A Constraint Acquisition Method for Data Clustering

João M.M. Duarte[1,2], Ana L.N. Fred[1], and Fernando Jorge F. Duarte[2]

[1] Instituto de Telecomunicações, Instituto Superior Técnico, Lisboa, Portugal
{jduarte,afred}@lx.it.pt
[2] GECAD - Knowledge Engineering and Decision-Support Research Center, Institute of Engineering Polytechnic of Porto, Portugal
{jmmd,fjd}@isep.ipp.pt

Abstract. A new constraint acquisition method for parwise-constrained data clustering based on user-feedback is proposed. The method searches for non-redundant intra-cluster and inter-cluster query-candidates, ranks the candidates by decreasing order of interest and, finally, prompts the user the most relevant query-candidates. A comparison between using the original data representation and using a learned representation (obtained from the combination of the pairwise constraints and the original data representation) is also performed. Experimental results shown that the proposed constraint acquisition method and the data representation learning methodology lead to clustering performance improvements.

Keywords: Constraint Acquisition, Constrained Data Clustering.

1 Introduction

Data clustering is an unsupervised learning technique which aims to find structure in data. Domain objects are grouped into clusters such that objects that are alike are placed in the same cluster while dissimilar objects are assigned to different clusters [1]. Due to its unsupervised nature, a data clustering algorithm only has access to features that describe the objects or to (dis)similarities between pairs of objects, and the clustering solution is obtained by optimizing the same objective-function, irrespectively the application.

In many situations, the data analyst may have extra information for a particular application, or may want to express his preferences or conditions to guide data clustering. To accomplish it, the data representation can be manipulated (e.g.: by adding, removing or modifying data features) although it can be very difficult or impractical. A simpler and more intuitive way of doing it consists of using constraints in data clustering. Constrained data clustering algorithms [2–4] use *a priori* knowledge about the data, mapped in form of constraints, to produce more useful clustering solutions. The constraints can be set at a general level by defining rules which are applied to the entire data set, such as data clustering with obstacles [5]; at an intermediate level, where clustering is guided by rules involving the data features [6] or the groups' characteristics, such as, the minimum and maximum capacity [7]; or at a more particular level, where

J. Ruiz-Shulcloper and G. Sanniti di Baja (Eds.): CIARP 2013, Part I, LNCS 8258, pp. 108–116, 2013.

the constraints are applied to the domain objects, by using some labeled data [3] or defining relations between pairs of objects [2]. Relations between pairs of objects, usually represented as must-link and cannot-link constraints, have been the most studied due to their versatility. Many constraints on more general levels can be transformed into must-link and cannot-link constraints.

It would be expected that the use of constraints should always improve, or at least not to worsen, the quality of data clustering. However, it was demonstrated that the use of constraints may in fact harm clustering performance, even when the set of constraints is not noisy [8]. Therefore, the acquisition of constraint sets that effectively improve clustering performance is a very important topic in constrained clustering. Some active learning algorithms for constraint acquisition have already been proposed regarding the search of labels for some data [9, 10] and the identification of relations between pairs of objects [2, 11].

It is known that learning distance metrics can improve the performance of classification and clustering. Typically, distance learning algorithms can be categorized into the supervised and unsupervised categories, depending on the existence of class labels for the objects. Nonetheless, some methods [12–14] can use the pairwise constraints to learn a new distance function or data representation.

In this work, we propose a new method for acquiring useful pairwise constraints. Our method tries to identify relevant query candidates for a given clustering algorithm, ranks the candidates according to their importance, and then selects the top candidates to query the user. We compare the proposed method with the random acquisition of constraints and the Explore-Consolidate [11] approach. We also evaluate the effectiveness of using distance metric learning in constrained clustering, the effect of the constraint acquisition methods in the distance learning, and the corresponding impact in the quality of data clustering.

The rest of the paper is organized as follows. In section 2 we briefly present some related work on constrained clustering and distance metric learning with constraints. A new approach for selecting pairwise constraints is presented in section 3. The performance of the proposed method is evaluated in section 4. The conclusions and future work is presented in section 5.

