CLUMP: A Scalable and Robust Framework for Structure Discovery

Kunal Punera Joydeep Ghosh

{kunal,ghosh} @ ece.utexas.edu

IDEAL-2006-02 *

Intelligent Data Exploration & Analysis Laboratory

(Web: http://www.ideal.ece.utexas.edu/)

Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas 78712
U.S.A.

May 2005

Abstract

We introduce a robust and efficient framework called CLUMP (CLustering Using Multiple Prototypes) for unsupervised discovery of structure in data. CLUMP relies on finding multiple prototypes that summarize the data. Clustering the prototypes enables our algorithm to scale up to extremely large and high-dimensional domains such as text data. Other desirable properties include robustness to noise and parameter choices. In this paper, we describe the approach in detail, characterize its performance on a variety of datasets, and compare it to some existing model selection approaches.

1 Introduction

Clustering has been an active area of research for many years and numerous techniques have been proposed for it [JD88]. Many of the approaches, like the popular KMeans algorithm, rely on the user to pre-specify the number of clusters desired. The model selection problem of determining the appropriate number of clusters automatically is itself a very active area of research, and once again numerous approaches have been proposed [HBV01, JD88, LRBB04, TWH01].

Agglomerative clustering [JD88] provides a natural way to discover the number of clusters present in the data. The Dendrogram generated by the algorithm encodes a series of nested clustering solutions with different number of clusters (k). Various approaches exist for deducing the correct value of k from a Dendrogram [SC04, TWH01]. A key advantage of agglomerative methods like single-link is their ability to discover arbitrary non-spherical structures in the data. On the other hand, validation methods for KMeans clustering solutions are constrained by KMeans' objective function which is best suited for finding compact spherical structures in the data. There are, however, several problems with agglomerative approaches. They tend to be greedy and this makes them extremely susceptible to outliers as well as noise in the data. Also, agglomerative clustering methods find the pairwise similarity between all points before clustering them hierarchically. Pairwise similarity computation is quadratic in the number of data points, making these approaches impractical for large datasets. Our framework deals with both these problems.

In order to overcome these problems, we summarize the objects by a few prototypes that are concentrated around the dense regions of data and are much more sparse around the non-dense regions. We show that agglomerating over these prototypes leads to more noise-resilient and computationally efficient clustering.

1.1 Contributions

The following are the key contributions of our work.

- We propose a novel framework to discover the true number of arbitrarily shaped clusters in the data.
- We evaluate our approach on a variety of datasets and show that it statistically significantly outperforms Evidence Accumulation [FJ02] and Gap Statistic [TWH01] in terms of quality of clusterings found.
- We show that our approach is more robust to outliers and noise than algorithms that agglomerate over the data.
- We demonstrate the scalability of our algorithm by performing experiments on several large, high-dimensional datasets, including text data.

1.2 Organization of paper

In the next section we will introduce CLUMP, our approach for discovering the true number of clusters in the data. Related work in the area of cluster validation is then reviewed in Section 3. Section 4 provides a small intuitive demonstration of how CLUMP finds the true structure in the data in spite of the presence of significant amounts of noise. In Section 5, we evaluate our algorithm on a variety of datasets and compare its performance against Evidence Accumulation and Gap Statistic.

2 CLUMP: CLustering Using Multiple Prototypes

In this section we will describe the CLUMP framework for structure discovery in data and provide an algorithm for it. Later in the section we explain the rationale behind it.

2.1 Notation

Let $X = \{x_1, ..., x_n\}$ denote the dataset of n d-dimensional patterns. We apply r clustering algorithms $\{A^{(q)}\}_{q=1}^{r}$ to the data, each grouping the objects into $\{k^{(q)}\}_{q=1}^{r}$ clusters. Each clustering algorithm $A^{(q)}$ outputs a $n \times k^{(q)}$ matrix $I^{(q)}$ such that $I^{(q)}_{(i,j)} = 1$ if x_i is placed in cluster j by $A^{(q)}$, other wise $I^{(q)}_{(i,j)} = 0$. Centroid-based algorithms, like KMeans, describe each cluster by the centroid of the data in the cluster. In addition to $I^{(q)}$, these algorithms output a $k^{(q)} \times d$ matrix $C^{(q)}$ which contains the $k^{(q)}$ cluster centers. There are a total of $m = \sum_{q=1}^{r} k^{(q)}$ clusters.

The algorithm proceeds in the following three steps.

