A Set Correlation Model for Partitional Clustering

Nguyen Xuan Vinh[1,2,3] and Michael E. Houle[3]

[1] School of Electrical Engineering and Telecommunications
The University of New South Wales, Sydney, NSW 2052, Australia
[2] National ICT Australia (NICTA)
[3] National Institute of Informatics,Tokyo, Japan
n.x.vinh@unsw.edu.au, meh@nii.ac.jp

Abstract. This paper introduces GlobalRSC, a novel formulation for partitional data clustering based on the Relevant Set Correlation (RSC) clustering model. Our formulation resembles that of the K-means clustering model, but with a shared-neighbor similarity measure instead of the Euclidean distance. Unlike K-means and most other clustering heuristics that can only work with real-valued data and distance measures taken from specific families, GlobalRSC has the advantage that it can work with any distance measure, and any data representation. We also discuss various techniques for boosting the scalability of GlobalRSC.

Keywords: Clustering, correlation, shared neighbor, RSC, SASH.

1 Introduction

Clustering is the art of partitioning a data set into groups such that objects from the same group are as similar as possible, and objects from different groups are well differentiated. To support clustering, a measure of similarity (or distance) between data objects is needed. Popular distance measures for clustering include the class of general L_p norms (which includes the Euclidean distance L_2), and the cosine similarity measure. The k objects most similar to a data item v are often referred to as the k-nearest-neighbor (k-NN) set of v.

An interesting and appealing class of 'secondary' similarity measures, the so-called *shared-neighbor* (SN) measures, can be derived from any other ('primary') similarity measure. SN measures typically are expressed as a function of the intersection size of the k-NN sets of the two objects whose similarity is to be computed, where the neighborhoods are computed using the primary similarity measure. The use of SN-based similarity in clustering can be traced back to the merge criterion of the agglomerative algorithm due to Jarvis and Patrick [1]. Other agglomerative clustering methods with SN-based merge criteria include the hierarchical algorithm ROCK [2] and the density-based algorithm SNN (Shared Nearest Neighbor) [3]. SNN is essentially an improved version of the well-known DBSCAN [4] clustering algorithm; like DBSCAN, SNN is able

M.J. Zaki et al. (Eds.): PAKDD 2010, Part I, LNAI 6118, pp. 4–15, 2010.

to produce clusters of different sizes, shapes and densities. However, the performance of SNN greatly depends on the tuning of several non-intuitive parameters by the user. In practice, it is difficult (if not impossible) to determine appropriate values for these parameters on real datasets.

A common requirement of most SN-based clustering methods is that a fixed neighborhood size — either in terms of the number of neighbors k as in SNN and in Jarvis & Patrick's method, or in terms of the radius r of the neighborhood ball as in ROCK — needs to be chosen in advance, and then applied equally to all items of the data set. However, fixed choices of neighborhood sizes k or radius r are known to lead to bias in the clustering process [5], the former with respect to the sizes of the clusters discovered, and the latter with respect to the density of the regions from which the clusters are produced.

Recently, an SN-based clustering model was proposed that allows the sizes of the neighborhoods to vary. The *Relevant Set Correlation* (RSC) model [5] defines the relevance of a data item v to a cluster C in terms of a form of 'set correlation' between the memberships of $|C|$ and the $|C|$-nearest-neighbor set of v. RSC quality measures can be used to evaluate the relative importance of cluster candidates of various sizes, avoiding the problems of bias found with other shared-neighbor methods that use fixed neighborhood sizes or radii. The same paper introduced a clustering algorithm based on RSC, called GreedyRSC, that generates cluster candidates in the vicinity of every object of the dataset, evaluates the quality of the candidates according to the model, and greedily selects them in decreasing order of their quality. GreedyRSC is a 'soft' clustering algorithm, in that the clusters produced are allowed to overlap. It does not require that the user choose the neighborhood size or specify a target number of clusters; instead, the user simply specifies the minimum allowable cluster size, and the maximum allowable correlation between any two clusters. Unlike many other clustering algorithms, GreedyRSC uses only local criteria for the formation of cluster candidates — the clustering process is not guided by a global optimization criterion.

In this paper, we present a new RSC-based partitional clustering algorithm, *GlobalRSC*, that allows the user to specify the number of clusters to be generated, K. Unlike GreedyRSC, GlobalRSC emulates the well-known K-means clustering algorithm in that it seeks to optimize a global objective function. Given an initial clustering configuration, both K-means and GlobalRSC attempt to optimize their objective function through an iterative hill-climbing improvement process. GlobalRSC, however, replaces the Euclidean distance of K-means by a shared-neighbor similarity measure, and can therefore be applied (in principle) to any form of data, and using any appropriate similarity measure.