2 Related Work

Let $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ be a data set composed of n domain objects $\mathbf{x}_i$, $\mathcal{R}_=$ the set of must-link constraints which contains pairs of objects $(\mathbf{x}_i, \mathbf{x}_j)$ that should belong to the same cluster, and $\mathcal{R}_\neq$ the set of cannot-link constraints containing pairs of objects that should belong to different clusters. The goal of a constrained clustering algorithm consists of dividing $\mathcal{X}$ into K clusters regarding both the data representation (e.g. vectorial and (dis)similarity representations) and the constraints expressed in $\mathcal{R}_=$ and $\mathcal{R}_\neq$, resulting in a data partition $P = \{C_1, \cdots, C_K\}$ where C_k represents an individual cluster.

The Constrained Average-Link (CAL) [15] is based on the agglomerative hierarchical clustering algorithm Average-Link [16]. The algorithm works as follows. It starts with n clusters, one for each domain object $\mathbf{x}_i$. Then, at

each step, the two closest clusters, according to a distance measure between clusters, are merged. The process iterates until some stopping criteria is met (e.g. a predefined number of clusters K is reached) or all objects belong to same cluster. The distance between clusters measures the average distance between all pairs of objects belonging to different clusters plus a penalization for each constraint that is not satisfied. This distance is defined as $d(C_k, C_l) = \frac{1}{|C_k||C_l|}(\sum_{i=1}^{|C_k|}\sum_{j=1}^{|C_l|} \text{dist}(\mathbf{x}_i, \mathbf{x}_j) - \text{I}_{=}(\mathbf{x}_i, \mathbf{x}_j) + \text{I}_{\neq}(\mathbf{x}_i, \mathbf{x}_j))$, where $\text{I}_a(\mathbf{x}_i, \mathbf{x}_j) = \text{p}$ if $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{R}_a$ and 0 otherwise. $\text{p} \geq 0$ is a user parameter that influences the "softness" of the constraints. In our experiments we defined p as the maximum distance between objects in a data set.

An easy but naive way to generate pairwise constraints is the Random Acquisition of Constraints (RAC) and consists of randomly selecting, iteratively, two objects $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{X}$ that were not previously tested and ask the user (or some oracle) if both objects should be assigned to the same group. If the answer is "Yes", a must-link constraint is added to the set of must-link constraints, $\mathcal{R}_{=} = \mathcal{R}_{=} \cup \{(\mathbf{x}_i, \mathbf{x}_j)\}$. If the answer is "No" a cannot-link constraint is added to the set of cannot-link constraints $\mathcal{R}_{\neq} = \mathcal{R}_{\neq} \cup \{(\mathbf{x}_i, \mathbf{x}_j)\}$. If the user cannot decide, simply skip to the next iteration. The process repeats until a predefined number of constraints is achieved.

The Explore-Consolidate [11] is another method for constraint acquisition and consists of two phases: the Explore phase, where the algorithm identifies a neighborhood $\mathcal{N}_k$ for each cluster in the data set which defines a skeleton of the clusters' structure; and the Consolidate phase, where objects not attributed to any neighborhood are assigned to one of them. The Explore algorithm starts by selecting a random object which forms the first neighborhood. Then, while the maximum number of queries is not reached and until K disjoint neighborhoods are not found, the farthest object $\mathbf{x}$ from all the existing neighborhoods is selected. Queries between $\mathbf{x}$ and a random object belonging to each neighborhood are posed. If $\mathbf{x}$ does not belong to any neighborhood, a new one is formed with $\mathbf{x}$. The Consolidate algorithm first computes the centroids $\overline{\mathbf{x}}_k$ of each neighborhood $\mathcal{N}_k$. Then, while the maximum number of queries is not reached, an object $\mathbf{x}$ that does not belong to any neighborhood is randomly selected. Queries are posed between $\mathbf{x}$ and each neighborhood by increasing order of its distance to the centroids $\overline{\mathbf{x}}_k$ until a must-link is obtained. After the Explore and Consolidate phases, the pairwise constraint sets are formed by adding a must-link constraint for each pair of objects that belong to the same neighborhood, and a cannot link constraint for each pair of objects belonging to different neighborhoods.

