Clustering Using Subset Groupings

Kalyan Veeramachaneni and Una-May O'Reilly

No Institute Given

Abstract. We introduce a new problem we call *Resolving Groupings of Subsets* which emerges from scenarios in which multiple computational agents (algorithms or people) group distinct or overlapping, relatively small, subsets of a very large dataset. The goal is to fuse this partial information into a globally coherent grouping of the entire dataset. We contribute fusion strategies and a means of determining whether there is sufficient information from the subsets' groupings to yield a stable consensus grouping. For the opportunity when subsets can be actively composed while the solution framework is executing, i.e. in an online setting with active learning, we devise multiple strategies for adaptive subset composition. The strategies rely on computationally inexpensive, non-linear transformations of local evidence, such as accumulated pairwise element co-occurrence in a group. The local evidence facilitates efficient subset composition which contributes to attaining better accuracy more quickly. Frequent adaptation rather than delayed adaptation provides the most accurate empirical results.

Keywords: clustering, unsupervised learning, crowd sourcing

1 Introduction

We are interested in developing algorithmic frameworks for situations when numerous, very small, subsets of a large dataset are each independently grouped (a.k.a clustered, sorted, associated) and these groupings must subsequently be fused to derive a consensus representing a grouping of the entire dataset. This context is motivated by distributed data storage and cloud computing as well as scenarios when human experts (instead of algorithms) form the groupings within subsets of data. We provide detailed motivational examples in Section 3.

This problem, which we call *Resolving Groupings of Subsets* and abbreviate as *RGS*, is a new problem instance within clustering. Until recently, most clustering problems have assumed the availability of the entire data at a central location and the algorithm's goal is to cluster the entire dataset. Most recently, ensemble approaches which employ diverse clustering methods and include a consensus building component have been devised to address the unsupervised nature of the clustering problem, which precludes a notion of ground truth or one true answer [1, 2]. From a broad perspective, *RGS*, introduces a new goal within clustering. Rather than identify clusters automatically within the dataset, its goal is to resolve, via consensus, multiple groupings that have already been

identified, though on numerous, small subsets of the dataset, rather than the entire dataset. *RGS* assumes that the entire dataset is covered collectively by the subsets. It allows that independent subsets, of equal composition, might yield different groupings, leading to a subproblem of resolving inconsistencies. It allows any composition of the subset with respect to elements of the entire dataset. That is, subsets may overlap with each other, or be distinct. It assumes that the subset size, m, is greatly smaller than the entire data set of size n, i.e. $m << n$. Intuitively, this assumption presents the problem of resolving missing information. No information is available regarding how the dataset elements that were *not* included in the subset would be grouped. If each subset was, in fact, the entire dataset, the problem would reduce to be solely one of ensemble fusion. If the subset size was much closer to the dataset size, the problem would reduce to that of resolving missing labels.

This paper presents a novel framework for addressing *RGS*. Its contributions address each of the challenges arising in developing *RGS* algorithms:

Challenge 1: How should multiple grouping outcomes, which are incomplete with respect to the entire dataset and which collectively completely cover the entire dataset, be accurately fused to identify a grouping of the entire dataset? This challenge is pronounced because the subset size is assumed to be greatly smaller than the entire data set of size. The assumption makes the challenge significantly different from missing label approaches such as [3], though a modification is possible. Our contribution, described in Section 4 is a multidimensional scaling (MDS) based approach which uniquely identifies outliers in the subset groupings before fusing them.

Challenge 2: Can it be efficiently determined whether there is sufficient information from the groupings of the subsets to yield a reliable consensus grouping? Inherent in each *RGS* subset is missing information because the groupings of elements of the dataset which are not in the subset are unknown. Intuitively, the fusion algorithm "knits" together only partial information. How quickly the fusion algorithm can confidently group each element of the dataset will depend upon how much subsets overlap in composition, how efficiently they completely cover the entire dataset and the consistency of element groupings across subsets. We describe a systematic means of determining the information content and information gain between fusing successive groupings into the consensus. We define a stability index whose convergence indicates that additional grouping information will not significantly contribute to a change in the consensus.

Challenge 3: If subsets can be actively selected from the entire dataset, how should they be composed?
We assume this opportunity presents itself in two different versions. In the first version there is some means available *completely ahead of grouping and fusion,* to compose every subset. With no other knowledge of groupings, the best strategy is to randomize subset selection.The second version of active subset selection is both incremental and dynamic in nature. The challenge is, in view of current grouping information and the application of a *RGS* algorithm, to dynamically select the contents of one or more subsets to follow, incorporate their group-

ings into up-to-date consensus groupings, then either iterate again or conclude with a stable fusion. Our contribution, described in Section 5, introduces adaptive strategies that use information about currently amassed groupings to select subsets of data elements to present for grouping. The most simple approach iteratively composes subsets of random elements. We present approaches that are superior to random element subset composition and focus upon computationally inexpensive approaches.

Another contribution of this paper is a public release of a *RGS* problem and algorithm testing framework. The framework has two means of defining a problem, its simulation data and ground truth: full scale emulation of agents and dataset features, statistical sample of ground truth provided by a probabilistic co-occurence matrix. This will encourage consistent definition and facilitate the design and empirical evaluation of new fusion and adaptive subset composition strategies.

