Soft Competitive Learning for large data sets

Frank-Michael Schleif, Xibin Zhu, and Barbara Hammer

Abstract Soft competitive learning is an advanced k-means like clustering approach overcoming some severe drawbacks of k-means, like initialization dependence and sticking to local minima. It achieves lower distortion error than k-means and has shown very good performance in the clustering of complex data sets, using various metrics or kernels. While very effective, it does not scale for large data sets which is even more severe in case of kernels, due to a dense prototype model. In this paper, we propose a novel soft-competitive learning algorithm using core-sets, significantly accelerating the original method in practice with natural sparsity. It effectively deals with very large data sets up to multiple million points. Our method provides also an *alternative fast kernelization* of soft-competitive learning. In contrast to many other clustering methods the obtained model is based on only few prototypes and shows natural sparsity. It is the first natural sparse kernelized soft competitive learning approach. Numerical experiments on synthetical and benchmark data sets show the efficiency of the proposed method.

1 Introduction

Clustering algorithms are successful unsupervised machine learning algorithms partitioning a data set into groups. They are able to deal with complex data sets by employing appropriate distance measures. Soft-Competitive-Learning (SCL), an extension of k-means, is a very effective clustering algorithm [10]. It has been successfully applied in different domains like time series prediction [10], image and signal processing [8], bioinformatics and others. In [11, 12] different kernelizations and improvements thereof were proposed, leading to models, competitive to e.g. kernel k-means [15].

CITEC centre of excellence, Bielefeld University, 33615 Bielefeld, Germany,
{fschleif|xzhu|bhammer}@techfak.uni-bielefeld.de

Kernel clustering methods got much attention in the last years [3, 4]. The basic idea of kernel methods is to map the low-dimensional data into a high-dimensional feature space, induced by a kernel function, to obtain linear separability. For SCL this is either done by directly kernelizing the original method [11], its batch variant [7], or by using specific differentiable kernel-functions [13].

For other methods, e.g. kernel k-means, also the support vector description was used [3], referred to as kernel-grower. Kernel grower is based on the classical k-means algorithm and the one-class SVM concept. It maps the data to the kernel-induced feature space and processes the data in an iterative strategy like k-means until a stopping criterion is met. However, instead of computing the centers directly based on the data it computes the smallest sphere enclosing the data by means of the support vector data description (SVDD) [17]. While very effective and flexible from a theoretical point of view it is inefficient for large data sets. Liang et al. [9] proposed to use the core-set concept instead of SVDD in kernel-grower leading to an efficient kernelized core-set clustering with linear complexity in the number of points.

Kernel k-means and its extensions suffer from initialization dependence, which is still subject of research [19]. These issues are fundamental in the k-means algorithm and also kernel-grower and scaled kernel-grower are affected by this.

SCL provides an interesting alternative to achieve less initialization dependent clustering solutions in a k-means like manner [10] but the kernelized approaches of SCL like [11, 13] suffer from its high complexity. Kernel-SCL (KSCL) [11] represents the prototypes or cluster centers by a linear combination of the data points, using a learned coefficient matrix. This coefficient matrix is typically dense such that a lot of data points contribute to the representation of a prototype. Also accelerations of kernel SCL and the introduction of additional sparsity costs in the kernel SCL cost function (see [12]) still do not scale for (very) large data.

In this paper we propose a new algorithm for the kernelization of SCL by means of the core-set technique, which we call core-SCL. The main innovation is to calculate the center, or prototypes of a cluster by means of a small core-set of points which can be efficiently identified by the core-set algorithm. However, the final centers are not directly obtained from the core-set solution like in [9] but by calculating a soft competitive learning solution as detailed in the following. In our algorithm the competitive learning update has much lower complexity, because it is based only on a typically very small set of points defined by the core-set solutions. The overall complexity of our algorithm is still $C \times O(N)$ for a full training procedure, which is the same as for classical SCL but the constant costs C are substantially lower, leading to feasible runtime under practical conditions in contrast to the classical SCL. Core-SCL permits the clustering of up to multiple million of points with linear complexity and is less sensitive to initialization or local minima.