There may be contradictions between the relations of objects in the original representation of the data and sets of constraints. We are interested to find out how learning a new data space representation, which simultaneously represents both the original data and the clustering preferences, influences the performance of data clustering. The Discriminant Component Analysis (DCA) [14] is a distance metric learning algorithm capable of learning a new data representation from the original data and a set of constraints. The DCA builds a set of chunklets $\mathcal{Q} = \{Q_1, \cdots, Q_q\}$, i.e. groups of domain objects connected by

must-link constraints, and a set of discriminative chunklets $\mathcal{S} = \{S_1, \cdots, S_q\}$, one for each chunklet Q_i. Each element of the discriminative chunklet S_i indicates the chunklets that have at least one cannot-link constraint connecting a object in Q_i. Then DCA learns a data transformation which minimizes the variance between domain objects in the same chunklet Q_i and maximizes the variance between discriminative data chunklets S_i. The covariance matrices, $\mathbf{C}_b$ and $\mathbf{C}_w$, store the total variance between domain objects in each $S_i \in \mathcal{S}$ and the total variance within domain objects in the same chunklets $\forall Q_i \in \mathcal{Q}$. These matrices are computed as $\mathbf{C}_b = \frac{1}{\sum_{i=1}^{q} |S_i|} \sum_{i=1}^{q} \sum_{i \in S_j} (\mathbf{m}_j - \mathbf{m}_i)(\mathbf{m}_j - \mathbf{m}_i)^\top$ and $\mathbf{C}_w = \frac{1}{q} \sum_{j=1}^{q} \frac{1}{|Q_j|} \sum_{\mathbf{x}_i \in Q_j} (\mathbf{x}_i - \mathbf{m}_j)(\mathbf{x}_i - \mathbf{m}_j)^\top$, respectively, where $\mathbf{m}_j$ is the mean vector of Q_j. The optimal transformation matrix $\mathbf{A}$ is obtained by optimizing $J(\mathbf{A}) = \arg\max_{\mathbf{A}} \frac{|\mathbf{A}^\top \mathbf{C}_b \mathbf{A}|}{|\mathbf{A}^\top \mathbf{C}_w \mathbf{A}|}$.

3 A New Method for Acquiring Pairwise Constraints

The idea of our method is to identify good intra- and inter-cluster query-candidates given a data partition, and select the q most relevant candidates to prompt the user. The motivation for using a data partition as input relates to the importance of finding constraints sets with high informativeness, i.e., with high level of information that the clustering algorithm cannot determine on its own [8]. We also want to avoid performing redundant queries, i.e., queries involving similar pairs of objects. The details of the methods are given below. The proposed (dis)similarity-based constraint acquisition method consists of four phases:

1. Identify intra-cluster candidates. Pairs of objects which are far from each other have higher probability of having different labels than pairs of objects which are close. Therefore, the proposed method selects as candidates intra-cluster pairs of objects which are far apart. Given a distance matrix $\mathbf{D}$, c candidates Q_l are selected for each cluster $C_k \in P$ (more detail on defining c will be given later). Iteratively, the most distant pair of objects $(\mathbf{x}_i, \mathbf{x}_j)$ in C_k according to $\mathbf{D}$ is selected as candidate, i.e., $(\mathbf{x}_i, \mathbf{x}_j) = \arg\max_{\mathbf{x}_i \in C_k, \mathbf{x}_j \in C_k} \mathbf{D}(i, j)$, and is added to the set of query-candidates $Q = Q \cup Q_l$, $Q_l = \{(\mathbf{x}_i, \mathbf{x}_j)\}$. Then, $\mathbf{D}$ is updated such the distance between the objects belonging to the neighborhoods of $\mathbf{x}_i$ and $\mathbf{x}_j$ become 0, i.e., $\mathbf{D}(q, r) = 0$, $\mathbf{D}(r, q) = 0$, $\forall \mathbf{x}_q \in \mathcal{N}_i$, $\forall \mathbf{x}_r \in \mathcal{N}_j$ where $\mathcal{N}_l$ corresponds to the set of the m^{th} closest objects to $\mathbf{x}_l$ in C_k (including itself). The neighborhood size $1 \leq m \leq |C_k|$ is a parameter that should be defined as a compromise between selecting redundant (values close to 1) and non-interesting (values close to $|C_k|$) query-candidates. Note the $\mathcal{N}_i$ and $\mathcal{N}_j$ are computed using the original distance matrix. The process repeats until c candidates are found.