Infact this area is currently of great interest among the research community. [4] present an active strategy to seek distance measurements between points in a database in order to cluster them. Similarly [5] seeks to fill a similarity matrix for spectral clustering in minimum number of pairwise queries. Both these approaches are substantially different from ours as they are not seeking inputs from multiple agents about subsets. In addition [4] assumes availability of features pertaining to the data. Perhaps, an example closest to our work is [6]. They present a similar problem of clustering via crowd sourcing in which multiple people cluster subsets of data (images). The focus is laid on fusing multiple subset groupings via a Bayesian approach. But they do not provide an active means of subset selection which is the focus of this paper.

We proceed as follows: Section 2 formalizes the problem description. Section 3 provides motivating examples of *RGS* with additional details. Section 4 presents the fusion algorithm with MDS. Section 5 presents multiple active subset composition strategies. Section 6 presents the experimental setup and results achieved on test problems. Finally section **??** summarizes the paper.

2 Problem Formalization

We are given a dataset $\mathcal{D}(n)$ and multiple stochastic grouping agents $\mathcal{A}$. We provide a subset of data $\mathbb{Q}_h = \{i | i \in \mathcal{D}(n), |\mathbb{Q}_h| = m\}$ as an input to an agent $\mathcal{A}_h$. The agent groups this data into an arbitrary number of groups r denoted as $G_h = \{g_{1h}, \ldots g_{rh}\}$ where each $g_{ih} = \{j | j \in g_{ih}, j \in \mathbb{Q}_h\}$ represents a group of data elements. For evaluation purposes we may assume there exists a true grouping of the entire data denoted as Π_G. We define two problems:

***Static Resolving Groupings of Subsets*:** Multiple subsets $\mathbb{Q}_{1\ldots h}$ such that the union of these subsets equals the entire data set are given. Each agent groups one of these subsets and provides $G_{1\ldots h}$. The goal is to fuse the multiple groupings to form a consensus grouping C that is optimal according to some qualitative metric of consensus. In a testing and validation framework C can be compared to Π_G.

***Dynamic Resolving Groupings of Subsets*:** This problem is an iterative process toward fusing multiple groupings to form a consensus grouping. In an iteration, t, groupings are sought for $\mathbb{Q}^t_{1\ldots h}$ via agents $\mathcal{A}_{1\ldots h}$. These, and any groupings of previous iterations, are fused into consensus C. At the end of an iteration, whether to halt and return C as the consensus grouping or to continue with another iteration is decided upon. If continuing, new subsets $\mathbb{Q}^{t+1}_{1\ldots h}$ based on the information $G^t_{1\ldots h}$ can be formulated via an active subset composition algorithm.

2.1 Formal Agent Models

Our goal is to derive a grouping as close as possible to the true grouping Π_G of the data $\mathcal{D}(n)$ whether the problem is static or dynamic. The true grouping is assumed to be non-retrievable because subset groupings have errors with respect to it. The source of error in subset groupings is the stochastic nature of the *grouping* agent. Agent stochasticity injects errors between ground truth and agent responses that are assumed to arise because an agent uses only its subset of data as bases for comparison, agents are free to use different criteria to group subsets, group separation may be uncertain, or a number of other factors.
We present two possible models of a stochastic *grouping* agent:

2.2 Co-occurrence matrix based agent model

In this model, the ground truth clustering for the dataset $\mathcal{D}(n)$ is captured in a $n \times n$ co-occurrence matrix Π where $\Pi_{ij} = 1$ if data elements i and j belong to the same cluster. A probabilistic version of this co-occurrence matrix, $P(\Pi)$ is used to represent the stochasticity and deviation from the ground truth. A $Bernoulli(\mathbf{B}(n,p))$ distribution is used to model the probabilistic co-occurrence matrix given by $P(\Pi_{ij}) = \mathbf{B}(1, p_{ij}))$. A *grouping* agent first generates a co-occurrence matrix by sampling entries in Π via $Pi_{ij} \sim \mathbf{B}(1, p_{ij})$. Referencing this, it can group any subset of data $\mathbb{Q}$.

2.3 Data driven, statistical, parameterized agent model

In this model, a dataset $\mathcal{D}(n,k)$ is generated where k denotes the feature dimensionality of the each data element. The true number of clusters $\mathbf{s}$ and the number of data elements in each cluster n_i where $i = \{1 \ldots \mathbf{s}\}$ and $n = \sum_{i=1}^{\mathbf{s}} n_i$ is provided as input to the data generation method. The method uses a parameterized multivariate distribution $P(x_1, \ldots x_k|\Theta_i)$ to generate data elements for each cluster. The parameters of the multivariate model, Θ_i, can be varied to generate clustering tasks of varied difficulty. A parameterized Gaussian dataset using $\mathcal{N}_i(\bar{\mu}, \mathbf{\Sigma})$ is generated in [7] to evaluate the Bayesian k-means algorithm.

The agent is modeled via a machine learning clustering algorithm. An agent is initialized with the parameters to execute the clustering algorithm and (the subset $\mathbb{Q}$ (which provides features for each data element). The agent executes

the clustering algorithm which references this dataset and the algorithm outputs the groupings.