The obtained clustering model is finally based on the adapted core-set information of each cluster. The number of core-set points is typically very small compared to the original cluster size. The underlying coefficient matrix is more sparse compared to KSCL. The cluster centers are determined using these few coefficients.

We evaluate our new approach on synthetic and real life data and compare with different alternative clustering techniques. The paper is organized as follows. In section 2 and 3 we give some preliminary information, followed by a review of kernel soft-competitive learning (kernel-SCL). Section 4 presents the core-SCL algorithm. In section 5, an empirical evaluation on different benchmarking data are shown. We conclude with an outlook and open problems in section 6.

2 Soft Competitive Learning

Let $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ be a data set with vectors $\mathbf{v}_j \in \mathbb{R}^d$, d denoting the dimensionality, N the number of samples. We call *codebook* the set $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ with each elements $\mathbf{w}_i \in \mathbb{R}^d$ and n as the number of codebook vectors or prototypes. Typically $n \ll N$ and scales in the expected number of clusters. $\mathbf{v}_i$ are represented by one prototype $\mathbf{w}_j$ as explained in more detail subsequently. The prototypes induce a clustering by means of their receptive fields which consist of the points $\mathbf{v}$ for which $d(\mathbf{v}, \mathbf{w_i}) \leq d(\mathbf{v}, \mathbf{w_j})$ holds for all $j \neq i$, $d(\cdot, \cdot)$ denoting a distance measure, typically the Euclidean distance.

Kernel methods are attractive approaches to analyze complex data which are not linear separable e.g. in the Euclidean domain. The data are mapped using a non-linear transformation function $\Phi : V \to \mathscr{F}$ into a potentially high-dimensional feature space where a linear separation of the data can be achieved. Typically this is done implicitly using the so called kernel trick [14] and a kernel function.

A kernel function $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is implicitly induced by the feature mapping Φ into some possibly high dimensional feature space $\mathscr{F}$ such that

$$\kappa(\mathbf{v}_1, \mathbf{v}_2) = \langle \Phi(\mathbf{v}_1), \Phi(\mathbf{v}_2) \rangle_{\mathscr{F}} \tag{1}$$

holds for all vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, where the inner product in the feature space is considered. Hence κ is positive semi-definite. Using the linearity in the Hilbert-space, we can express dot products of elements of the linear span of Φ in the form $\sum_i \alpha_i \Phi(\mathbf{v}_i)$ and images $\Phi(\mathbf{v})$ via the form $\sum_i \alpha_i \kappa(\mathbf{v}_i, \mathbf{v})$. This property is used in [11], to derive a kernelization of soft competitive learning but also many other prototype based approaches have been extended by kernel concepts [4].

3 The Soft Competitive Learning Network

The SCL algorithm is a type of vector quantizer providing a compact representation of the underlying data distributions [10]. Its goal is to find prototype locations $\mathbf{w}_i$ such that these prototypes represent the data V, distributed according to $\mathbb{P}$, as accurately as possible, minimizing the energy function:

$$E_{SCL}(\gamma) = \frac{1}{C(\gamma,n)} \sum_{i=1}^{N} \int \mathbb{P}(\mathbf{v}) \cdot h_\gamma(\mathbf{v}_i, \mathbf{W}) \cdot (\mathbf{v} - \mathbf{w}_i)^2 \, d\mathbf{v} \tag{2}$$

with neighborhood function of Gaussian shape:

$$h_\gamma(\mathbf{v}_i, \mathbf{W}) = \exp(-r_i(\mathbf{v}, \mathbf{W})/\gamma) \tag{3}$$

$r_i(\mathbf{v}, \mathbf{W})$ yields the number of prototypes $\mathbf{w}_j$ for which the relation $d(\mathbf{v}, \mathbf{w}_j) \leq d(\mathbf{v}, \mathbf{w}_i)$ is valid, i.e. the winner rank. $C(\gamma, n)$ is a normalization constant depending on the neighborhood range γ. The SCL learning rule is derived by stochastic gradient descent:

$$\triangle \mathbf{w}_i = \varepsilon \cdot h_\gamma(\mathbf{v}_i, \mathbf{W}) \cdot (\mathbf{v} - \mathbf{w}_i) \tag{4}$$

with learning rate ε. Typically, the neighborhood range γ is decreased during training.