2.2 Obtain Prototypes

Our first step is to summarize the data by a set of m prototypes. These prototypes are obtained by running a clustering algorithm repeatedly to obtain r clustering solutions. Each x_i is now represented by r prototypes, one from each run. We obtain the overall membership matrix $I = (I^{(1)}...I^{(r)})$ by concatenating all the individual membership matrices. For centroid-based clustering algorithms all individual centroid matrices are concatenated to yield a $m \times d$ matrix $C = (C^{(1)T}...C^{(r)T})^T$. For non-centroid based algorithms, we can simply have $C = I^T$. Here each cluster is described by a binary vector indicating whether each data-point belonged to the cluster or not.

In practice, we set all the $\{k^{(q)}\}_{q=1}^{r}$ to the same value, between two to three times a rough estimate of the actual number of clusters in the data. The reason for this choice is explained later in the paper. The clustering algorithm we use for exposition purposes and for our experiments is KMeans, but any clustering algorithm that finds clusters with the desired properties can be used. We describe these properties in detail later in this section.

2.3 Cluster the Prototypes

The set of m prototypes is clustered using the single-link agglomerative method to yield a tree structure of nested clusterings called a Dendrogram [JD88]. If the underlying clustering algorithm is centroid-based, then its measure of distance is used for computing the pair-wise distances between prototypes. For instance, when the clustering algorithm used is KMeans, distance between prototypes is the Euclidean distance. For non-centroid based algorithms, pair-wise distances between prototypes can be defined in terms of the number of data points they share: the Tanimoto coefficient (Intersection/Union) between rows of C. We plot a graph with the x-axis denoting the number of clusters and the y-axis denoting the value of the similarity function when merging at x clusters. The *knee/elbow* of the curve, or the point of maximum curvature, is used to return the appropriate number of clusters. Salvador and Chan [SC04] discuss many methods for finding the number of clusters from such graphs. We note that finding the number of clusters from the unnormalized curve of the similarity function performed better than using a normalization procedure like Gap Statistic [TWH01]. Moreover, Gap Statistic is very computationally expensive, since it requires multiple clustering runs for each value of number of clusters.

Say, we obtain K such meta-clusters, *i.e.* clusters of prototypes. For each of the K meta-clusters we obtain a membership vector of size n which describes each data-point's association with that meta-cluster. This association can be calculated as the sum of the point's association to all the prototypes that are clustered into that meta-cluster. Once, we know which points associate with which meta-clusters, we select a minimal set of meta-clusters that *covers* all the points. A data point is said to be covered if it has a non-zero association to one of the selected meta-clusters. In practice a greedy process of selecting meta-clusters in the order of decreasing number of *uncovered* points they cover is seen to work very well. As explained later in this section, this process helps eliminate many clusters that formed solely of noisy data-points and outliers.

2.4 Obtain the Final Clustering

At the end of the previous step we obtain a set of k meta-clusters such that all points have a non-zero association with at least one of them. In the final step, points are assigned to the meta-cluster to which they are most associated. Ties in associations are broken randomly. This gives us a final set of k clusters and the points assigned to them.

2.5 Rationale behind the approach

Our approach tries to mitigate the ill-effects of the greediness and computational complexity of the agglomerative methods by converting the problem of clustering data-points into the problem of describing the data with a few prototypes and then clustering these prototypes. The prototypes are obtained using algorithms that have a time complexity linear in size of data. Since CLUMP agglomerates over the prototypes that are far fewer in number than the data-points, its computational complexity is much lower than algorithms that agglomerate over the whole dataset. Moreover, if we can ensure that the prototypes are concentrated around the dense regions and spread out in the sparser regions of the data, the boundaries of the true clusters will be well demarcated. This would ensure that greedy agglomerative clustering methods do not merge true clusters early in the clustering process.

The prototypes are obtained by running an algorithm like KMeans multiple times over the dataset. Each KMeans run is seen as trying to fit a mixture of spherical Gaussians model to the data. Since we don't know exactly how many clusters are present in the data, we over-cluster (use a large k) the data so that KMeans finds small compact clusters. As we are over-clustering, it is highly probable that each dense region in the data will contain at least one KMeans centroid. Moreover, since each KMeans run is only guaranteed to find a local minima of its global objective function, we obtain prototypes from multiple runs of KMeans with different initialization. Hence, prototypes are likely to be concentrated in the dense regions of data and spread out in the non-dense regions. Therefore, KMeans centroids serve our requirements of prototypes of the data very well.