This paper is organized as follows. In Section 2 we review those elements of the RSC model upon which GlobalRSC is based, and introduce the GlobalRSC clustering criterion. In Section 3, we give the details of the GlobalRSC clustering algorithm. Experimental results are presented in Section 4, and concluding remarks appear in Section 5.

2 Shared Neighbor Similarity and the RSC Model

In this section, we present a brief introduction to the Relevant Set Correlation (RSC) model for clustering, and the set correlation similarity measure upon which it is based.

Let S be a dataset of $|S| = n$ data items $\{s_1, s_2, \ldots, s_n\}$. Any subset A of S can then be represented as a n-dimensional zero-one characteristic vector, where the value of the i-th coordinate is 1 if and only if $s_i \in A$. The simplest SN-based similarity measure between the two sets A and B is the 'overlap' or intersection size $|A \cap B|$, which can be expressed as the inner product between the two characteristic vectors. Another popular measure, the cosine similarity $\cos(A, B) \triangleq \frac{|A \cap B|}{\sqrt{|A||B|}}$, is the inner product of the normalized characteristic vectors for A and B, which in turn equals the cosine of the angle between them. Values of the cosine measure lie in the range $[0, 1]$, with 1 attained whenever A is identical to B, and 0 whenever A and B are disjoint.

In [5], the *set correlation* measure was proposed as the value of the Pearson correlation between the coordinate pairs of characteristic vectors of A and B. After derivation and simplification, this expression becomes:

$$R(A, B) \triangleq \frac{|S|}{\sqrt{(|S| - |A|)(|S| - |B|)}} \left(\cos(A, B) - \frac{\sqrt{|A||B|}}{|S|} \right). \tag{1}$$

Values of the set correlation lie in the range $[-1, 1]$. A value of 1 indicates that A and B are identical, and a value of -1 indicates that A and B are complements of each other in S.

Despite their simplicity and their popularity in practice, the overlap and cosine measures both have the disadvantage of bias relative to the sizes of the two sets A and B. To see this, let A be fixed with size $|A| = a$, and let B be selected uniformly at random from the items of S with the constraint that $|B|$ equals some fixed value $b > 0$. Let $a = |A|$. Under these assumptions, the overlap is known to be hypergeometrically distributed with expected value $\mathbf{E}[|A \cap B|] = \frac{ab}{n}$, and the expected value of the cosine measure is therefore $\mathbf{E}[\cos(A, B)] = \frac{\mathbf{E}[|A \cap B|]}{\sqrt{ab}} = \frac{\sqrt{ab}}{n}$. When used to rank the similarity of sets with respect to A, both measures are biased towards sets B of larger sizes. On the other hand, the expected value of the set correlation under the same assumptions can be shown to be $\mathbf{E}[R(A, B)] = 0$, indicating no bias with respect to the sizes of A and B. Therefore, of these three SN-based similarity measures, the set correlation measure is best suited for those applications in which the neighborhood size is variable.

Under the RSC model, the quality of a given cluster candidate set A is assessed in terms of the set correlation between the candidate and neighborhood sets (the 'relevant sets') based at its members. Let Q_k^v denote the set of k-nearest neighbors of v with respect to S. The RSC model uses the set correlation $R(Q_{|A|}^v, A)$ between $Q_{|A|}^v$ and A as a measure of relevance of item v to the cluster candidate A. Note that in this formulation, the neighborhood size is taken to be

the cardinality of A. This definition eliminates the need for specifying a fixed neighborhood size, and avoids the bias associated with such choices.

For more details concerning the quality measures of the RSC model, see [5].