We now briefly review the main concepts used in Kernelized Soft Competitive Learning (kernel-SCL) proposed in [11]. Kernel-SCL optimizes the same cost function as SCL but with the Euclidean distance substituted by a distance induced by a kernel. Since the feature space is unknown, prototypes are expressed implicitly as linear combination of feature vectors

$$\mathbf{w}_i = \sum_{l=1}^{N} \alpha_{i,l} \Phi(\mathbf{v}_l) \tag{5}$$

$\alpha_{i,\cdot} \in \mathbb{R}^N$ is the corresponding coefficient vector. The coefficient vector are stored in a matrix $\Gamma \in \mathbb{R}^{n \times N}$. Distance in feature space for $\Phi(\mathbf{v}_j)$ and $\mathbf{w}_i$ is computed as:

$$d_{i,j}^2 = \|\Phi(\mathbf{v}_j) - \mathbf{w}_i\|^2 = \|\Phi(\mathbf{v}_j) - \sum_{l=1}^{N} \alpha_{i,l} \Phi(\mathbf{v}_l)\|^2 \tag{6}$$

$$= k(\mathbf{v}_j, \mathbf{v}_j) - 2 \sum_{l=1}^{N} k(\mathbf{v}_j, \mathbf{v}_l) \cdot \alpha_{i,l} + \sum_{s,t=1}^{N} k(\mathbf{v}_s, \mathbf{v}_t) \cdot \alpha_{i,s} \alpha_{i,t} \tag{7}$$

The update rules of SCL can be modified by substituting the Euclidean distance by the formula (6) and taking derivatives with respect to the coefficients $\alpha_{i,l}$. Further, substituting the prototypes by linear combinations, the adaptation rule of the Γ matrix becomes

$$\alpha_{jl} := \alpha_{jl}(1 - \eta h_\sigma(r_{ij})) \text{ if } l \neq i \tag{8}$$

$$\alpha_{ji} := \alpha_{ji}(1 - \eta h_\sigma(r_{ij})) + \eta h_\sigma(r_{ij}) \tag{9}$$

For a Gram matrix, we can w.l.o.g. restrict the coefficients such that prototypes are contained in the convex hull of the data points in the feature space, i.e. the coefficients are non-negative and sum up to one. It should be noted that the kernel-SCL and its batch variant [7] have a complexity of $O(N^2)$, each. Both methods represent the prototypes by linear combinations of the data points. The coefficients α_{ij}

are stored in the matrix Γ which is typically dense. Therefore the distance calculations between a prototype and a datapoint requires almost all points due to the reconstruction of the prototype via the linear combination.

4 Core Soft Competitive Learning

The notion of core-sets appears in solving the approximate minimum enclosing ball (MEB) problem in computational geometry [1].

Definition 1 (Minimum enclosing ball problem). Let $\{\mathbf{v}_1,\ldots,\mathbf{v}_m\} \in \mathbb{R}^d$ a set of points. The objective is to find a minimum enclosing ball with radius R and center point $\mathbf{c}$ such that $\|\mathbf{c}-\mathbf{v}_i\|^2 \leq R^2\ \forall i$. Hence $B(c,R) = \{\mathbf{v}|R \geq \|c-\mathbf{v}\|, \mathbf{v} \in V\}$.

As shown in [18], the MEB problem can be equivalently express as a quadratic dual optimization problem:

$$MEB = \min_{\alpha_i \geq 0, \sum \alpha_i = 0} \alpha K \alpha^\top - \sum_i \alpha_i K(i,i)$$
$$\text{with } K \text{ the kernel matrix defined on } V$$

and $\mathbf{c}$ and $\mathbf{v}$ represented in a kernel space as shown before. The radius R is obtained by solving $R = \sum_i \alpha_i K(i,i) - \alpha K \alpha^\top$. The center is used only in the distance calculation which can be expressed using the kernel trick and the linear combination of $\mathbf{c}$ based on the obtained α-vector, $\mathbf{c} = \sum_{i=1} \alpha_i \phi(\mathbf{v}_i)$ as shown for kernel-SCL.