A number of choices are available for machine learning clustering algorithms, e.g. *kmeans*, *fuzzy-cmeans*. Parameters for each clustering algorithm are $r, \gamma, \bar{\alpha}$ where r is the number of clusters into which the agent will group the data, γ is the number of features the agent will use and $\bar{\alpha}$ represents the probabilities of selecting of the γ features. To generate multiple heterogeneous *grouping* agents, a distribution over these three parameters and the clustering algorithm choices is defined and an agent is formulated by sampling it. For example, an agent $A_1(\mathcal{C} = kmeans, \beta = 3, \gamma = 4, \bar{\alpha})$ clusters the subset into 3 clusters by only using 4 out of the k features selected based on the probabilities in $\bar{\alpha}$ using *kmeans* algorithm. More fine grained parameterization in which one could select the distance functions and other parameters within clustering algorithm is obviously possible but not a focus of this paper.

Distributions over the parameterizations allows control of the properties of the group of agents and hence the diversity in their groupings.

3 Motivation

We are motivated by a number of real world scenarios:
Situation Assessment Systems with Human Agents: In this scenario the agents which group subset data are human analysts. The entire dataset is huge, of diverse media and unstructured. It might include news material, internet blog feeds and actor profiles. It is much too large and complex for any one agent to completely analyze independently. Further, any one agent does not have sufficiently broad expertise to assess the dataset from multiple vantage points. Each analyst processes a subset of the complex, unstructured dataset and is required to group the subset's specific elements. Any basis for grouping is admissible. It may be directed from some authority or it may be left open. The problem is to develop a system which adaptively unites the groupings from the domain experts. The system must knit together, i.e. synthesize, integrate or merge, human discerned structure as it is elicited when analysts interpret heterogenous unstructured data sources.

In this scenario, *RGS* exploits the latent knowledge of an analyst and computational clustering by agent algorithms is undesirable. The lattter would require features and features extraction would reduce the knowledge being considered. The key motivation for using analysts is to take advantage of their integrative assessment capabilities.They are not expected to provide any explicit annotation, descriptions or features of the subset and their groupings.
Clustering a big dataset: In this scenario we need to group a big dataset located at a central location and our compute server is limited by its local memory size, compute power and bandwidth. It therefore cannot cluster the entire dataset. To address this the server queries the for subsets multiple times and each time an agent groups a subset. Then the server fuses the subset groupings.

Distributed datasets: Our final example is a distributed data storage scenario. Due to multiple restrictions: *bandwidth*, data *size* and *privacy*, it is not feasible to aggregate the data at a central location and then cluster. Consider multiple data storage locations $\mathcal{D}_{1...d}$ that each have an overlapping subset of the dataset $\mathcal{D}(n, k)$. Each data storage location has an associated compute unit that can execute an agent to cluster all its data or a subset of its data locally and communicate the results. One server is designated to fuse the subset grouping.

4 Fusing Subset Groupings

In *RGS*, each *grouping* agent $\mathcal{A}_h$ groups a subset of the larger data and these groupings must be fused. A variety of solutions are available in the literature for building consensus when groupings are available for an entire dataset. However, only [3], which is within the consensus clustering literature, explicitly attempts to build consensus from subset groupings. It treats the problem as one of missing data and solves it by estimating a mixture model via **EM** in the presence of missing data. The authors show that the loss in accuracy is significant as the amount of missing data increases from 10% to 50% [3]. Some of the accuracy can be regained by increasing the number of *grouping* agents, h. This approach is not applicable for our scenario because of the following reasons[1]:

Sparsity: *RGS* subset groupings yield very sparse information, in that, each o *grouping* agent only clusters $\sim 5\%$ of the data. Hence the amount of missing information in our case is always$> 50\%$.

Inconsistency across groupings: *RGS* assumes that agents behave stochastically when grouping. It also assumes that each agent could be attending to different aspects of the subset elements and can group the data into an unrestricted number of clusters. This intended inconsistency and flexibility make the problem of finding consensus in the subset groupings extremely hard.

Our choice for fusion is a multidimensional scaling based approach [8], see Algorithm 1. In this approach we first convert each partial solution G_h into a co-occurrence matrix $\mathbb{C}^h$ of size $n \times n$. $\mathbb{C}^h_{ij} = 1$ if the pair $\{i, j\}$ has been put together by $\mathcal{A}_h$ and the diagonal entries are initialized as $\mathbb{C}^{(h)}_{ii} = 1$. By doing this we force the data elements for which we do not have grouping information (from this *grouping* agent) each into their own group.

Each co-occurrence matrix is then converted to a distance matrix $(1-\mathbb{C})$ and squared. Two more transformations are further applied to this matrix resulting in a so called "cross product matrix", see Algorithm 1 for details.

Next we combine the multiple cross product matrices. This could be achieved by simple averaging. However, a weighted average will make consensus more accurate. We identify weights by ranking the cross product matrices based on

[1] In fact we initially adopted this approach but found that the MDS approach presented later performs better.

their correlation with each other and then assigning weights proportional to their rank. Specific details are in Algorithm 1. This method assigns higher weight to the subset grouping which has the most agreement with others. The grouping which disagrees the most is considered as an outlier and has the lowest weight. Note that assigning data elements, for which grouping information is missing, to their own groups does not introduce disagreements between subset groupings which include them and subset groupings which do not. This is because in both cases the diagonals for $\mathbb{C}$ are 1.

We next apply *PCA* to the combined cross product matrix (which represents the agreement between multiple partial groupings) to identify its principle components. These components are a new set of "features" for the data elements, (represented by F in Algorithm 1), in a new space. We execute a *kmeans* clustering algorithm with *silhouette* distance based automatic cluster identification on this new feature representation of the original dataset to find the groupings of the entire dataset $\mathcal{D}(n)$.