Prototypes are grouped into meta-clusters using the single-link agglomerative algorithm which helps our approach find arbitrarily complex shapes in the data. The *knee* (Figure 1(d)) in the plot of the single-link similarity function *w.r.t.* the number of meta-clusters gives us an estimate of the number of groups in the data.

The set of meta-clusters obtained is now pruned to remove the redundant ones as described above. The single-link algorithm groups prototypes concentrated in the dense regions of data into meta-clusters early in the clustering process, leaving prototypes in sparse regions of data as singleton or small meta-clusters. We eliminate these smaller meta-clusters if the objects they cover are already covered by larger meta-clusters. This often helps us eliminate meta-clusters that are formed of noisy points and outliers. The pruning process would be adversely affected if prototypes from some KMeans run were larger clusters than prototypes from others. The meta-clusters containing these larger prototypes would have a much better chance of having other meta-clusters removed from the final set of clusters. In order to avoid this situation we use the same $k^{(q)}$ for all the KMeans runs.

CLUMP has two user-defined parameters. r is the number of KMeans runs to combine. We show later in our experimental evaluation that $r = 10, ..., 15$ performs well for many real-life datasets. The second parameter is the number of clusters for each KMeans run. In most real-life domains, we have a vague idea about how many clusters, k, exist in the data. It will be shown that clusterings obtained for $k^{(q)} = 2 \times k, ..., 3 \times k$ are good and stable.

3 Related Work

Agglomerative clustering approaches provide a very natural way to discover the clusters in the dataset [JD88]. They are, however, susceptible to outliers, and very computationally expensive. Here we briefly mention some notable efforts on alleviating these problems. Readers are referred to the original papers for more details.

BIRCH [ZRL96] was proposed as a method to cluster a large database in a single pass. It tries to reduce the input data size by an incremental and approximate pre-clustering phase that represents data by small spherical clusters whose size depends on main memory constraints. In a single pass of the data, each data point is assigned to a centroid if the closest centroid is within a threshold distance; else it forms its own cluster. BIRCH was designed primarily for computational efficiency and has been shown to be sensitive to outliers in the data.

CURE [GRS98] discovers arbitrarily shaped clusters by representing each cluster by a set of *well scattered* points, that are moved toward the cluster center by a fraction α. This makes the algorithm less vulnerable to outliers. However, its time complexity is $O(n^2 \log n)$. The authors try to make CURE scalable by first sub-sampling the data. In contrast, CLUMP can afford to use the whole dataset, since its time complexity is linear in the number of data-points.

Fred and Jain [FJ02] introduced the Evidence Accumulation (EA) framework based on the idea of combining the results of multiple clusterings. Like our approach, EA also over-clusters the dataset multiple times to obtain a series of clusterings. These clusterings are however used to transform the original data into a pairwise similarity space, where two objects are considered similar if they occur in the same cluster in many clusterings. A single-link method is then used to cluster the points. The single-link algorithm is stopped when a similarity threshold of $t = 0.5$ is reached. Since agglomerative clustering is performed over the set of data-points, the time complexity of this method is at least quadratic in the size of the dataset making it impractical for large datasets. In contrast, our approach clusters the prototypes, which are far fewer in number than the data-points. Also, as shown in Section 4 and Subsection 5.4, CLUMP is more far more resilient to noise and outliers in the data than EA.

Gap Statistic [TWH01] is a popular technique for cluster model selection. It works by estimating, for each k, the expected value of a statistic on randomly distributed data. The smallest k for which the gap between the value of the statistic on the data being clustered and its expected value stops increasing is output as the true number of clusters. Often a statistic's expected value is estimated using Monte Carlo sampling. This involves running the clustering algorithm multiple times for each different values of k under consideration making Gap Statistic very expensive computationally.

Outside of the cluster validation literature there exist some techniques that bear slight resemblance to our approach. In the Cluster Ensemble framework, Strehl and Ghosh [SG02] introduced the Meta-CLustering Algorithm (MCLA) to combine multiple clustering solutions. MCLA achieves this by matching the clusters from different solutions by grouping them using graph partitioning methods. MCLA, however, requires that the desired number of clusters be provided.

Bradley and Fayyad [BF98] describe a method to use multiple clustering runs to find a good initialization for KMeans. They initially cluster multiple sub-samples of the data using KMeans with a specified value of k. All the centroids obtained are then reclustered using KMeans to obtain k meta-clusters which serve to initialize a KMeans run of the full dataset. The problem we are tackling is completely different since we do not know the true k and try to discover it.