3 The GlobalRSC Clustering Algorithm

3.1 GlobalRSC and *K*-Means

The GlobalRSC clustering criterion has the same general form as that of K-means. In the standard K-means formulation, a partition $\mathcal{A} = \{A_1, A_2, \ldots, A_K\}$ of the data set is sought which maximizes the following objective function:

$$\mathcal{D}(\mathcal{A}) = \frac{1}{n} \sum_{i=1}^{K} \sum_{v \in A_i} D(v, c(A_i)), \tag{2}$$

where $c(A)$ is a function which returns the center of mass of a cluster A (computed as $c(A) = \frac{1}{|A|} \sum_{v \in A} v$), and the distance measure D is generally taken to be the square of the Euclidean distance. The proposed formulation of GlobalRSC replaces the distance measure $D(v, c(A_i))$ by the average set correlation between cluster A_i and the neighborhood $Q^v_{|A_i|}$ based at v, as follows:

$$\mathcal{R}(\mathcal{A}) = \frac{1}{n} \sum_{i=1}^{K} \sum_{v \in A_i} R(Q^v_{|A_i|}, A_i). \tag{3}$$

Both D and R serve as measures of the relevance of an item to its assigned cluster. However, unlike R, D can only be computed when the data can be represented as real-valued vectors. As discussed earlier, the use of set correlation in the formulation of GreedyRSC is preferred over that of the overlap or cosine measure, due to the bias of the latter measures with respect to set sizes.

3.2 A Hill-Climbing Heuristic

The problem of optimizing the GlobalRSC objective function (3) greatly resembles that of optimizing the K-means objective function (2). Despite its simple appearance, the K-means clustering problem with squared Euclidean distance is known to be NP-hard even for $K = 2$ [6]. Although the hardness of the GlobalRSC optimization problem is still an open question, a heuristic approach seems to be indicated.

In this section, we propose an iterative hill-climbing solution, which we simply refer to as *GlobalRSC*. The core idea is as follows: at each round the algorithm iterates through the items of S looking for items whose reassignment to a different cluster leads to an improvement in the value of the objective function $\mathcal{R}$. As is the case with K-means, two reassignment schemes can be employed for GlobalRSC: *incremental update*, in which reassignment is performed as soon as an improvement is detected, and *batch update*, in which all the membership changes are

applied only at the end of each round, when the algorithm has completed a full iteration through all data items. The advantage of the batch update scheme is that the recomputation of $\mathcal{R}$ can be performed very efficiently through the use of *inverted neighborhood sets* (as defined in Fig. 1). However, it is possible that delaying the reassignment of items until the end of each round could result in a decrease in the value of the objective function, even if each reassignment would have led to an increase if it were applied individually. In practice we often observe that first few rounds of the batch update scheme quickly improves the objective value. It would therefore be beneficial to begin with several rounds of batch updating, followed by an incremental update phase to further refine the clustering.

It should be noted that in an incremental reassignment of v from cluster A_i to cluster A_j, the contributions $\mathcal{R}(A) = \sum_{w \in A} R(Q^w_{|A|}, A)$ to $\mathcal{R}$ for an individual cluster A do not need to be recomputed except for $A = A_i$ and $A = A_j$. To verify whether the reassignment would increase the value of $\mathcal{R}$, it suffices to perform the test $\mathcal{R}(A_j \cup \{v\}) + \mathcal{R}(A_i \setminus \{v\}) - \mathcal{R}(A_j) - \mathcal{R}(A_i) > 0$.

The recomputation of $\mathcal{R}$ after the reassignment of a single item v would be relatively expensive if all $K - 1$ possible reassignments were considered. We therefore limit the tentative reassignment of v to those candidate clusters found in the vicinity of v; that is, those clusters containing at least one element in the neighborhood $Q^v_{|A_i|}$, where A_i is the cluster to which v currently belongs. If one of these tentative reassignments of v would result in an increase in the value of $\mathcal{R}$, then the reassignment that results in the greatest such increase is applied. Otherwise, v is not reassigned. If the size of the new cluster A_j is larger than that of the currently-stored neighborhood of v, then that neighborhood would need to be expanded. Accordingly, whenever it is necessary to recompute a neighborhood for item $v \in A_j$, we choose the size to be $\min\{\lceil (1+b)|A_j| \rceil, m\}$ for some fixed real parameter values $b > 0$ and $m > 0$. In our implementation of GlobalRSC, b is set to a default value of 0.5, and m is set to 50.

A pseudocode description of the basic GlobalRSC heuristic is shown in Fig. 1. The heuristic can easily be shown to converge within a finite number of steps.

3.3 Complexity Analysis

The algorithm requires storage for the neighbor lists of all n data items, each of which has size proportional to that of the cluster to which it has been assigned. The total space required is of order $\sum_{i=1}^{K} |A_i|^2$. Let μ and σ be respectively the mean and standard deviation of the cluster sizes; in terms of μ and σ, the space required is proportional to $K(\sigma^2 + \mu^2)$.