5 Active *Resolving Groupings of Subsets*

Having described a method to fuse the groupings of subsets that works for both static and active *RGS*, we now turn to the distinctive aspects of active *RGS*. Our algorithm is outlined as follows:

In the first iteration subsets are composed randomly. The groupings from the agents are fused and then examined to decide whether to continue or to halt and return the current fused consensus grouping. If continuing, then, based on the estimated consensus, a sampling distribution $P(S|\{\mathbb{C}_1 \dots \mathbb{C}_h\})$ is formed and the next set of subsets are composed by sampling data elements from this distribution.

In Subsection 5.1 we present the loop halting criteria. In Subsection 5.2 we describe multiple strategies for forming the distribution $P(S|\{\mathbb{C}_1 \dots \mathbb{C}_h\})$.

5.1 Halting Iteration

To determine whether a consensus is stable and no further subsets need to be grouped, we compare the consensus achieved at iteration t, Π_C^t to the consensus achieved in iteration $t-1$, Π_C^{t-1}. We evaluate the mutual information between the two groupings. Mutual information provides a comprehensive measure between two sets of groupings [1]. Iteration stops if the mutual information between some number of successive iterations is no greater than ϵ. This indicates that no more consensus information is being gained via additional subset groupings. This criteria could also be used to infer that the current active subset composition strategy has converged to a solution and might be exchanged for another.

5.2 Active subset composition

All our methods depend on sampling a subset $\mathbb{Q}_h = \{i | i \in \mathcal{D}(n), i = 1 \dots m\}$ by iteratively sampling from one of the two distributions:

I A probability distribution $P(S)$ where $S = \{1 \dots n\}$.
II A bivariate probability distribution $P(s_i, s_j)$ where $s_i, s_j = \{1 \dots n\}$, and $i \neq j$

We provide multiple strategies to set up this probability distribution as more evidence is gathered. We gather two forms of evidence from multiple $G_{1 \dots h)}$:

1. a cumulative pairwise co-occurrence matrix, $\mathbb{C} = \sum_{i=1}^{h} \mathbb{C}_i$
2. a cumulative pairwise sampling matrix, $\mathbb{P} = \sum_{i=1}^{h} \mathbb{P}_i$

Different strategies follow:

Baseline: Randomized selection In this method a subset consisting of m data elements is generated by randomly drawing without replacement from $\mathcal{D}$. A uniform random distribution $P(S) = U[1n]$is used. Subsets for multiple agents are generated by repeating this procedure h times.

Explore: Balanced pairwise presentation This method consults the accumulative counts of the pairwise presentation matrix which records how many times a pair of data elements has appeared together in a subsets. The pairwise presentation matrix is first normalized with its maximum value. Each value is then transformed into a probability of sampling for pair ij via

$$P(s_i, s_j) = \frac{\exp(-\mathbb{P}_{ij})}{\sum_{ij} \exp(-\mathbb{P}_{ij})} \tag{1}$$

This method attempts to balance the presentation of data element pairs such that they are all equally presented. Balanced pairwise presentation is an ill-posed problem because each pair presented so far is not independent of the previous pairs presented due the possibility of repetition. To satisfy the constraint that we present at least the required number of data elements, for one subset we sequentially draw from the entire set while removing the repeated elements until a total number of m unique data elements are selected.

Exploit-pairwise: Disambiguation of confused pairings This method relies on the pairwise co-occurrence matrix, $\mathbb{C}$ which is updated and accumulated each iteration. The intuition behind a naive co-occurrence based method is that we decide not to present pairs of data elements that have been paired together consistently. There are two limitations in this approach. One is that only pairwise co-occurrence is considered. The second is that this only considers the data points that have been consistently paired together, but not the ones that are consistently not paired together. To overcome the first drawback, we divide each co-occurrence matrix entry, representing a pair, by the number of times the pair has been presented together in a query. That is, we first element-wise divide the pairwise co-occurrence matrix by the $\mathbb{P}$.

$$E_{ij} = \frac{\mathbb{C}_{ij}}{\mathbb{P}_{ij}} \tag{2}$$

This arithmetic produces values between $[0, 1]$. A value of 1 implies that the products were put together 100% of the times they were presented together. A value of 0 implies that they were not put together even though they were presented together in one or multiple queries. A value of 0.5 indicates that the pair was put together only 50% of the times they were presented together. The closer this value to 0.5, the more confused the data is with respect to deciding whether to consider the pair similar. They have been grouped in an ambiguous way. To quantify this sense of ambiguity, we transform the E value into an *Ambiguity Index* using a Gaussian function with a mean value at 0.5 and setting a standard deviation of σ given by

$$AI_{ij} = \mathbf{G}(E_{ij}; \mu, \sigma). \tag{3}$$

The Gaussian function is given by

$$\mathbf{G}(e, \mu, \sigma) = \exp\left(-1 \times \left(\frac{e-\mu}{\sigma}\right)^2\right) \tag{4}$$

The Gaussian function's standard deviation, σ, controls how quickly the *Ambiguity Index* drops off with respect to E. We then attempt to sample the pairs from the data based on this index. We transform this index into probabilities by

$$P_{ij} = \frac{AI_{ij}}{\sum_{ij} AI_{ij}} \tag{5}$$

Based on the above equation, pairs with a higher *Ambiguity Index* will have higher probability of being drawn from the dataset. It is likely that a higher value of σ initially allows more exploration and, as iterations are executed, shrinking σ will focus the subset selection towards more highly ambiguous pairs.