2. Identify inter-cluster candidates. Two objects in different clusters which are nearby have higher probability of belonging to the same *natural* cluster than objects which are distant. Hence, the algorithm selects as query-candidates pairs of objects in different clusters which are close. For each pair of clusters (C_l, C_o), $C_l \in P$, $C_o \in P$, $l < o$, c query-candidates are selected the following

way. First, the closest pair of objects $(\mathbf{x}_i, \mathbf{x}_j)$ in different clusters are selected, i.e., $(\mathbf{x}_i, \mathbf{x}_j) = \arg\min_{\mathbf{x}_i \in C_l, \mathbf{x}_j \in C_o} \mathbf{D}(i,j)$, and is added to Q. Next, the neighborhoods $\mathcal{N}_i$ and $\mathcal{N}_j$ are computed as the sets of the m_i^{th} and m_j^{th} closest objects to $\mathbf{x}_i$ in C_1 and $\mathbf{x}_j$ in C_l, and the distances between objects in distinct neighborhoods are set to ∞, i.e., $\mathbf{D}(q,r) = \infty$, $\mathbf{D}(r,q) = \infty$, $\forall \mathbf{x}_q \in \mathcal{N}_i$, $\forall \mathbf{x}_r \in \mathcal{N}_j$. Again, this will restrict the algorithm from choosing identical query-candidates. The procedure goes on until the desired number of candidates c is reached.

3. Rank candidates. This phase consists in ranking the candidates in descending order of interest. For this purpose a score is calculated for each candidate taking into account two situations: if the candidate has been obtained during the intra-cluster phase, the shorter the distance between one of its objects with any object from another cluster the more interesting the candidate is considered; if the candidate has been selected during the inter-cluster phase, the smaller the distance between the two objects of the query-candidate the higher the interest. Thus, the score S_l for each candidate $Q_l \in Q$, $(\mathbf{x}_i, \mathbf{x}_j) = Q_l$ is computed as $S_l = \min_{m:\mathbf{x}_m \in X \setminus C_{P_i}} \min\left[\mathbf{D}(i,m), \mathbf{D}(j,m)\right]$ if Q_l is an intra-candidate and $S_l = \mathbf{D}(i,j)$ otherwise. The sorted set of candidates Q^{sorted} is obtained by sorting the candidates according to their scores $\{S_i\}_{i=1}^{|Q|}$ in ascending order.

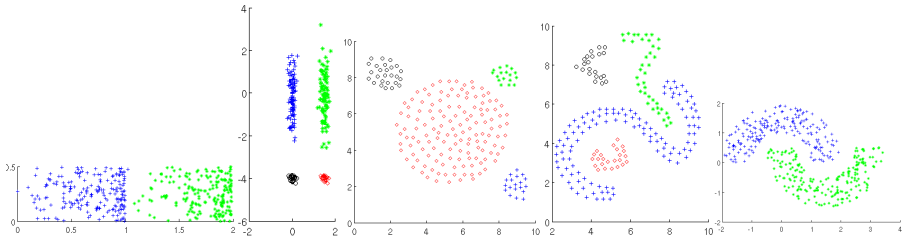
4. Query the user. Finally, the set of must-link and cannot-link constraints are obtained by querying the user if a pair of objects in a sorted query-candidate should belong to the same cluster, starting from the first query-candidate Q_1^{sorted} and stopping when the predefined number of queries q is obtained.