At the initialization step, the neighborhood list for each data item must be calculated. A straightforward implementation requires the computation of $O(n^2)$ distances. Once computed, these distances can also be used to generate an initial clustering. Since the neighborhoods must be constructed in sorted order, the total time required for preparing neighborhoods is $\sum_{i=1}^{K} |A_i|(|A_i| + |A_i| \log |A_i|)$ $= O((\sigma^2 + \mu^2)K \log n)$ using linear-time methods for determining order statistics [7]. The total time required for initialization is thus $O(dn^2 + (\sigma^2 + \mu^2)K \log n)$, where d is the cost of computing a single distance.

Input: The data set S; the number of desired clusters K; (optionally) a hard initial clustering $\mathcal{A} = \{A_1, A_2, \ldots, A_K\}$ on S; minimum neighborhood set size m; neighborhood set buffer size b.

0. **Initialization:** If no initial clustering was provided, compute an initial clustering $\mathcal{A} = \{A_1, A_2, \ldots, A_K\}$ as follows:
   (a) Select K items of S uniformly at random as the seeds of the K clusters.
   (b) Assign each data item v to the cluster corresponding to the seed closest to v.
   (c) Build the neighborhood Q^v for each item $v \in A_i$, with size equal to $\min\{\lceil(1+b)|A_i|\rceil, m\}$.
1. **Batch phase:**
   (a) Calculate $\mathcal{R}(A_1), \mathcal{R}(A_2), \ldots, \mathcal{R}(A_K)$ and set the termination flag $halt \leftarrow$ FALSE.
   (b) Repeat until $halt$ = TRUE:
      i. Set $halt \leftarrow$ TRUE.
      ii. For each $v \in S$, build the *inverted neighborhood* I^v, where $r \in I^v$ if and only if $v \in Q^r$.
      iii. For each data item v currently in cluster A_i:
         A. Build the list $\mathfrak{C}$ of clusters (other than A_i) containing at least one item of Q^v.
         B. Tentatively reassign v to each of the A_j in $\mathfrak{C}$.
         C. Using the inverted neighborhood sets, calculate the index j for which the improvement value $\mathcal{R}(A_j \cup \{v\}) + \mathcal{R}(A_i \backslash \{v\}) - \mathcal{R}(A_j) - \mathcal{R}(A_i)$ is maximized.
         D. If the improvement value is positive, record (v, j) for future reassignment.
      iv. Tentatively apply all the recorded reassignments for this round, and let $\mathcal{A}' = \{A'_1, A'_2, \ldots, A'_K\}$ be the resulting clustering. Calculate $\mathcal{R}(\mathcal{A}')$. If $\mathcal{R}(\mathcal{A}') > \mathcal{R}(\mathcal{A})$, set $halt \leftarrow$ FALSE and $\mathcal{A} \leftarrow \mathcal{A}'$. Otherwise, proceed to the incremental phase.
2. **Incremental phase:**
   (a) Set $halt \leftarrow$ FALSE.
   (b) Repeat until $halt$ = TRUE:
      i. Set $halt \leftarrow$ TRUE.
      ii. For each data item v currently in cluster A_i:
         A. Build the list $\mathfrak{C}$ of clusters (other than A_i) that contribute items to Q^v.
         B. Tentatively reassign v to each of the A_j in $\mathfrak{C}$, and calculate the index j for which the improvement value $\mathcal{R}(A_j \cup \{v\}) + \mathcal{R}(A_i \backslash \{v\}) - \mathcal{R}(A_j) - \mathcal{R}(A_i)$ is maximized.
         C. If the improvement value is positive, reassign v to A_j immediately, adjust the values of $\mathcal{R}(A_j)$ and $\mathcal{R}(A_i)$, and set $halt \leftarrow$ FALSE.

Fig. 1. A pseudocode description of the basic GlobalRSC variant

During each round of the batch phase, building the inverted neighbor sets requires that the values of $K(\sigma^2 + \mu^2)$ integer variables be copied. Recalculating $\mathcal{R}$ for the tentative reassignment of item v from cluster A_i to cluster A_j requires time proportional to $|A_i| + |A_j| + 2|I_v|$ when using the inverted neighbor lists. Assuming that v needs to be tentatively reassigned to each of the other $K - 1$ clusters, the cost of reassignment is $O(n)$. Adding the cost over all choices of v, the total cost of reassignment per phase is at most $O(n^2)$. The neighbor list can be reconstructed in $O(n \log n + nd)$ time if required; the total cost of reconstruction will be no worse than that of initialization, which is $O(dn^2 + (\sigma^2 + \mu^2)K \log n)$. The worse case complexity of each batch phase iteration through the data set is therefore $O(dn^2 + (\sigma^2 + \mu^2)K \log n)$. However, in practice we expect a much lower time cost, since there are typically only a limited number of nearby clusters for each data item, and few if any neighborhoods require reconstruction.