A Hybrid Strategy: Explore-Exploit-Tradeoff Neither *Exploit-Pairwise* and *Exploit-One-vs-Rest* consider weighting the *Ambiguity Index* to take into account the contrasting number of times sample pairs have been presented. Consider, for example, the two pairs. Pair $\{1, 2\}$ has an E value of 0.5 which corresponds to and *Ambiguity Index* of 1 and the pair $\{2, 5\}$ has an E value of 0.57 which corresponds to an *Ambiguity Index* of 0.956. However, the first pair has been sampled 20 times where as the second has been sampled 14 times. This contrast in pairwise presentation indicates that the pair $\{1, 2\}$ is more frequently confused because its evidence of 0.5 is based on more presentations.

To consult this information the hybrid strategy *Explore-Exploit-Tradeoff* combines the *Explore* and *Exploit-Pairwise* selection strategies via decision logic. Pairwise presentation matrix based selection provides the ability to achieve uniform presentation of pairs. Co-occurrence based selection merely relies on evidence generated from the $\mathbb{C}$. Combining these two strategies enables us to explore new combinations of samples as well as exploit the evidence we have already accumulated. *Explore-Exploit-Tradeoff* first picks a sample i from the data set

randomly. Given i we collect its *Ambiguity Index* and pairwise presentation with every other product as:

$$\begin{cases} AI^i = AI[i] \\ \mathbb{P}^i = \mathbb{P}[i]. \end{cases} \tag{6}$$

We then divide the remaining samples into three non overlapping sets determined with the help of two parameterized thresholds, β_1 and β_2. β_1 is a presentation count, β_2 is an *Ambiguity Index*:

$$\begin{cases} s_1 = \{i | \mathbb{P}^i < \beta_1 \& AI^i > \beta_2\} \\ s_2 = \{i | \mathbb{P} < \beta_1\} \\ s_3 = \{i | AI^i < \beta_2\}. \end{cases} \tag{7}$$

We first attempt to make choice from the first list s_1 probabilistically with probabliity of the i element proportional to wieghted sum of the ambiguity index and the presentation count. These are the samples that are at least as ambiguous as β_2 with respect to i and have at least been presented β_1 number of times with i. If s_1 is empty we flip a coin to choose between s_2 and s_3 and then choose a product probabilistically from one of the two. This encourages the algorithm to switch between exploration (*Explore*) and exploitation (*Exploit-Pairwise*). This decision logic has a few parameters, i.e., β_1, β_2, μ and σ. β_1 is an integer and forces the algorithm to present every pair of samples at least β_1 number of times. Once the next sample j is selected the process starting from equation 7 is repeated by replacing i with j and removing i and j from the available set of points.

6 Experimental Results and Discussion

We used the last model presented in section 2.3 to emulate an agent and the grouping task. This model is most comprehensive. We generated the data via multivariate Gaussian and set the dimensions $k = 8$ and specified the number of clusters as 10. We first generated the centers for each cluster (which are the μ's for the means of the multiple variables in the Gaussian distribution). We then generated the standard deviations for each dimension such that no two clusters are closer than $\tau \times \frac{\sigma_i + \sigma_j}{2}$. Thus τ gives us a, parameterized way to vary the difficulty level of grouping task. We varied the τ value between [0.1, 0.2, 0.5, 1, 2, 5, 10, 20], 0.1 being the hardest and 20 being the easiest.

We ran experiments for two sizes of data 10^2 (case 1)and 10^3 (case 2). For the Case 1 when the data size was 10^2 we had 8 data points per subset (8%) and limited the number of subset presentations to 1000. For Case 2, where we had 10^3 we had 50 (5% of the entire data) data points per subset and limited the number of subsets to 100. Due to larger data size, smaller fraction of data elements per subset, and small number of subsets, Case 2 is harder to solve than Case 1. For each of these cases we tested by varying the update frequency (number of subset groupings performed before the active subset composition algorithm is used).

Note that the update frequency is the same as the number of *grouping* agents, h, since we are querying these agents in parallel. Based on the update frequency the number of adaptive updates (only in the adaptive algorithms) is equal to $\frac{1000}{h}$ in the first case and $\frac{100}{h}$ in the second case. Finally, due to the stochastic nature of our strategies we did 10 independent runs and present average results. In all our experiments we compare the consensus achieved with the ground truth via mutual information (higher the better).

Dynamic, static or complete In our first test we compared three different algorithms. First algorithm generates subsets dynamically, the second algorithm generates subsets randomly. Finally as a benchmark we designed an algorithm, called *Complete*, in which we created multiple presentations each consisting of the entire dataset. Multiple full presentations enable us to overcome the stochasticity in the *grouping* agent. We used Case 1 to evaluate these algorithms. We fixed the number of presentations to 500 and present the average result from 10 trials. Figure 1 shows the results achieved for grouping tasks of different difficulty level (as τ increases the difficulty of the grouping task reduces). We see that the static (random) subset selection performs poorly even for an easy task. For a hard task ($\tau = 1$, we see that in fact adaptive subset selection performs better then the complete presentation. As the difficulty level of the task reduces we see that the adaptive algorithm increases in performance and attempts to approach the performance of the *complete*

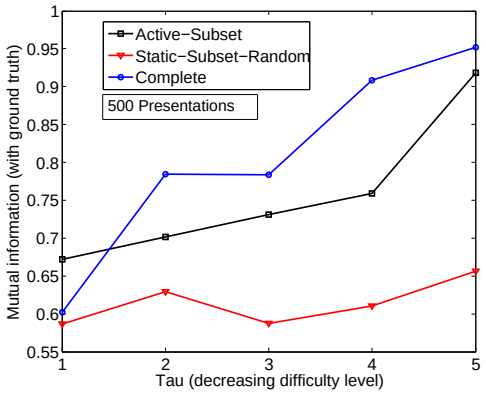


Fig. 1. Comparison of three algorithms, *Exploit*, *Explore-Exploit*, and *Random*. The difficulty of the problem increases as τ increases.