In this work, the number of candidates c for each intra- and inter-cluster search is the same (but it is not required). To ensure that at least q candidates are obtained (the number of candidates must be equal or higher than the number of desired queries) the following inequality must hold: $q \leq ck + c\frac{k(k-1)}{2}$, where k is the number of clusters. Thereby, $c \geq \lceil \frac{2q}{k^2+k} \rceil$. It is usually helpful to generate more candidates that the strictly required because some clusters are more interesting than others (e.g. touching clusters). By doing so, candidates from non-interesting regions will naturally be ruled out in the ranking phase.

The obtained constraint set can be used by a constrained clustering algorithm using the original data representation to partition the data. However, we hypothesize that using a learned space which represents both the original data and the constraints (e.g. DCA) can further enhance clustering quality.

4 Experimental Results

In our experiments, 5 synthetic data sets (shown in figure 1) and 5 real data sets taken from the UCI ML repository (http://archive.ics.uci.edu/ml/) were used to assess the performance of the constraint acquisition approach. A brief description for each real data set is given next. The Iris data set consists of 50 objects from each of three species of Iris flowers (setosa, virginica and versicolor) characterized by four features. The Breast Cancer data set is composed of 683 domain objects characterized by nine features and divided into two clusters: benign and malignant. The Optdigits is a subset of Handwritten Digits data set



(a) Bars (b) Cigar (c) D1 (d) D2 (e) Half Rings

Fig. 1. Synthetic data sets

containing only the first 100 objects of each digit, from a total of 3823 domain objects characterized by 64 attributes. The House Votes data set is composed of two clusters of votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the Congressional Quarterly Almanac (125 democrats and 107 republicans). The Wine data set consists of the results of a chemical analysis of wines divided into three clusters with 59, 71 and 48 objects, described by 13 features.

Table 1 shows the average accuracy and standard deviation of the partitions obtained using CAL algorithm (K was set as the real number of clusters) using no constraints, constraints acquired using RAC, Explore-Consolidate and the proposed method, with the original data representation and a learned data representation obtained using DCA (identified with "+L"). The average values were computed over 50 repetitions using data resampling with replacement (the size of the samples corresponds to the original size of the data sets). Constraints sets were obtained by performing 10, 20, 30, 40 and 50 queries. Answers were given using ground-truth information. The number of clusters was defined as the "natural" number of cluster for each data set. The number of candidates for each intra-cluster and inter-cluster searches was defined as $c = \lceil 2 \times \frac{2q}{k^2+k} \rceil$, and the size of the neighborhood of an object $\mathbf{x}_i \in C_k$ was set to $m = \lceil 0.35|C_k| \rceil$. The partitions obtained using the original data representation with constraints were usually (not always) better than the ones produced without constraints. Also, the average accuracy (percent of correctly clustered objects) of the proposed method and the Explore-Consolidate were generally superior than the accuracy achieved by the RAC method. However, by comparing our method with the Explore-Consolidate no method was clearly a winner. The results achieved by using a learned data representation representing both the constraint sets and the original data representation were usually superior than the results obtained using the original representation. The distance learning algorithm obtained better data representations with the proposed and Explore-consolidate approaches than the random acquisition of constraints, since the corresponding partitions are more accurate.

Table 1. Accuracy for CAL using no constraints, the constraints acquired by RAC, Explore-Consolidate and the proposed method, with the original and learned representations