In the incremental phase, tentatively moving an item v from a cluster A_i to another A_j requires $O(|A_i|^2 + |A_j|^2)$ operations for the direct recalculation of the objective function. If v is tentatively reassigned to each of the other $K - 1$ clusters, the cost is $O(K(\sigma^2 + \mu^2))$; over all possible choices of v (one full round), the total cost of reassignment becomes $O(nK(\sigma^2 + \mu^2))$. Reconstruction of the neighborhood for each item, if required, can be performed in

$O(dn^2 + n^2 \log n)$ total time. The worst case complexity of a full round is therefore $O(dn^2 + n^2 \log n + nK(\sigma^2 + \mu^2))$. However again, the observed cost should be much lower in practice. Since the incremental phase is often considerably more expensive than the batch phase, to improve time efficiency for large data sets, we employ a 'reduced' variant of the incremental phase, in which a data item v is considered for reassignment to another cluster if and only if that cluster already contains the majority of the neighbors of v. This variant scheme focuses on items with neighborhoods of low consistency, with the worst case complexity of each round being reduced to that of the batch phase.

In practice, the standard deviation of the cluster size, σ, is typically of the same order of the mean cluster size $\mu = n/K$, leading to an overall space complexity of $\widetilde{O}(n^2/K)$, and a time complexity (for the reduced incremental phase variant) of $\widetilde{O}(dn^2 + n^2(\log n)/K)$. Optionally, if a full distance matrix is to be stored in order to speed up neighborhood list computation, then the space complexity would attain its worst-case value of $\Theta(n^2)$.

3.4 Scalability

In this section we present several techniques that can boost the scalability of GlobalRSC for large, high dimensional data sets. The challenges faced by GlobalRSC (and other SN-based clustering algorithms) as the dimensionality increases are: (i) the construction of neighborhoods becomes more expensive, due to an effect known as the 'curse of dimensionality' [8]; (ii) the optimization of the objective function becomes more difficult, as local optimization approaches such as hill-climbing are more easily trapped at local maxima that may be far from the global optimum.

In order to accelerate the construction of neighborhoods, we propose the use of the Spatial Approximation Sample Hierarchy (SASH) developed in [8].

If the data set contains many large clusters, the calculation of set correlation scores with respect to these clusters may be prohibitively expensive. One of the simplest ways of avoiding the high costs associated with large cluster candidates is through the restriction of neighborhood sizes. In our implementation of GlobalRSC, we restricted the maximum size of the neighborhood to be 1000. Only the average relevance score for items in cluster of size smaller than this threshold is calculated exactly, while the membership of larger clusters are 'frozen'. Items are permitted to be reassigned from smaller clusters to frozen clusters and vice-versa, as long as such a movement increases the average relevance score of the smaller clusters.

4 Experimental Results

In this section we report the results of our experiments on various real data sets taken from several domains. GlobalRSC, implemented in C++ and tested on a Pentium IV 3.2GHz workstation equipped with 4Gb of main memory, was compared against a MATLAB implementation of 'Fast' K-means [9] (available from

the author's website), and the 'bisecting' K-means algorithm available as part of the CLUTO clustering toolkit [10]. CLUTO was run with its default distance measure, the cosine similarity, using repeated bisecting clustering followed by a final global optimization phase. For large data sets, we used GreedyRSC to initialize GlobalRSC, as well as the SASH to speed up neighborhood construction. We report the mean and standard deviation values of the quality metrics, plus the average execution time and the number of iterations performed by each algorithm (if known). We did not include the SNN clustering algorithm, due to the difficulty in tuning its parameters, and since it often leaves a large number of items unassigned to any cluster. For interested readers, a comparison between SNN, K-means and GreedyRSC on several data sets can be found in [5].