Definition 2 (Core set). Let $S = \{\mathbf{v}_1,\ldots,\mathbf{v}_k\} \subseteq V$, then $Q \subset S$ is a core set, if $S \subset B(c,(1+\varepsilon)R)$ and $B(c,R) = MEB(Q)$.

An encouraging property of core-sets is that the number of elements in it is independent of the data dimensionality and size [1]. The main concepts involved in core-sets are illustrated in Figure 1 (left).

The core-SCL algorithm can be calculated in the Euclidean space or using a kernel-induced feature space. In the following we will present core-SCL for the more generic case of arbitrary kernels. The core-SCL clustering is initialized by specifying the number of prototypes n and an initial coefficient matrix $\Gamma = n \times N$. We initialize the matrix Γ randomly such that for each row (α_i) two entries have a value of 0.5. Further a learning rate and neighborhood cooperation is initialized equivalently to the standard SCL and a copy of the Γ-matrix, $\hat{\Gamma}$ is stored for later use in the update. The α_i encode the prototype positions using Equation (5). In the learning phase the Γ-matrix is adapted such that the prototype positions are optimized following the optimization scheme of SCL. The algorithm iterates the following steps:

1. calculate the receptive fields using Eq. (6) with $\hat{\Gamma}$
2. if the receptive field has not changed, continue with step 4
3. calculate the core-set for each receptive field using the algorithm of [1]

4. calculate distances between **W** and all *core-set points* using Eq. (6)
5. calculate the corresponding neighborhood function using Eq. (3)
6. apply Eq. (8) to Γ using $\hat{\Gamma}$ with respect to the core-set points
7. store Γ as $\hat{\Gamma}$, continue with step 3

Like in [9] we limit the influence of potential outliers in the current approximation of the receptive field, by modifying the step 3 in the prior algorithm such that a fraction τ of the points can be left out during the core-set approximation. Further we use probabilistic speedup [18]. The overall algorithm stops if the receptive fields did not change for a number of iterations or an upper number of cycles is reached. In each cycle the learning rate and neighborhood range is adapted in the same manner as for kernel-SCL.

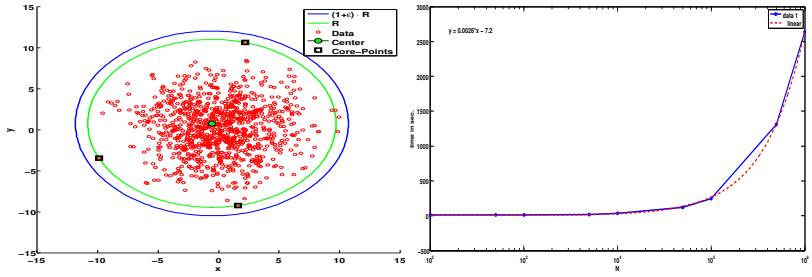

Fig. 1 (Left) Core-Set schema: Minimum enclosing ball solution of a set of data points. The core-set consists of the points indicated by squares, implying a minimum enclosing ball of radius R and its approximation with a radius $(1+\varepsilon)\cdot R$ (outer circle). The outer circle includes all points of the data within a tolerance of $(1+\varepsilon)\cdot R$. In this example 1000 points are described by a circle defined upon 3 core-set points. (Right) Runtime analysis of core-SCL varying N on a checkerboard data set. Note that the x-axis uses log-scale. A linear fit (red) agrees well with the expected complexity of $O(N)$.

4.1 Runtime analysis

The core-SCL calculates prototype positions for a constant number of cycles. In each cycle the receptive fields have to be calculated, with $n \times N$ distance calculations and n sorting operations to determine the winner. Hence step 3 has a complexity of $O(N)$. This is a substantial difference to kernel-SCL or batch kernel-SCL where these calculations have $O(N^2)$ complexity. The quadratic complexity of these algorithms also remains if a k-approximation of the Γ matrix is done (e.g. only the k largest alpha values are kept), because this can only be applied after an initial number of steps, typically 10 cycles or in the final model to ease later interpretation.