4 Illustration on a Toy Dataset

In this section we show, with the help of artificial data, how CLUMP implicitly performs outliers detection and finds clusters even in the presence noisy data when Evidence Accumulation (EA) [FJ02] fails to do so. The 2 dimensional dataset used for this experiment (Figure 1(a)) consists of 400 points sampled from two Gaussians of unit variance centered at $(1, 1)$ and $(7, 7)$ (indicated by blue dots and black pluses respectively). In order to make the problem harder, 30 noise points are placed uniformly between the two clusters (indicated by red triangles in Figure 1(a)). CLUMP was run with parameters $k^{(q)} = 6$ and $r = 15$ reducing the dataset to 90 prototypes shown in Figure 1(b). As we can see most prototypes are concentrated in the dense regions of the data, with some clearly corresponding to noise data points. The key thing to note here is the absence of too many prototypes from the sparser areas of the data. This absence of prototypes clearly demarcates the boundaries of clusters.

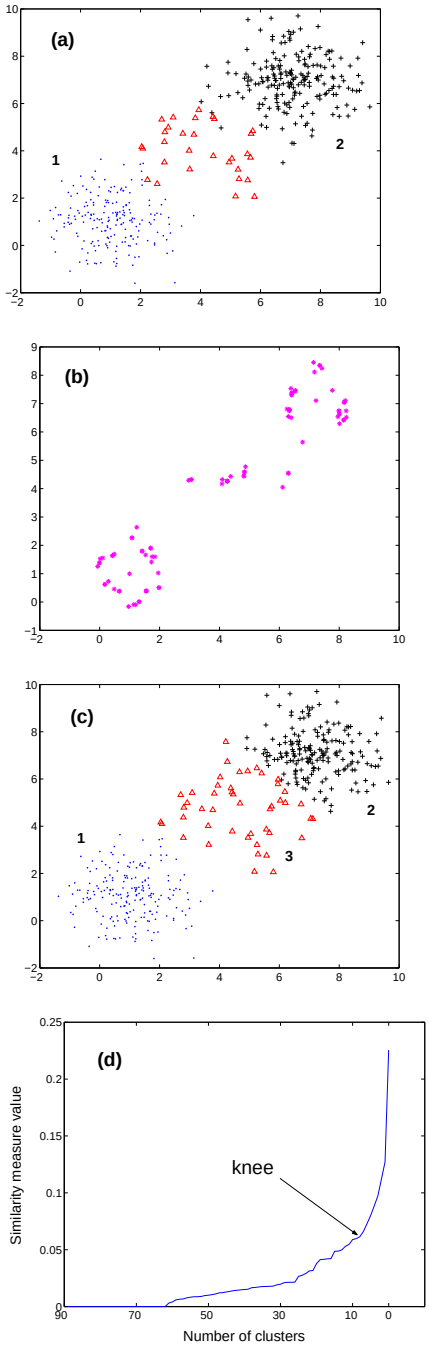

Figure 1: (a) The data with 2 clusters and noise marked by red triangles. (b) Prototypes obtained by CLUMP. (c) Final clustering correctly identifies clusters 1 and 2 and separates noise into cluster 3. (d) The graph of similarity function and number of clusters. The knee of the curve is shown.

CLUMP discovers three clusters in the data marked by different shapes in Figure 1(c). We can see that CLUMP succeeds in identifying Clusters 1 and 2 and groups the noise into Cluster 3 yielding a clustering with a NMI score of 0.902. EA (run with $k^{(q)} = 20$ and $r = 200$) when applied to the dataset, merged all the data into 1 cluster. Further, since CLUMP only had to agglomerate over 90 prototypes, our algorithm ran much faster than EA. On the current problem EA on an average took about 40 seconds to run on a 3 GHz machine with 2GB of RAM. Our algorithm took less than 1 second.

5 Experiments on Real Datasets

Our approach seeks to discover natural clusters in the data. Many researchers use extrinsic measures such as class labels to evaluate cluster quality. These class labels, however, may or may not correspond to the *natural structure* in the data. For example, certain classes may be multi-modal and represented by two clusters in the data. Hence, while we will use the class labels to reflect the natural structure in the data, we will report results using multiple clustering quality measures. These additional measures would give more insight into the quality of clusterings found. These measure are introduced and explained later in this section.

We compare CLUMP to Evidence Accumulation (EA) [FJ02], and KMeans (with k guessed using the Gap Statistic [TWH01]) on a variety of datasets. We show that on a majority of datasets, CLUMP statistically significantly outperforms both EA and KMeans+Gap. Furthermore, CLUMP is shown to be much faster and more robust to noise than EA. We also show how CLUMP parameters can be set globally for all datasets.We end this section with experiments with text data to showcase the scalability of our approach.