For each of the data sets considered, the clustering results are assessed against a ground-truth classification according to 5 different quality measures: the well-known Adjusted Rand Index (ARI) from statistics [11]; the recently developed Adjusted Mutual Information (AMI) [12] from information theory; the Expected Precision (EPrec), Recall (ERec) and Cosine (ECos) measures [5] from information retrieval. For all these measures, higher values represent better clusterings, with a maximum possible value of 1. A low expected precision score is an indication of cluster fusion, occurring when too few clusters are produced, whereas a low expected recall indicates cluster fragmentation, occurring when too many clusters are generated. A high expected cosine score can be taken as evidence that the clustering avoids extremes of cluster fusion and cluster fragmentation. Due to lack of space, we do not give details of these measures here. Interested readers are referred to the original publications for more information.

4.1 Biological Data

We tested the algorithms on several gene expression microarray data sets:

- **B1:** This set consists of 384 genes whose expression level peak at different time points corresponding to the five phases of a cell cycle [13].
- **B2:** This set consists of 237 genes corresponding to four categories in the MIPS database. The four categories (DNA synthesis and replication, organization of centrosome, nitrogen and sulphur metabolism and ribosomal proteins) were shown to be reflected in clusters from the yeast cell cycle data [13]. These four functional categories form the four classes in the external criterion for this data set.
- **B3:** A subset of 205 genes from the yeast galactose data set [14]. The expression patterns reflect four functional categories in the Gene Ontology (GO) listings.

As is popular practice in microarray data analysis, the data was row-normalized to have zero mean and unit variance. Under this normalization scheme, the Euclidean distance and cosine similarity are equivalent. From the experiment results shown in Table 1, averaged over 100 runs, CLUTO appears to perform best, closely followed by GlobalRSC and then K-means.

Table 1. Experimental results (the highest quality index values are in bold)

Data	Algorithm	ARI	AMI	EPrec	ERec	ECos	Loops	Time (s)
B1	K-means	0.42±0.05	0.48±0.02	0.55±0.02	0.56±0.04	0.55±0.03	11±4	0±0
	CLUTO	**0.50**±0.00	**0.52**±0.00	**0.59**±0.00	0.61±0.00	**0.60**±0.00	N/A	0±0
	GlobalRSC	0.48±0.01	0.50±0.01	0.56±0.01	**0.62**±0.02	0.58±0.01	11±3	2±1
B2	K-means	0.49±0.03	0.33±0.01	0.65±0.01	0.54±0.02	0.59±0.01	8±3	0±0
	CLUTO	**0.51**±0.00	**0.34**±0.00	0.65±0.00	**0.56**±0.00	**0.60**±0.00	N/A	0±0
	GlobalRSC	0.50±0.02	0.29±0.01	**0.69**±0.01	0.48±0.01	0.56±0.01	11±3	1±1
B3	K-means	0.83±0.09	0.78±0.08	0.92±0.02	0.83±0.09	0.86±0.06	7±4	0±0
	CLUTO	**0.96**±0.00	**0.91**±0.00	**0.95**±0.00	0.96±0.00	**0.96**±0.00	N/A	0±0
	GlobalRSC	0.92±0.04	0.84±0.07	0.92±0.05	**0.96**±0.01	0.93±0.03	3±0	0±1
I1	K-means	0.48±0.01	0.72±0.00	0.66±0.01	0.56±0.01	0.56±0.01	20±4	1278±347
	K-means*	0.49±0.01	0.74±0.00	0.61±0.01	0.59±0.01	0.55±0.01	23±5	993±416
	CLUTO	0.53±0.00	0.75±0.00	**0.67**±0.00	0.61±0.00	0.59±0.00	N/A	1015±51
	CLUTO*	0.53±0.00	0.75±0.00	0.62±0.00	0.62±0.00	0.58±0.00	N/A	847±9
	GreedyRSC	0.59±0.00	0.77±0.00	0.64±0.00	0.68±0.00	0.62±0.00	N/A	328±24
	GlobalRSC	**0.61**±0.01	**0.78**±0.00	0.66±0.00	**0.71**±0.00	**0.64**±0.00	17±3	417±29
I2	CLUTO	0.41±0.00	0.73±0.00	0.48±0.00	**0.52**±0.00	0.48±0.00	N/A	7893±16
	CLUTO*	0.48±0.00	0.69±0.00	0.80±0.00	0.35±0.00	0.50±0.00	N/A	27302±192
	GreedyRSC	0.56±0.00	0.72±0.00	0.82±0.00	0.42±0.00	0.56±0.00	N/A	4352±53
	GlobalRSC	**0.59**±0.00	**0.74**±0.00	**0.85**±0.00	0.46±0.00	**0.59**±0.00	28±3	5002±127
T2	CLUTO*	0.02±0.00	0.24±0.00	0.60±0.00	0.02±0.00	0.10±0.00	N/A	1201±14
	GreedyRSC	0.04±0.00	0.26±0.00	0.61±0.00	0.04±0.00	0.12±0.00	N/A	495±2
	GlobalRSC	**0.09**±0.00	**0.29**±0.00	**0.64**±0.01	**0.07**±0.00	**0.16**±0.00	17±4	1616±167
T1	GreedyRSC	0.01±0.00	0.25±0.00	0.71±0.00	0.01±0.00	0.06±0.00	N/A	10657±334
	GlobalRSC	**0.02**±0.00	**0.26**±0.00	**0.72**±0.00	**0.02**±0.00	**0.08**±0.00	18±5	19044±756