Data set	Acquisition Method	Number of queries				
		10	20	30	40	50
Bars	No const			97.48 (2.98)		
	Random	97.64 (2.76)	97.78 (2.70)	97.94 (2.28)	98.03 (2.36)	**98.76** (1.25)
	Expl.Consol.	97.28 (3.07)	**97.83** (2.70)	**98.07** (2.25)	**98.10** (2.28)	98.52 (1.59)
	Proposed	**97.66** (2.79)	**97.83** (2.81)	98.03 (2.76)	98.09 (2.70)	98.23 (2.53)
	Random+L	95.10 (8.38)	97.28 (2.40)	97.59 (2.30)	98.02 (1.87)	97.99 (1.98)
	Expl.Consol.+L	96.36 (3.58)	97.45 (3.17)	**98.06** (2.44)	98.09 (2.54)	**98.56** (1.74)
	Proposed+L	**97.52** (2.96)	**98.01** (2.34)	98.04 (2.48)	**98.11** (2.60)	98.17 (2.37)
Half Rings	No const			83.08 (5.78)		
	Random	82.26 (6.32)	82.23 (5.60)	82.62 (5.76)	81.65 (6.24)	81.79 (5.86)
	Expl.Consol.	**82.86** (6.34)	81.98 (5.16)	**82.93** (6.60)	**86.13** (6.58)	**86.39** (7.53)
	Proposed	82.59 (6.23)	**83.47** (6.03)	82.05 (5.65)	82.29 (5.52)	83.53 (5.90)
	Random+L	84.15 (7.08)	86.84 (6.05)	**89.67** (5.32)	89.07 (4.50)	88.93 (5.10)
	Expl.Consol.+L	81.21 (4.27)	82.14 (3.17)	83.01 (3.41)	83.44 (3.15)	84.61 (2.70)
	Proposed+L	**85.96** (6.90)	**89.01** (3.69)	89.45 (3.44)	**89.58** (3.90)	**89.87** (2.77)
Cigar	No const			75.06 (14.09)		
	Random	75.48 (14.13)	75.87 (14.56)	78.32 (14.14)	77.30 (14.10)	78.90 (13.46)
	Expl.Consol.	74.77 (15.12)	**79.52** (12.59)	79.68 (13.25)	**83.96** (10.21)	**87.18** (10.07)
	Proposed	**75.63** (13.31)	78.23 (12.67)	**81.51** (11.77)	82.11 (10.79)	84.96 (8.70)
	Random+L	**91.67** (11.00)	94.90 (6.04)	93.07 (6.96)	94.79 (5.89)	97.05 (4.88)
	Expl.Consol.+L	86.11 (7.66)	94.02 (7.25)	96.69 (4.83)	98.11 (3.92)	98.22 (2.99)
	Proposed+L	91.58 (9.46)	**96.02** (5.72)	**98.02** (3.82)	**98.67** (3.10)	**99.17** (1.93)
D1	No const			68.78 (13.56)		
	Random	68.99 (14.45)	73.60 (14.27)	77.47 (15.15)	79.48 (14.34)	**85.21** (13.41)
	Expl.Consol.	**72.94** (14.36)	**75.72** (13.60)	**82.46** (11.96)	**80.34** (12.22)	82.74 (12.46)
	Proposed	71.42 (14.37)	70.81 (14.50)	72.79 (14.51)	76.60 (13.90)	76.70 (13.09)
	Random+L	67.82 (8.58)	71.82 (10.93)	74.79 (12.10)	78.65 (15.57)	81.11 (14.38)
	Expl.Consol.+L	**69.55** (11.13)	**76.53** (10.39)	**78.58** (11.14)	**81.98** (10.44)	**82.73** (10.22)
	Proposed+L	63.13 (12.83)	70.29 (10.40)	72.45 (14.33)	74.52 (12.43)	77.33 (12.94)
D2	No const			55.12 (6.84)		
	Random	53.45 (6.19)	53.67 (6.02)	53.60 (4.73)	54.04 (4.70)	53.90 (4.43)
	Expl.Consol.	53.64 (5.14)	52.00 (5.13)	54.96 (8.15)	52.77 (7.17)	55.15 (8.10)
	Proposed	**55.50** (7.14)	**55.63** (7.14)	**56.09** (7.69)	**56.71** (8.26)	**57.18** (7.97)
	Random+L	52.83 (6.38)	53.47 (5.85)	50.88 (5.53)	53.08 (4.63)	53.29 (5.11)
	Expl.Consol.+L	53.11 (4.74)	52.91 (5.32)	53.54 (5.90)	53.82 (6.07)	**58.19** (8.46)
	Proposed+L	**55.16** (7.29)	**55.43** (7.05)	**56.48** (7.99)	**57.60** (7.96)	58.02 (7.60)