5.1 Evaluation measures

To give a complete picture of the performance of our algorithm we use multiple evaluation criteria in tandem.

Number of Clusters k: We report the number of clusters found by the algorithms being evaluated. Showing the distribution of number of clusters found would have been more semantically meaningful, but to conserve space we only present the average number of clusters obtained by CLUMP, EA, and KMeans+Gap. In case of some datasets, finding as many clusters as classes indicates that the true structure in the data has been found.

Classification via Clustering (CVC): As explained earlier there might exist more clusters than classes in the data. In such cases a user would be satisfied with a clustering that finds multiple clusters for each class such that the purity of each cluster in terms of the class labels of its constituents is high. CVC is simply the classification accuracy assuming all members of a cluster were predicted to be members of the majority class in that cluster. Each cluster's contribution to CVC is weighed by its size, so that very small pure clusters don't compensate for large impure ones. CVC is biased in favor of solutions with large number of clusters, but for a fixed number of clusters, it can be useful for comparing clustering solutions.

Normalized Mutual Information (NMI): The NMI of a clustering *w.r.t* the class labels is the Mutual Information between the two labellings normalized by each of their entropies. $NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}}$ where, $I(X,Y)$ denotes the mutual information between two random variables X and Y and $H(X)$ denotes the entropy of X. This measure was introduced by Strehl and Ghosh [SG02] and is frequently used for evaluating clusterings. It has some nice properties such as $NMI(X,X) = 1$ and if Y has only one cluster label for all instances $NMI(X,Y) = 0$.

Another measure of clustering accuracy is Adjusted RAND [HA85]. The Adjusted RAND compares two labellings based on whether pairs of objects are placed in the same or different clusters in them. The maximum value it takes is 1, and its expected value is 0. We computed the Adjusted RAND score for each solution and found it to be highly correlated to the NMI score. Hence we will only report the NMI score in the paper.

Name	#classes	#features	#instances
Iris	3	4	150
Wine	3	13	178
Glass	6	9	214
Pendigits	10	16	550
8D5K	5	8	500
WBC	2	9	700
Waveform	3	41	600
Vowel	11	10	990
20News-diff	3	7666	3000
20News-sim	3	10083	3000

Table 1: Properties of datasets used in experiments

5.2 Datasets Used

To objectively evaluate our approach against EA and KMeans+Gap we used many datasets with widely different properties to reduce any bias of dataset selection. Some of these properties are listed in Table 1. WBC stands for the Wisconsin Breast Cancer dataset. Many of the datasets are from the UCI Machine Learning repository [BM98], while 8D5K was used in [SG02] and can be obtained from `www.strehl.com`. The Pendigits, Waveform, and 8D5K datasets (that originally contained 10000, 5000, and 1000 points respectively) were sub-sampled to obtain smaller datasets of size 500, 600, and 500 points respectively while maintaining the class priors of the original dataset. This was necessary as experiments with EA and Gap Statistic are not very computationally expensive on datasets with more than 1000 points. For text-data experiments we used different subsets of the 20Newsgroup dataset [HB99]. EA and KMeans+Gap was not evaluated on text-data, because of computational costs.

5.3 Comparison with EA and KMeans+Gap

Table 2 shows the results of running CLUMP, EA, and KMeans (with Gap Statistic) on 8 datasets. CLUMP was run with parameters: $k^{(q)}$ was randomly set between $2 \times k$ and $3 \times k$, $r = 15$. Parameter settings for EA were $r = 200$, and $k^{(q)} = \lfloor \sqrt{n} \rfloor$ as recommended in [FJ02]. For KMeans the number of clusters were estimated by Gap Statistic, whose B parameter (controls the number Monte Carlo sample datasets) was set to 20. For all algorithms, each number reported in Table 2 was obtained by averaging over 100 runs. Entries in **Bold** indicate datasets on which CLUMP performed statistically better (level < 0.01) than EA or vice versa. Similarly, ***** next to an entry indicates that CLUMP performed statistically better than KMeans+Gap or vice versa on that dataset.