Comparison of different algorithms: Next we compare different adaptive strategies we designed in this paper among each other and the random subset selection strategy.

Case 1: 10^2 data elements, 8 elements per subset, 1000 subset presentations, h=1 Figure 2 shows results for grouping tasks of different difficulty level. We present the results for *Exploit-Pairwise*, *Random*, and *Explore-Exploit-Tradeoff* strategy. We see that the *Exploit-Pairwise* strategy performs better than *Random* in all cases and the *Explore-Exploit-Tradeoff* strategy performs slightly better than random. It should also be noted that the active subset selection strategy quickly converges to a good solution in fewer presentations. For tasks with higher difficulty level the *Exploit-Pairwise* strategy achieves the solution faster and possibly should be stopped after a few presentations.

Case 2: 10^3 data elements, 50 elements per subset, 100 subset presentations, h=1 Figure 3 presents the results for Case 2. We add an additional active algorithm for comparison called *Explore*. We notice that *Exploit-Pairwise* performs better than the rest. This case has more data elements. It is evident from these figures that the algorithm has not converged and we can achieve better results if we continue to do more presentations for *Exploit-Pairwise*.

Measuring stability In our results above we observe that for Case 2 the algorithm seems to be still improving in accuracy when compared to ground truth. Although we assume availability of ground truth for validation, in reality that is not the case. In such a scenario we use the stability measurement to decide whether to seek more subsets or not. Figure 4 tracks the stability measurement, defined as mutual information between two subsequent grouping consensus, for the two experiments. The improvment in performance of the consensus grouping is also shown. We see that for the first case the stability quickly rises to a value of 0.9 and stabilizes there. This implies that beyond this point the subset groupings are not providing any more information. For case 2 however the measurement is much more erratic and the overall value is much lower than the Case 1. We can notice that the performance for the case 2 is increasing.

Affect of number of *grouping* agent or update rate We now investigate whether it is better to dynamically select subsets more frequently. Up until now we have presented where we dynamically selected subsets after every iteration. That is we updated our probabilistic model for selection presented in Section 5 every time we acquired a subset grouping result from the agent.

In Figure 5 we show the performance of different algorithms for Case 2 (100 queries, 10^3 data points). First we note that irrespective of our update rate the *Exploit* strategy performs well across problems of different difficulty. The margins are higher when the difficulty level is higher, which is what we expect. *Explore* only strategy performs the worst and never improves. Another significant observation we make is that for higher update rate (lower h) we are able to retrieve results that are closer to ground truth. For example, compare the results at problem difficulty level $\tau = 1$. With update rate of $h = 1$ we are able to retrieve a solution with a mutual information of 0.45 with the *Exploit* algorithm. In contrast, with an update rate of $h = 8$, we are able to retrieve only somewhere close to 0.1 even after 100 queries. Additionally at such a low update rate all the algorithms perform equally well including the random one. These margins however are less pronounced (still exist) for problems with lower difficulty levels.

7 Conclusions

In this paper, we addressed the problem of *RGS* by actively seeking groupings from subsets. We developed a few active subset composition strategies and analyzed them under different conditions. We found that it is beneficial to dynamically select subsets after receiving each grouping information from a *grouping* agent. Our algorithms and analysis can be applied in a wide variety of scenarios including situations where humans are grouping data. In fact we provide a framework to emulate grouping of an arbitrary dataset via humans.