*: K-means and CLUTO run using the number of clusters K as determined by GreedyRSC

4.2 Image Data

We tested the clustering algorithms on the Amsterdam Library of Object Images (ALOI) [15], which consists of 110,250 images of 1000 common objects. Each image is represented by a dense 641-dimensional feature vector based on color and texture histograms (see [16] for details on how the vectors were produced). The following data sets were used:

- **I1-ALOI-var:** A subset of 13943 images, generated by selecting objects unevenly from among the classes, with the i-th object class having approximately $40000/(400+i)$ image instances selected.
- **I2-ALOI-full:** The entire ALOI library.

Since the appearance of individual objects varies considerably with the vantage points of the images, almost every class would be expected to generate several natural clusters. Over 20 runs, GreedyRSC estimated the number of clusters to be 843 ± 8 for the I1 data set, and 3724 ± 25 for the I2 data set. Fast K-means and CLUTO were executed twice with random initialization, for both the true numbers of clusters ($K = 1000$) and the number of clusters as determined by GreedyRSC. GlobalRSC was initialized using GreedyRSC, and a SASH was used to construct the neighborhood sets. All runs except those involving CLUTO were conducted using the Euclidean distance measure. Over 20 runs, GreedyRSC consistently achieves good clustering quality which is then further refined by GlobalRSC, as observed in Table 1. For the ALOI-full data set, the execution of Fast K-means failed to terminate due to insufficient main memory.

It can be observed that when a good initialization is provided and SASH is used, the execution time of GlobalRSC is significantly shorter, comparable to that of K-means and CLUTO.

4.3 Text Data

We tested the clustering algorithms on the Reuters Corpus Volume I (RCV1), an archive of over 800,000 manually categorized newswire stories recently made available by Reuters, Ltd. for research purposes [17]. The document class structure was simplified into 57 distinct classes. We then selected a subset T1 consisting of 200,000 documents classified to either exactly one subtopic or exactly one meta topic. We also constructed a smaller data set T2 consisting of 20,000 documents selected uniformly at random from T1. For both T1 and T2, TF-IDF weighting was used to construct the feature vectors, resulting in sparse vectors of length 320,648. Fast K-means was excluded in this experiment due to its lack of support for sparse numerical data. Since the number of external classes of these data sets was not as reliable as of the ALOI image data set, we first ran GreedyRSC to estimate the number of natural clusters K. The clustering result of GreedyRSC was used for the initialization of GlobalRSC, while CLUTO was run with the desired number of clusters also set to K. The cosine similarity measure was used for all runs. The clustering scores, averaged over 20 runs, are reported in Table 1. While all the algorithms successfully processed the small T2 set, on T1 CLUTO gave a memory failure message after a few hours of execution, leaving only the results of GreedyRSC and GlobalRSC available for evaluation. The observed low agreement between the clustering result and the class information in this experiment can be attributed to natural fragmentation of the classes within the data domain.

4.4 Categorical Data

The mushroom data set, drawn from the UCI machine learning repository [18], contains 8124 varieties of mushrooms, each recorded with 22 different categorical physical attributes (such as color, odor, size, and shape). Each record is classified as to whether its associated mushroom is poisonous or edible. The distance measure for this data set is taken as the straightforward mismatch count, with missing values treated as contributing to the count.