The core set calculation (step 3) of each receptive field has worst case runtime complexity of $O(N)$. This is because in each core-set iteration a new core-set vector is added until the MEB is found, which requires the test $R \geq \|c-\mathbf{v}\|, \mathbf{v} \in V$ for each

point in the receptive field which is not yet within the MEB. Employing probabilistic speedup [18], this complexity can be substantially reduced in practice, checking only a random subset of the points. As discussed in [18], it is sufficient to select one (or multiple) random subsets of 59 points $\mathbf{v} \in S$, with some moderate assumptions on the distance distribution. As shown in [16] by using a small random sample S' from S the closest point obtained from S' is with probability 95% among the closest 5% of points from the whole S.

The distance and rank calculations between the prototypes and the core-set points is done in $O(n \times Z)$ including sorting in $O(\log n)$, where Z is the size of all points which are core-set points. The obtained ranks are used to update Γ using $\hat{\Gamma}$.

Note that the number of core-set points per MEB calculation is bounded by $O(\frac{1}{\varepsilon^2})$ [1] and hence remains small with respect to each cluster. Due to the small number of core-set points per cluster we have $Z \ll N$. Hence we have a complexity of $O(N)$ for the steps 4 to 6.

Summarized, the overall complexity is linear $O(N)$ in N, since $n \ll N$. This is also reflected by the runtime analysis shown in Figure 1 (right).

4.2 Memory complexity analysis

The consumed memory is mainly determined by the size of the kernel-matrix $K = N \times N$ and the coefficient-matrix Γ which is $n \times N$. The full kernel matrix does not need to be calculated but only a sub-part is necessary in each iteration. In core-SCL the distance between the n prototypes and all N datapoints needs to be calculated. At this step the corresponding kernel sub-matrix has to be calculated. It consist of the rows of the non-vanishing α_{ij} for a row i. And relates directly to the identified core-sets indices causing the non-vanishing alphas. If we define $S_{all} = \{S_1, \ldots, S_n\}$ as the unique list of core-set index sets for each cluster, the corresponding sub-matrix in the calculation of the receptive field i is roughly $|S_i| \times N$, which is typically much smaller than N^2 and hence remains linear. The number of core-set points in an MEB calculation is bounded by $O(\frac{1}{\varepsilon^2})$ [1]. This calculation can potentially be further simplified by using the Nyström approximation as shown e.g. in [12]. The Γ matrix has a complexity of $n \times N$, which can probably be reduced further using appropriate storage-classes due to the sparsity of Γ. Accordingly, the memory complexity is roughly linear in $O(N)$.

5 Experiments

In the following we show the efficiency of core-SCL on some artificial and real life data sets and compare with results reported in [9]. First we start with a checker-board simulation of 3×3 fields. Each cluster is Gaussian with clear separation between the means and a small variance, causing small overlap. The 9 prototypes are initialized

randomly in the data and the objective is to position each prototype in a cluster. In Figure 2 we show a run of standard kernel k-means and core-SCL on the checkerboard data using a linear kernel. One observes a classical problem of kmeans which is efficiently overcome with the use of soft-competitive learning. K-means sticks in local minima and is unable to obtain a reasonable solution. As a further example

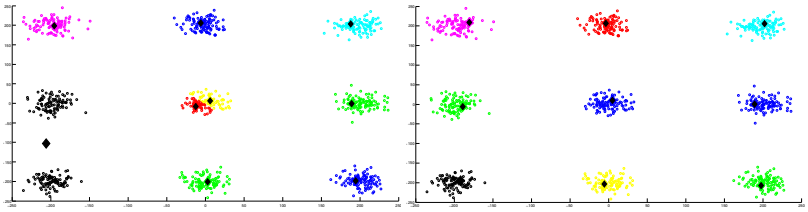

Fig. 2 Results for the checkerboard data with 100 points per cloud, black markers indicated the prototypes. Left: kmeans - obviously fails to point one prototype per cluster due to initialization problems and dead-lock situations. Right:core-scl, each cluster is represented by a prototype.

we use the classical ring data set which consist of two rings which are not linearly separable. Each ring has 4000 points with added Gaussian noise in $N(0, 0.25)$. The core-SCL needs 10 steps and an RBF kernel with $\sigma = 0.5$ to separate the two rings as shown in Figure 3. In a 10-fold crossvalidation core-SCL was always able to obtain a perfect clustering, 0%-error and the mean sparsity of the model was around 92%, hence ≈ 300 points have been used to describe the prototypes in each model. Also scaled kernel-grower achieved a perfect clustering for these data and the sparsity was even better using only ≈ 11 points per model to represent the prototypes.