The results in Table 2 show that CLUMP was overall the best of the three techniques compared. On all datasets, expect Pendigits, CLUMP performed statistically better than EA. On comparison with the KMeans and Gap Statistic combination, CLUMP performed statistically better on 6 out of 8 datasets. This might be because KMeans' underlying modeling assumption of spherical Gaussians may not hold for most real world datasets. On the other hand, since CLUMP agglomerates over prototypes of small dense regions, it can find arbitrarily shaped clusters. For all datasets other than Waveform, the average number of clusters found by CLUMP was closer than or as close to the *true* number of clusters as those found by Gap Statistic. However, on the Waveform dataset, CLUMP scored higher in terms of NMI and CVC than KMeans indicating that structure found by CLUMP matched the true structure of the data better.

Datasets		CLUMP	EA	KMeans+Gap
Iris	k	2.62/0.8	3.87/0.64	3.8/1.2
	NMI	**0.74/0.02 ***	0.69/0.01	0.71/0.03
	CVC	0.74/0.08	0.81/0.05	0.87/0.06
Wine	k	4.22/0.71	4.33/0.47	1/0.0
	NMI	**0.40/0.02 ***	0.38/0.00	0.0/0.0
	CVC	0.68/0.03	0.72/0.00	0.4/0.0
Glass	k	7.28/0.5	9.65/0.73	3.15/1.8
	NMI	**0.47/0.02 ***	0.46/0.00	0.35/0.04
	CVC	0.55/0.01	0.59/0.00	0.53/0.1
Pendigits	k	10.88/1.5	10.38/1.19	10.1/2.9
	NMI	0.7/0.01 *	0.71/ 0.03	0.66/0.05
	CVC	0.72/0.05	0.66/0.06	0.72/0.12
8D5K	k	5/0.14	4.02/0.14	4.3/1.1
	NMI	**1/0.01***	0.91/0.01	0.91/0.1
	CVC	1/0.01	0.8/0.02	0.87/0.15
Wisconsin Breast Cancer	k	3.11/0.47	8.76/1.82	2.8/1.1
	NMI	**0.63/0.03**	0.34/0.17	0.69/0.06 *
	CVC	0.96/0.0	0.83/0.12	0.96/0.01
Waveform	k	7.42/1.12	11.2/2.3	5.35/0.59
	NMI	**0.49/0.03 ***	0.07/0.06	0.47/0.01
	CVC	0.79/0.04	0.37/0.06	0.76/0.02
Vowel	k	12.23/3.5	2.95/0.82	6.9/1.3
	NMI	**0.42/0.07**	0.17/0.05	0.43/0.02
	CVC	0.38/0.07	0.14/0.02	0.35/0.04

Table 2: Comparison of CLUMP with EA and KMeans+Gap. Each entry is mean/std. dev. of 100 runs.

5.4 Clustering in the presence of noise

In order to evaluate CLUMP on noisy datasets, we added some noisy data points to existing datasets. The noisy points were created by picking two points from different classes uniformly at random and adding a new point at a random position on the line joining the two selected points. This process creates data points in the gap between classes making it harder for cluster discovery algorithms to discern the presence of clusters. We report results on two datasets, but similar trends were seen for most datasets.

Figure 2(a) shows how CLUMP and EA performed on the Pendigits dataset with different levels of noise. As we increased the number of noisy points in the data, the accuracy of EA dropped drastically, while CLUMP's accuracy was unaffected. Figure 2(b) shows similar results for the 8D5K dataset. These results confirm our intuition that because CLUMP clusters prototypes instead of the original data-points, it is more resilient to noise in the dataset.

5.5 Scalability of CLUMP

In Figure 3 we plot the amount of time taken by CLUMP and EA to cluster different datasets. For each dataset, the height of the solid bar is equal to the time taken by CLUMP divided by the sum of the time taken by CLUMP and EA. Both algorithms need about the same amount of time to cluster the Iris dataset, but as the datasets become larger, EA takes much longer than CLUMP. This is because the running time of CLUMP increases linearly with the number of data-points making it much more scalable that EA.

5.6 Independence of Parameters Values

In this subsection we show how the CLUMP parameters r and $k^{(q)}$ can be set. The parameter r controls the number of times KMeans is run to obtain the prototypes. Table 3 shows how the NMI of final clusterings varies