Crabs	No const			53.08 (2.31)		
	Random	52.85 (2.20)	53.57 (2.89)	53.43 (2.45)	53.12 (2.11)	53.47 (2.88)
	Expl.Consol.	53.53 (2.74)	**54.03** (2.71)	**54.59** (3.15)	**56.17** (3.56)	**59.04** (3.94)
	Proposed	**53.70** (2.50)	53.91 (2.98)	53.26 (2.14)	54.00 (2.84)	54.14 (3.26)
	Random+L	**65.74** (14.95)	65.69 (16.10)	**72.01** (17.14)	68.74 (18.14)	**72.62** (18.83)
	Expl.Consol.+L	54.20 (3.22)	54.78 (4.13)	57.59 (7.72)	59.46 (5.71)	60.98 (5.62)
	Proposed+L	62.24 (12.98)	**70.12** (14.51)	70.14 (16.45)	**72.64** (17.06)	66.75 (17.36)
House Votes	No const			89.22 (2.49)		
	Random	89.41 (2.92)	89.16 (2.84)	90.07 (2.98)	89.55 (3.34)	90.32 (3.26)
	Expl.Consol.	89.32 (2.64)	89.73 (2.41)	**90.51** (2.35)	90.47 (2.35)	**90.92** (2.06)
	Proposed	**89.62** (2.55)	**90.03** (2.45)	90.38 (2.73)	**90.48** (2.84)	90.78 (2.56)
	Random+L	67.36 (15.42)	67.71 (16.53)	66.97 (18.10)	65.34 (18.37)	63.58 (18.67)
	Expl.Consol.+L	**89.02** (2.42)	**89.47** (2.53)	**90.09** (2.25)	**90.45** (2.38)	**90.51** (1.91)
	Proposed+L	69.14 (17.32)	75.77 (18.10)	74.78 (20.03)	80.68 (18.84)	82.22 (19.95)
Wine	No const			60.98 (5.87)		
	Random	**60.96** (5.94)	62.04 (5.94)	62.37 (6.41)	62.36 (5.23)	63.63 (5.44)
	Expl.Consol.	59.94 (5.80)	60.99 (5.58)	61.51 (5.57)	63.34 (5.75)	**65.11** (6.12)
	Proposed	60.82 (5.67)	**62.54** (6.05)	**63.76** (5.60)	**64.47** (5.12)	64.47 (4.73)
	Random+L	51.17 (11.51)	47.09 (8.54)	49.28 (10.92)	54.97 (14.03)	58.47 (12.91)
	Expl.Consol.+L	59.30 (12.05)	**63.54** (7.75)	**68.42** (7.27)	**69.06** (6.39)	**72.13** (8.90)
	Proposed+L	**60.53** (11.75)	57.24 (11.55)	58.22 (11.66)	59.74 (9.68)	58.78 (11.66)
Iris	No const			78.40 (10.18)		
	Random	78.16 (9.92)	78.15 (9.94)	79.60 (10.42)	79.04 (10.89)	82.11 (10.78)
	Expl.Consol.	80.73 (10.05)	**87.12** (8.48)	**90.15** (7.44)	**92.03** (5.27)	**93.24** (3.21)
	Proposed	**81.99** (9.72)	83.61 (9.82)	85.27 (8.44)	87.56 (8.27)	87.87 (7.70)
	Random+L	61.69 (22.48)	77.41 (18.44)	82.65 (12.52)	85.67 (13.60)	85.97 (13.33)
	Expl.Consol.+L	81.01 (13.43)	82.53 (12.41)	90.45 (9.45)	93.65 (5.15)	94.64 (3.21)
	Proposed+L	**85.43** (13.58)	**89.45** (11.51)	**93.23** (8.60)	**94.39** (8.37)	**96.49** (4.75)
Breast Cancer	No const			95.36 (1.48)		
	Random	95.07 (1.61)	95.41 (1.48)	95.43 (1.56)	95.45 (1.48)	95.58 (1.53)
	Expl.Consol.	95.17 (1.57)	95.53 (1.57)	95.32 (1.62)	95.54 (1.26)	95.67 (1.47)
	Proposed	**95.53** (1.53)	**95.70** (1.44)	**95.80** (1.44)	**95.75** (1.42)	**95.76** (1.56)
	Random+L	80.52 (12.61)	74.54 (9.27)	72.98 (8.57)	71.23 (8.21)	69.09 (5.94)
	Expl.Consol.+L	**95.80** (4.84)	**96.24** (1.66)	**95.84** (1.94)	**96.37** (1.48)	**96.18** (1.50)
	Proposed+L	76.40 (11.91)	74.87 (12.18)	76.03 (12.74)	76.13 (13.28)	76.50 (13.81)