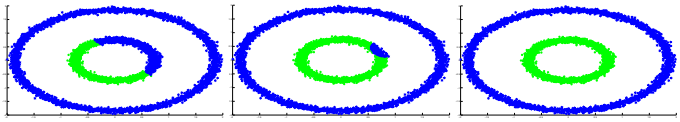

Fig. 3 Core-SCL with 2 prototypes applied to ring dta (inner - circles, outer - diamond). Perfect post-labeling at iteration 10 with the inner ring, separated from the outer ring.

In the following we show the effectiveness of core-SCL for two real life, standard data sets taken from the UCI machine learning repository [2] and compare to different clustering approaches. For comparison we consider, affinity propagation, a generic clustering approach based on a factor-graph model [6]. It provides natural sparsity, since the model consists of exemplars in the original data space, equivalent to the number of requested clusters. However AP does not scale for larger data sets and even in the range of about a thousand data points the algorithm is already slow. Additionally we consider a batch variant of K-SCL [7]. In the batch version all data points are processed in one step. This is very efficient, but can not be done for extremely large data sets ($N \gg 10000$) without approximation or sub-sampling.

We consider the Landsat satellite data with 36 dimensions, 6 classes, and 6435 samples, the Phoneme data with 20 dimensions, 13 classes, and 3656 samples and the spam dataset with two classes, 57 dimensions and 4601 samples. The results are given in Table 1[1].

To show the effectiveness of our approach also for large data sets we consider a large version of the checkerboard data, as described above, but with ≈ 1 million points and the KDD-intrusion detection taken from [18] with training data of $\approx$ 5 million points. It contains connection records of network traffic. The task is to separate normal connections from attacks on the preprocessed raw data as detailed in [18]. The results for these large scale experiments are shown in Table 2 for core-SCL only, since the other methods took too long for the complete experiments.

In all experiments K-SCL and core-SCL are initialized randomly. For AP we used the standard settings described in [6]. Experiments run until convergence, with an upper limit of 100 cycles. We use the standard linear kernel and the ELM kernel, a defacto parameter-free, optimal RBF kernel as discussed in [5]. The number of clusters and used kernel are: LANDSAT (100,linear), PHONEME (100,linear), SPAM (2,ELM), CHECKER (9,linear), INTRUSION (2,linear).

	core-SCL	AP	K-SCL (batch)
Landsat			
accuracy	$86.00\% \pm 1.3$	$\mathbf{86.3}\% \pm 0.2$	$86.1\% \pm 0.2$
runtime	**126.43**	1478	329
q-error	1704	1568	1545
sparsity	**0.11**%	–	3.6%
Phoneme			
accuracy	$84.05\% \pm 1.8$	$87.1\% \pm 0.2$	$\mathbf{87.9}\% \pm \mathbf{0.3}$
runtime	**92.41**	1237	104
q-error	2656	2578	2487
sparsity	**0.23**%	–	3.6%
Spam			
accuracy	$82.29\% \pm 4.4$	$76.00\% \pm 4.0$	$\mathbf{84.35}\% \pm \mathbf{1.7}$
runtime	**13.92**	304	77
q-error	420.26	606	588.25
sparsity	**0.4**%	–	100%

Table 1 Mean classification accuracy ($\pm$ standard dev.) of core-SCL to other methods within a 5-fold cross-validation. The runtime is given in seconds. The dual costs are given for the test data averaged over all runs. The sparsity is the percentage of points used to reconstruct all prototypes.

6 Conclusions

We proposed a novel kernelization of Soft Competitive Learning employing core-sets. The approach automatically leads to sparse models with low memory consumption compared to the traditional kernel-SCL. The results show that the runtime

[1] The number of examples in AP is the same as the number of cluster, hence the AP model is always the most sparse model possible.