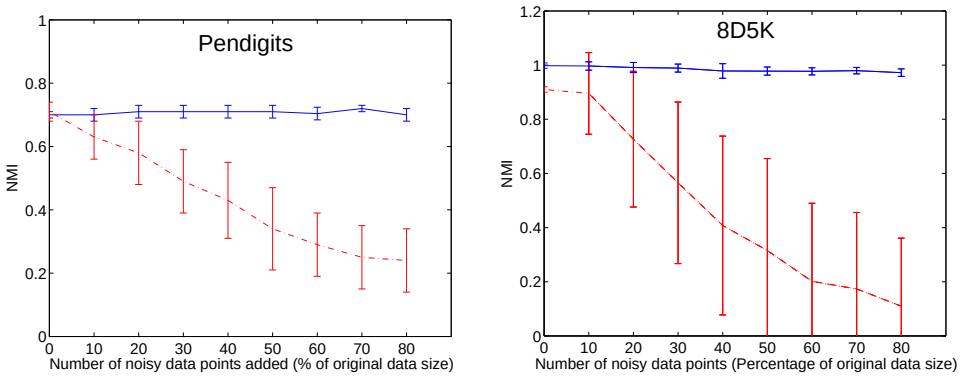

Figure 2: The x-axis shows increasing levels of noise in the dataset. Accuracy of EA (dotted line) drops dramatically with noise while CLUMP (solid line) remains unaffected.

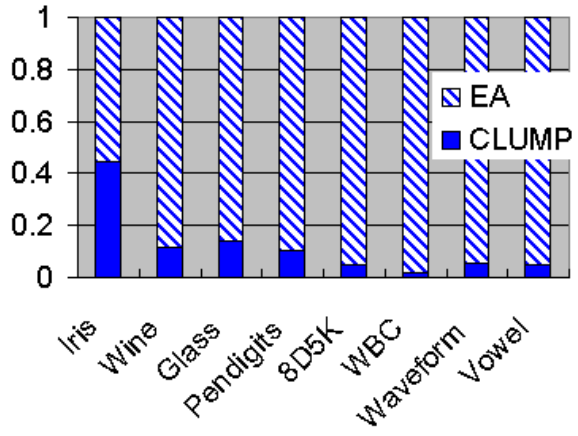

Figure 3: The solid bar indicates the time taken by CLUMP to cluster the dataset. The height of the striped bar is proportional to the amount of time taken by EA. Clearly, CLUMP is much faster than EA, and the gap becomes wider for larger datasets

while changing the r parameter. As is clear from the table, for most datasets the change in the value of NMI is negligible. This invariance enables us to set $r = 15$ for all datasets. But r can be reduced to 10 in case $k^{(q)}$ is large and we don't want to agglomerate over too many prototypes.

Parameter $k^{(q)}$ controls the number of prototypes generated from each clustering run. If the true number of clusters in the data was known, we would use a KMeans like approach. But in many domains, we have only a vague idea of the true number of clusters. Table 4 shows the change in NMI *w.r.t.* changing values of $k^{(q)}$. The numbers in the first row are the ratio of $k^{(q)}$ and the true k for each dataset. For most datasets, the value NMI obtained for $k^{(q)} = 2 \times k ... 3 \times k$ is fairly high and stable. In fact, for all datasets other than WBC the NMI value after some initial instability doesn't change much over the range of values of $k^{(q)}$.

Datasets	5	10	15	20	25
Iris	0.73	0.74	0.74	0.76	0.75
Wine	0.40	0.40	0.40	0.40	0.40
Glass	0.46	0.46	0.46	0.46	0.46
Pendigits	0.71	0.70	0.70	0.70	0.69
8D5K	0.998	0.998	0.998	0.998	0.998
WBC	0.63	0.63	0.64	0.64	0.64
Waveform	0.46	0.49	0.49	0.48	0.49
Vowel	0.39	0.43	0.42	0.43	0.43

Table 3: Change in NMI *w.r.t.* the parameter (r)

$\frac{k^{(q)}}{k} \rightarrow$	1	1.5	2	2.5	3	4	5
Iris	0.73	0.77	0.75	0.74	0.74	0.74	0.74
Wine	0.45	0.4	0.38	0.4	0.39	0.39	0.39
Glass	0.34	0.41	0.46	0.47	0.47	0.49	0.5
Pendigits	0.61	0.66	0.70	0.70	0.72	0.73	0.73
8D5K	0.78	0.86	0.97	0.986	0.998	0.998	0.998
WBC	0	0.71	0.67	0.63	0.59	0.61	0.64
Waveform	0.36	0.36	0.43	0.48	0.49	0.42	0.36
Vowel	0.42	0.43	0.42	0.42	0.39	0.33	0.26

Table 4: Change in NMI *w.r.t.* the ratio of $k^{(q)}$ and k

5.7 Experiments with Text

The 20-Newsgroup dataset [HB99] contains 1000 documents for each of the 20 different Newsgroups. Since some pairs of classes are extremely similar and have overlapping components, it is unclear how many actual clusters of documents are present in the data. In order to objectively evaluate our algorithm, we create two smaller datasets. 20News-diff contains 1000 documents each from classes alt.Atheism, rec.sport.Baseball, and sci.Space. As is evident, these classes are about extremely different topics, and we expect to find atleast 3 distinct clusters. 20News-sim contains three extremely similar classes talk.politics.guns, talk.politics.mideast, and talk.politics.misc. These classes have articles about the very similar topics and many documents are cross-posted across them. Some properties of these datasets are summarized in Table 1.