References

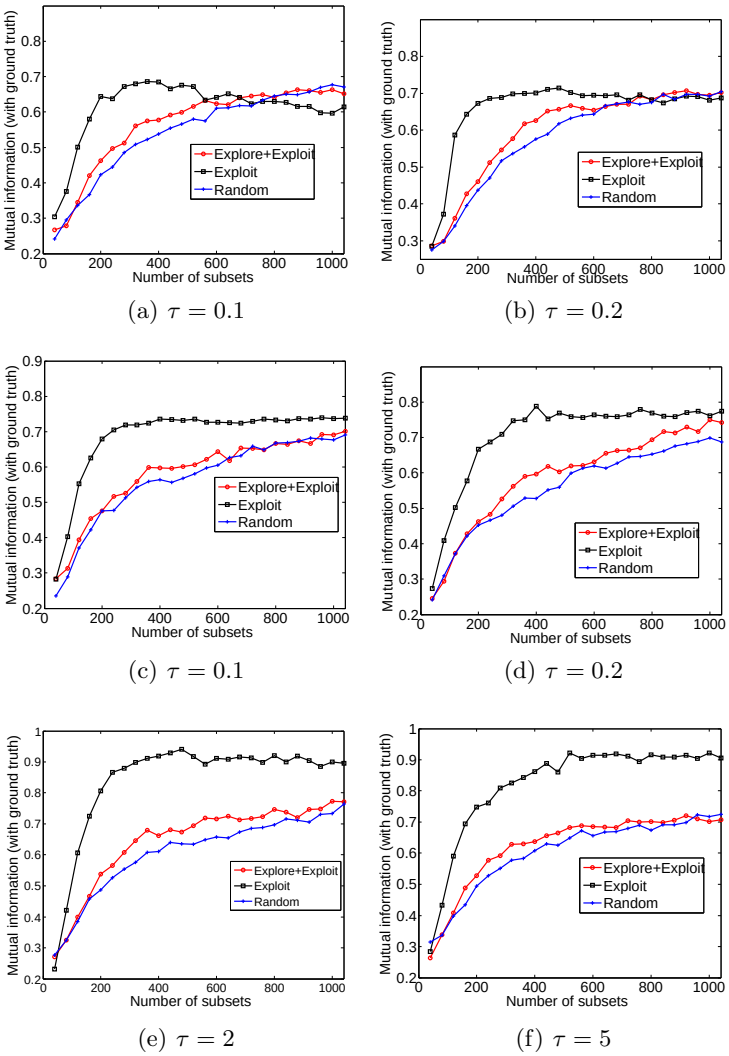
1. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. The Journal of Machine Learning Research **3** (2003) 583–617
2. Topchy, A., Jain, A., Punch, W.: Clustering ensembles: Models of consensus and weak partitions. IEEE Transactions on Pattern Analysis and Machine Intelligence (2005) 1866–1881
3. Topchy, A., Jain, A., Punch, W.: A mixture model for clustering ensembles. In: Proceedings of SIAM Conference on Data Mining. (2004) 379–390
4. Voevodski, K., Balcan, M.F., Roglin, H., Teng, S.H.: Active clustering of biological sequences. Journal of Machine Learning Research **13**
5. Shamir, O., Tishby, N.: Spectral clustering on budget. Proceedings of Fourteenth International Conference on Artificial Intelligence and Statistics(AISTATS) (2011)
6. Ryan, G., Welinder, P., Krause, A., Perona, P.: Crowdclustering. Neural Information Processing Systems (2011)
7. Welling, M., Kurihara, K.: Bayesian k-means as a maximization-expectation algorithm. In: Sixth SIAM International Conference on Data Mining. Volume 22. (2006)
8. Abdi, H., Valentin, D., Chollet, S., Chrea, C.: Analyzing assessors and products in sorting tasks: Distatis, theory and applications. Food quality and preference **18**(4) (2007) 627–640

Algorithm 1 Fusing subset groupings

function COMBINE-MDS($\{\mathbb{C}^{(1)} \dots \mathbb{C}^h\}$)
  for each $\mathbb{C}^h$ **do**
    $\mathbf{D}^h = 1 - \mathbb{C}^h$
    $\mathbf{S}_h = -\frac{1}{2}\Xi\mathbf{D}_h\Xi^{\mathbf{T}}$
    where $\Xi = \mathbf{I} - 1^{\mathbf{T}}$, $\mathbf{I}$ is an identity matrix, and $\mathbf{m} = \frac{1}{n}$
    Divide $\mathbf{S}_h$ by its first eigen value
  end for
  Evaluate $\mathbf{R}_{ij} = \frac{trace\{\mathbf{S}^{\mathbf{T}}_{(i)}\mathbf{S}_{(j)}\}}{\sqrt{trace\{\mathbf{S}^{\mathbf{T}}_{(i)}\mathbf{S}_{(i)}\} \times trace\{\mathbf{S}^{\mathbf{T}}_{(j)}\mathbf{S}_{(j)}\}}}\ \forall i, j$
  $E \leftarrow$ EIGEN($\mathbf{R}$)
  $\mathbf{w} = \frac{\mathbf{e_1}}{\sum \mathbf{e_1}}$
  $\mathbf{S} = \sum_{i=1}^{h} w_i S_i$
  $[\ V_1, V_2] \leftarrow$ EIGEN($\mathbf{S}$)
  $F = V_1 \times$DIAG$(\sqrt{(V_2)}$
  return F
end function

Algorithm 2 Active *Resolving Groupings of Subsets*

Choose number of agents h, number of data points per subset q
Form subsets $\mathbb{Q}_{1\dots h}$ by selecting a random subset of data elements
Loop:
for each $\mathcal{A}_h$ **do**
  Present the subsets $\mathbb{Q}_{1\dots h}$ to agents $\mathcal{A}_{1\dots h}$
end for
Combine outputs of agents via Algorithm 1 to achieve Π_C
Decide whether to continue or halt.
If halting, return Π_C
Form a **sampling distribution** $P(S|\{\mathbb{C}_1 \dots \mathbb{C}_h\})$, $S = \{1, n\}$ where $S = i$ implies selection of data element i to form a query $\mathbb{Q}$
for each $\mathcal{A}_h$ **do**
  Build a subset $\mathbb{Q}_h = \{i | i \in \mathcal{D}(n), |\mathbb{Q}_h| = m\} \sim P(S)$
end for
Goto 3