	Checker	Intrusion
accuracy	$100 \pm \%0$	$93.84\% \pm 0.21$
runtime	107.63	434.50
q-error	1334	15267
sparsity	0.03%	$1e^{-7}\%$

Table 2 Mean classification accuracy ($\pm$ standard dev.) of core-SCL for the large data within a 5-fold cross-validation. The runtime is given in seconds. The dual costs are given for the test data averaged over all runs. The sparsity is the percentage of points used to reconstruct all prototypes. AP and K-SCL did not get feasible runtimes.

could be substantially improved using core-sets, avoiding the consecutive updates for all data points per iteration as in the traditional online kernel-SCL, but also compared to batch kernel-SCL, for larger data sets. The algorithm is similarly efficient with respect to the dual-quantization error and post-labeling accuracy compared to the other methods. While AP and K-SCL cannot be used to analyze very large data sets, core-SCL can be easily applied for such data.

Acknowledgment

This work has been supported by the German Res. Found. (DFG), HA2719/4-1 (Relevance Learning for Temporal Neural Maps) and in the frame of the centre of excellence 'Cognitive Interaction Technologies'.

References

1. Badoiu, M., Har-Peled, S., Indyk, P.: Approximate clustering via core-sets. In: STOC. pp. 250–257 (2002)
2. Blake, C., Merz, C.: UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science, available at: http://www.ics.uci.edu/ mlearn/MLRepository.html (1998)
3. Camastra, F., Verri, A.: A Novel Kernel Method for Clustering. IEEE TPAMI 27(5), 801–805 (2005)
4. Filippone, M., Camastra, F., Massulli, F., Rovetta, S.: A survey of kernel and spectral methods for clustering. Pattern Recognition 41, 176–190 (2008)
5. Frénay, B., Verleysen, M.: Parameter-insensitive kernel in extreme learning for non-linear support vector regression. Neurocomputing 74(16), 2526–2531 (2011)
6. Frey, B., Dueck, D.: Clustering by message passing between data points. Science 315, 972–976 (2007)
7. Hammer, B., Hasenfuss, A.: Topographic mapping of large dissimilarity data sets. Neural Computation 22(9), 2229–2284 (2010)
8. Labusch, K., Barth, E., Martinetz, T.: Soft-competitive learning of sparse codes and its application to image reconstruction. Neurocomputing 74(9), 1418–1428 (2011)
9. Liang, C., Xiao-Ming, D., Sui-Wu, Z., Yong-Qing, W.: Scaling up kernel grower clustering method for large data sets via core-sets. Acta Automatica Sinica 34(3), 376–382 (2008)
10. 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)

11. Qin, A.K., Suganthan, P.N.: A novel kernel prototype-based learning algorithm. In: Proc. of ICPR'04. pp. 2621–624 (2004)
12. Schleif, F.M., Villmann, T., Hammer, B., Schneider, P.: Effcient kernelized prototype-based classification. Journal of Neural Systems 21(6), 443–457 (2011)
13. Schleif, F.M., Villmann, T., Hammer, B., Schneider, P., Biehl, M.: Generalized derivative based kernelized learning vector quantization. In: Proceedings of IDEAL 2010. pp. 21–28 (2010)
14. Schoelkopf, B., Smola, A.: Learning with Kernels. MIT Press (2002)
15. Schölkopf, B., Smola, A.J., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10(5), 1299–1319 (1998)
16. Smola, A.J., Schölkopf, B.: Sparse greedy matrix approximation for machine learning. In: Langley, P. (ed.) ICML. pp. 911–918. Morgan Kaufmann (2000)
17. Tax, D.M.J., Duin, R.P.W.: Support vector domain description. Pattern Recognition Letters 20(11-13), 1191–1199 (1999)
18. Tsang, I.W., Kwok, J.T., Cheung, P.M.: Core vector machines: Fast svm training on very large data sets. Journal of Machine Learning Research 6, 363–392 (2005)
19. Tzortzis, G., Likas, A.: The global kernel k-means clustering algorithm. In: IJCNN. pp. 1977–1984. IEEE (2008)