In order to effectively obtain cluster prototypes in such high-dimensional space, we used Spherical KMeans (Sp-KMeans) [DM01]. Sp-KMeans normalizes the data to lie on a unit hyper-sphere and then uses a distance measure derived from cosine similarity to cluster the data. We used the same distance measure to obtain pair-wise distance between the prototypes for agglomerative clustering. In these experiments, CLUMP is compared against Sp-KMeans with the true number of clusters $k = 3$ provided. For both datasets, CLUMP was run with parameters $k^{(q)} = 10$ and $r = 15$. Since both EA and Sp-KMeans+Gap run Sp-KMeans several times, it was infeasible to use either approach to cluster this data. Table 5 summarizes the results. The *Average* results are averaged over 100 runs. For Sp-KMeans the *Best* result of the 100 runs is also reported.

CLUMP finds the correct number of clusters for the 20News-diff dataset. What is most remarkable is that CLUMP was able to find the true number of clusters after reducing 3000 documents to 150 prototypes. 20News-sim is a very tough dataset as is evident from Sp-KMeans results. The clustering obtained by CLUMP scored higher than Sp-KMeans on average. CLUMP found on average 5 clusters in the data, often splitting up the class talk.politics.misc since it had overlapping components with the other two classes.

Datasets		Sp-KMeans		CLUMP
		Best	Average	Average
20News-diff	k	3		2.99/0.69
	NMI	0.84	0.83/0.08	0.76/0.1
	CVC	0.97	0.95/0.06	0.89/0.13
20News-sim	k	3		5.2/1.38
	NMI	0.37	0.34/0.09	0.37/0.06
	CVC	0.62	0.62/0.07	0.70/0.09

Table 5: Evaluation of CLUMP on subsets of 20 Newsgroup dataset. Results are averaged over 100 runs. Sp-KMeans was provided $k = 3$ and its best result over of 100 runs is also reported.

6 Conclusions and Future Work

In this paper we introduced CLUMP, a generic framework for discovering structure in the data based on summarizing the data with multiple prototypes. On comparison with Evidence Accumulation and Gap Statistic, CLUMP was shown to obtain much better clustering solutions. We showed that CLUMP discovers the true structure in the data even in the presence of extremely large amounts of noise. CLUMP was also shown to be tolerant to variations in the design parameters. The scalability of our approach was demonstrated by clustering large, high dimensional text datasets. As a direction for future work, we would like to compare CLUMP to sampling based approaches like BIRCH and CURE. Further, we aim to extend CLUMP to the problems of Outlier Detection and Overlapping Clustering.

References

[BF98] Paul S. Bradley and Usama M. Fayyad. Refining initial points for k-means clustering. In *ICML '98*, pages 91–99, 1998.

[BM98] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/mlrepository.html, 1998.

[DM01] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, Jan 2001.

[FJ02] A.L.N. Fred and A. K. Jain. Data clustering using evidence accumulation. In *Proc. of the 16th Int.l Conference on Pattern Recognition, ICPR 2002*, pages 276–280, 2002.

[GRS98] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. In *SIGMOD '98*, pages 73–84, 1998.

[HA85] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[HB99] S. Hettich and S. D. Bay. The UCI KDD,archive [http://kdd.ics.uci.edu]., 1999.

[HBV01] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *J. Intell. Inf. Syst.*, 17(2-3):107–145, 2001.

[JD88] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[LRBB04] Tilman Lange, Volker Roth, Mikio L. Braun, and Joachim M. Buhmann. Stability-based validation of clustering solutions. *Neural Computation*, 16(6):1299–1323, 2004.

[SC04] S. Salvador and P. Chan. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *Proceedings of the 16th IEEE International. Conference on Tools with AI*, pages 576–584, 2004.

[SG02] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research (JMLR)*, 3:583–617, December 2002.

[TWH01] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters via the gap statistic. *Journal of Royal Statistical Society*, B, 63(2):411–423, 2001.

[ZRL96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *SIGMOD '96*, pages 103–114, 1996.