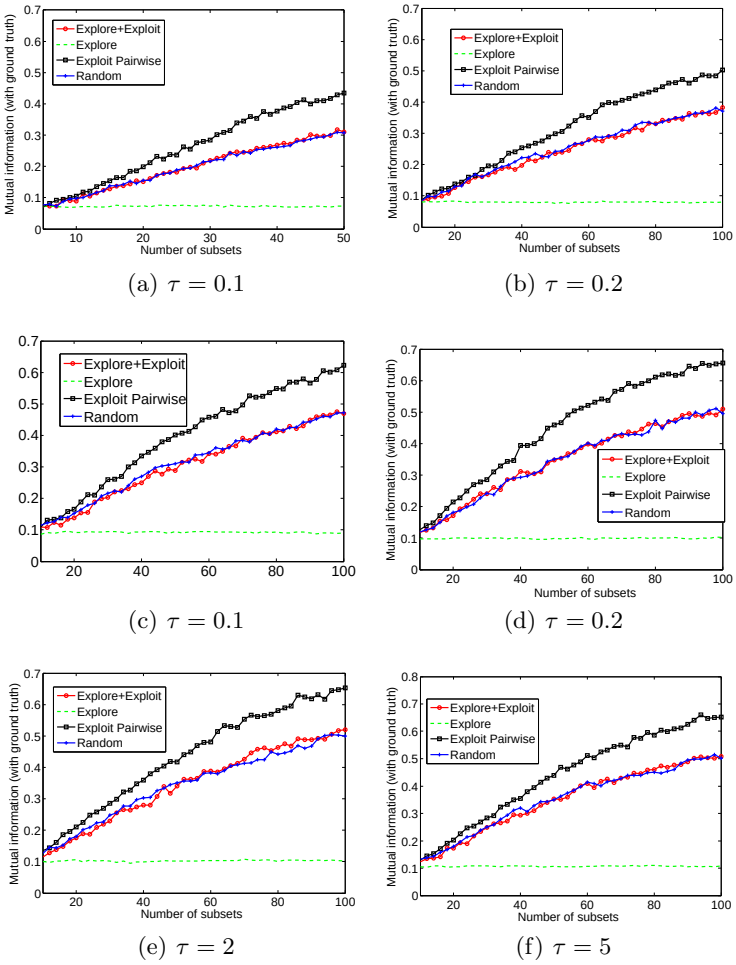


(a) $\tau = 0.1$ (b) $\tau = 0.2$

(c) $\tau = 0.1$ (d) $\tau = 0.2$

(e) $\tau = 2$ (f) $\tau = 5$

Fig. 2. Comparison of three algorithms, *Exploit*, *Explore-Exploit*, and *Random*. The difficulty of the problem decreases as τ increases.

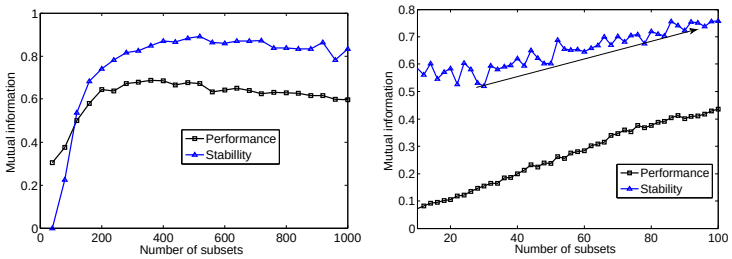


(a) $\tau = 0.1$ (b) $\tau = 0.2$

(c) $\tau = 0.1$ (d) $\tau = 0.2$

(e) $\tau = 2$ (f) $\tau = 5$

Fig. 3. Comparison of four algorithms, *Exploit-Pairwise*, *Explore-Exploit-Tradeoff*, *Explore* and *Random*. The difficulty of the problem decreases as τ increases.



(a) Case 1, h=1, *Exploit-Pairwise* (b) Case 2, h=1, *Exploit-Pairwise*

Fig. 4. Plot of stability measurement as we proceed with seeking groupings for subsets.

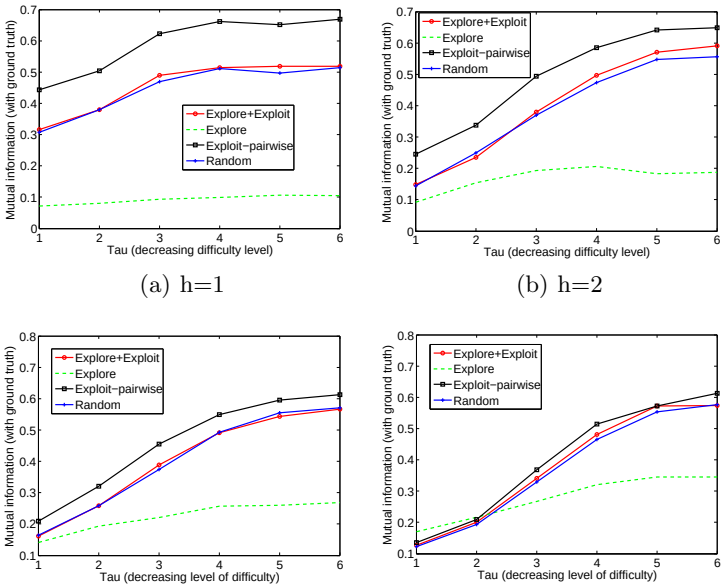


Fig. 5. Performance of different algorithms as we decrease the difficulty level of the grouping task (from left to right) by varying the parameter τ. Different algorithms performance is shown on a single plot. The update frequency is changed from 1-2-4-8 as we go from left to right top to bottom. The results are at the end of the 100 queries in total.