5 Conclusions

We proposed a new constraint acquisition method for constrained data clustering which identifies intra- and inter-cluster query-candidates, ranks them by decreasing order of relevance and uses the most interesting candidates to query the user. We assessed the proposed method against not using constraints at all, using random constraints, and using the Explore-Consolidate approach. Results shown the use of constraints obtained using the proposed and the Explore-Consolidate methods both results in better partitions than using random constraints or not using constraints at all. The use of data representations obtained from the set of constraints and the original data usually increases the clustering performance.

Acknowledgements. This work is supported by FCT "Fundação para a Ciência e a Tecnologia" under the project "LearningS" - PTDC/EEI-SII/2312/2012.

References

1. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern Recogn. Lett. **31**(8) (June 2010) 651–666
2. Wagstaff, K.L.: Intelligent clustering with instance-level constraints. PhD thesis, Ithaca, NY, USA (2002) Chair-Claire Cardie.
3. Basu, S.: Semi-supervised clustering: probabilistic models, algorithms and experiments. PhD thesis, Austin, TX, USA (2005) Supervisor-Mooney, Raymond J.
4. Davidson, I., Ravi, S.: Clustering with constraints feasibility issues and the k-means algorithm. In: 2005 SIAM International Conference on Data Mining (SDM'05), Newport Beach,CA (2005) 138–149
5. Tung, A.K.H., Hou, J., Han, J.: Coe: Clustering with obstacles entities. a preliminary study. In: PADKK '00: Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications, London, UK, Springer-Verlag (2000) 165–168
6. Béjar, J., Cortés, U.: Experiments with domain knowledge in unsupervised learning: Using and revising theories. Revista Iberoamericana de Computación. Computación y Sistemas **1**(3) (1998) 136–144
7. Ge, R., Ester, M., Jin, W., Davidson, I.: Constraint-driven clustering. In: KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2007) 320–329
8. Davidson, I., Wagstaff, K.L., Basu, S.: Measuring constraint-set utility for partitional clustering algorithms. In: Proc. 10th European Conference on Principles and Practice of Knowledge Discovery in Databases. (2006)
9. Huang, S.J., Jin, R., Zhou, Z.H.: Active learning by querying informative and representative examples. Advances in neural information processing systems **23** (2010) 892–900
10. Jain, P., Kapoor, A.: Active learning for large multi-class problems. In: IEEE Conference on Computer Vision and Pattern Recognition. (2009) 762–769

11. Basu, S., Banjeree, A., Mooney, E., Banerjee, A., Mooney, R.J.: Active semi-supervision for pairwise constrained clustering. In: In Proceedings of the 2004 SIAM International Conference on Data Mining (SDM-04. (2004) 333–344
12. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.J.: Distance metric learning with application to clustering with side-information. In Becker, S., Thrun, S., Obermayer, K., eds.: NIPS, MIT Press (2002) 505–512
13. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning a mahalanobis metric from equivalence constraints. J. Machine Learning Research **6**(1) (2006) 937
14. Hoi, S., Liu, W., Lyu, M., Ma, W.Y.: Learning distance metrics with contextual constraints for image retrieval. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. Volume 2. (2006) 2072–2078
15. Duarte, J.M.M., Fred, A.L.N., Duarte, F.J.F.: Evidence accumulation clustering using pairwise constraints. In: Proceedings of the International Conference on Knowledge Discovery and Information Retrieval. (2012) 293–299
16. Sokal, R.R., Michener, C.D.: A statistical method for evaluating systematic relationships. University of Kansas Scientific Bulletin **28** (1958) 1409–1438