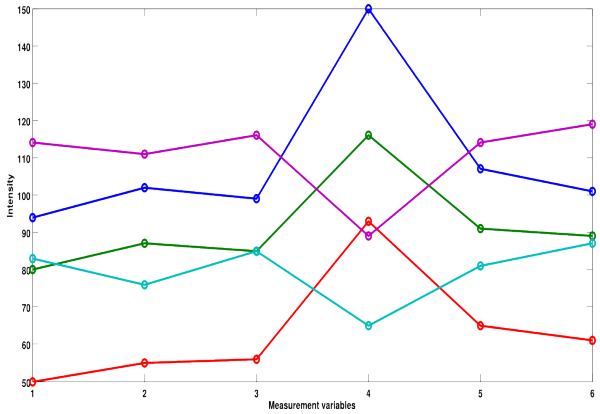

Fig. 8. Sample spectra from the remote satellite data

spectral bands. The spectral information, associated with each pixel of a LANDSAT scene is represented by a vector $\vec{v} \in \mathcal{V} \subseteq \mathbb{R}^{D_\mathcal{V}}$ with $D_\mathcal{V} = 6$. There are 14 labels describing different vegetation types and geological formations. The detailed labeling of the classes is given in Table III, here we also specify the used coloring for the subsequently generated images as obtained from the classification models[3]. The colors where chosen such that similar materials get similar colors in the RGB space.

[3]Colored versions of the image can be obtained from the corresponding author.

The size of the image is 1907×1784 pixels. For this dataset full labeling and measurement grid information is available. The labeling is used in the model application step to provide a coloring based on the obtained prototypes, only. The grid information is used to define f as well as to split the problem into bands. We choose $R = 60$ and $R' = 100$ taken from the data statistic as suggested before. This satellite image has been already used in [15] with $\approx 90\% - 96\%$ accuracy and $0.5\% - 10\%$ prototypes of the training data on a smaller subset of the image. Here we obtain a model using only 2% of the datapoints as prototypes and a labeling error of only 5%.

LKVQ and the experiments have been implemented in Matlab [10] on an Intel-dual core notebook computer with 2.8GHz and 2GB ram. LKVQ requires an efficient implementation of the linear programing problem also in case of larger problems therefore we used the *GNU Linear Program-*

ming Kit (GLPK) [8] providing an efficient implementation of different problem solvers in combination with a Matlab binding. To apply a LKVQ model to new data we have to identify the closest prototypes in accordance to the wta scheme mentioned before. This can get quite time consuming for larger codebooks. For Euclidean problems, we employ a kd-tree [2] implementation to store the prototypes providing log-linear search time in the wta scheme.

V. CONCLUSIONS

An improved version of the KVQ method has been proposed. It provides an inherent sparse solution rather a wrapper based sparsification. It has a guaranteed conversion due to the linear optimization model, but with additional costs due to the more complex optimization scheme. LKVQ automatically determines the number of prototypes but it is not necessarily the *minimal* number of prototypes which is a complex combinatorial problem but provides a good approximation by a *integer* linear programming approach.

LKVQ is capable to take local dependencies into account included in the cost function, to allow for optimization with dependencies. Additionally the cost function was modified such that the obtain solutions are more prototypical than for KVQ leading to improved reconstruction performance. LKVQ is now an effective clustering approach for medium-scale problems with local dependencies. In future work we will explore more data sets with dependencies and explore more advance optimization concepts also for very large problem settings.

Acknowledgment: This work was supported by the German Res. Fund. (DFG), HA2719/4-1 (Relevance Learning for Temporal Neural Maps) and by the Cluster of Excellence 277 Cognitive Interaction Technology funded in the framework of the German Excellence Initiative. The author would like to thank Vilen Jumutc for helpful discussions around local data dependencies. Thanks also to M. Augusteijn (University of Colorado) for providing this satellite remote imaging data and labeling.

REFERENCES

[1] Campbell, N.W., Thomas, B.T., Troscianko, T.: Neural networks for the segmentation of outdoor images. In: Solving Engineering Problems with Neural Networks. Proceedings of the International Conference on Engineering Applications of Neural Networks (EANN'96). Syst. Eng. Assoc, Turku, Finland. vol. 1, pp. 343–6 (1996)

[2] deBerg, M., vanKreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications. Springer (2000)

[3] Deininger, S.O., Gerhard, M., Schleif, F.M.: Statistical classification and visualization of maldi-imaging data. In: Proc. of CBMS 2007. pp. 403–405 (2007)

[4] Gersho, A., Gray, R.M.: Vector quantization and signal compression. Kluwer Academic Publishers, Norwell, MA, USA (1991)

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

[6] Jain, A.K.: Data clustering: 50 years beyond K-means. Pattern Recognition Letters 31, 651–666 (2010)

[7] Kohonen, T.: Self-Organizing Maps, Springer Series in Information Sciences, vol. 30. Springer, Berlin, Heidelberg (1995), (Second Extended Edition 1997)

[8] Makhorin, A.: Gnu linear programming kit (2010)

[9] Martinetz, T., Berkovich, S., Schulten, K.: "Neural-gas" Network for Vector Quantization and its Application to Time-Series Prediction. IEEE-Transactions on Neural Networks 4(4), 558–569 (1993)

[10] Mathworks Inc: Matlab 2010b. http://www.mathworks.com (20.01.2011) (2010)

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

[12] Pekalska, E., Duin, R.: The dissimilarity representation for pattern recognition. World Scientific (2005)

[13] Qin, A.K., Suganthan, P.N.: Kernel neural gas algorithms with application to cluster analysis. In: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4 - Volume 04. pp. 617–620. ICPR '04, IEEE Computer Society, Washington, DC, USA (2004), http://dx.doi.org/10.1109/ICPR.2004.520

[14] Richards, J.A., Jia, X.: Remote Sensing Digital Image Analysis. Springer, New York (1999), 3rd Ed.

[15] Schleif, F.M., Ongyerth, F.M., Villmann, T.: Supervised data analysis and reliability estimation for spectral data. NeuroComputing 72(16-18), 3590–3601 (2009)

[16] Simmuteit, S., Schleif, F.M., Villmann, T.: Hierarchical evolving trees together with global and local learning for large data sets in maldi imaging. In: Proceedings of WCSB 2010. pp. 103–106 (2010)

[17] Tipping, M.E., Schoelkopf, B.: A Kernel Approach for Vector Quantization with Guaranteed Distortion Bounds. In: Proc. of AISTAT'01 (2001)

[18] Vural, V., Fung, G., Krishnapuram, B., Dy, J.G., Rao, R.B.: Using local dependencies within batches to improve large margin classifiers. Journal of Machine Learning Research 10, 183–206 (2009)

[19] Weston, J., Elisseeff, A., BakIr, G., Sinz, F.: Spider toolbox (2011), http://www.kyb.tuebingen.mpg.de/bs/people/spider/ (last visit 28012011)