The mushroom data set was previously analyzed with ROCK [2], which in their paper was reported as finding 21 clusters. Most of the clusters consist of only one type of mushroom, either edible or poisonous. Only 32 mushrooms were misclassified by ROCK. On 20 runs with random initialization, GreedyRSC produced 22 ± 1 clusters, with 87 ± 120 mushroom instances misclassified. The result is further refined with GlobalRSC, which brought the number of mushroom species misclassified down to 46 ± 97. The classification errors of the 20 runs are reported in table 2. All the algorithms greatly outperformed the traditional hierarchical clustering implemented in [2], which produced 20 clusters within which 3432 out of 8124 items were misclassified. K-means was excluded from

Table 2. Experimental results on the Mushroom data set: classification errors over 20 runs

Classification errors																				
GreedyRSC	32	69	69	32	289	484	69	33	44	33	32	32	278	32	32	52	32	32	32	32
GlobalRSC	1	33	33	1	257	289	33	1	1	1	1	1	257	1	1	0	1	1	1	1

this experiment as it can not handle categorical data. A clustering algorithm from CLUTO, which operates on the similarity matrix, was tested but did not yield competitive results, with 1339 misclassified species. It should be noted that whereas ROCK required an estimate of the number of clusters, GreedyRSC automatically determined this number.

5 Conclusion

In this paper we have introduced a novel shared-neighbor clustering algorithm based on the Relevant Set Correlation (RSC) model. The key difference in our approach to clustering, compared to other shared-neighbor-based approaches, is that it requires the setting of only one main parameter — the number of clusters. The objective function greatly resembles that of K-means, and like K-means, the GlobalRSC method aims to discover compact, globular clusters. While this class of clusters appears to be restrictive, Dasgupta [19] has shown that for high dimensional data, random projection can transform highly eccentric clusters into more spherical ones, which in turn can be discovered by K-means or GlobalRSC. The techniques we presented for improving the scalability of our proposed GlobalRSC algorithm allow for practical application of the method for large, high-dimensional generic data sets under any reasonable measure of similarity.

Acknowledgement

This work was partially supported by NICTA. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Jarvis, R.A., Patrick, E.A.: Clustering using a similarity measure based on shared near neighbors. IEEE Trans. Comput. 22(11), 1025–1034 (1973)
2. Guha, S., Rastogi, R., Shim, K.: Rock: A robust clustering algorithm for categorical attributes. Information Systems, 512–521 (1999)
3. Ertöz, L., Steinbach, M., Kumar, V.: Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In: Proc. 3rd SIAM Intern. Conf. on Data Mining, SDM (2003)

4. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. 2nd Int. Conf. on Knowl. Discovery and Data Mining (KDD), pp. 226–231. AAAI Press, Menlo Park (1996)
5. Houle, M.E.: The relevant-set correlation model for data clustering. Stat. Anal. Data Min. 1(3), 157–176 (2008)
6. Dasgupta, S.: The hardness of k-means clustering. Technical Report CS2007-0890, University of California, San Diego (2008)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press, McGraw-Hill Book Company (2000)
8. Houle, M.E., Sakuma, J.: Fast approximate similarity search in extremely high-dimensional data sets. In: ICDE 2005: Proceedings of the 21st International Conference on Data Engineering, Washington, DC, USA, pp. 619–630. IEEE Computer Society, Los Alamitos (2005)
9. Elkan, C.: Using the triangle inequality to accelerate k-means. In: Proceedings of the Twentieth International Conference on Machine Learning, ICML 2003 (2003)
10. Karypis, G.: CLUTO – a clustering toolkit (2002)
11. Lawrence, H., Phipps, A.: Comparing partitions. Journal of Classification 2(1), 193–218 (1985)
12. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In: ICML 2009: Proceedings of the 26th international conference on Machine learning (2009)
13. Yeung, K.Y.: Cluster analysis of gene expression data. PhD thesis, University of Washington, Seattle, WA (2001)
14. Yeung, K., Medvedovic, M., Bumgarner, R.: Clustering gene-expression data with repeated measurements. Genome Biology 4(5), R34 (2003)
15. Geusebroek, J.M., Burghouts, G.J., Smeulders, A.W.M.: The amsterdam library of object images. Int. J. Comput. Vision 61(1), 103–112 (2005)
16. Boujemaa, N., Fauqueur, J., Ferecatu, M., Fleuret, F., Gouet, V., LeSaux, B., Sahbi, H.: Ikona: Interactive specific and generic image retrieval. In: International workshop on Multimedia ContentBased Indexing and Retrieval, MMCBIR 2001 (2001)
17. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: RCV1: A new benchmark collection for text categorization research. J. Mach. Learn. Res. 5, 361–397 (2004)
18. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
19. Dasgupta, S.: Experiments with random projection. In: UAI 2000: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, pp. 143–151. Morgan Kaufmann Publishers Inc., San Francisco (2000)