CLUMP: A Scalable and Robust Framework for Structure Discovery

Kunal Punera
Electrical and Computer Engineering
University of Texas at Austin
kunal @ ece.utexas.edu

Joydeep Ghosh
Electrical and Computer Engineering
University of Texas at Austin
ghosh @ ece.utexas.edu

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

The model selection problem of determining the appropriate number of clusters automatically is an active area of research, and numerous approaches have been proposed [4, 6, 8, 10, 11]. Model selection via Agglomerative Clustering [6] is a natural solution to the problem [8, 10]. A key advantage of agglomerative methods like single-link is their ability to discover arbitrary non-spherical structures in the data. On the flip side, these methods are susceptible to outliers as well as noise in the data. Also, the need for pairwise similarity computation makes these methods impractical for large datasets.

Our framework deals with both these problems. First, 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 then show that agglomerating over these prototypes leads to a more noise-resilient and computationally efficient clustering.

Contributions: We propose a novel framework to discover the true number of arbitrarily shaped clusters in the data. We then evaluate our approach on a variety of datasets and show that it significantly outperforms Evidence Accumulation [4] and Gap Statistic [10] in terms of clustering quality and robustness to outliers. Finally, we demonstrate the scalability of our algorithm by performing experiments on several large, high-dimensional datasets, including text data.

2. CLUMP: CLustering Using Multiple Prototypes

The CLUMP algorithm proceeds in the following three steps.

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 grouping objects $\{x_i\}$ into $\{k^{(q)}\}_{q=1}^{r}$ clusters. Each clustering solution is a $n \times k^{(q)}$ matrix $I^{(q)}$ such that $I^{(q)}_{(i,j)} = 1$ if x_i is placed in cluster j, otherwise $I^{(q)}_{(i,j)} = 0$. Each x_i is now represented by r prototypes, one from each run. We obtain the overall membership matrix as $I = (I^{(1)}...I^{(r)})$. Prototypes obtained from centroid-based clustering algorithms are represented by the d dimensional vectors corresponding to the centroids. For non-centroid based algorithms each prototype is described by a binary vector indicating whether each data-point belonged to the cluster or not. The clustering algorithm we use for exposition purposes and for our experiments is KMeans, but any fast clustering algorithm that finds clusters with the desired properties can be used. These properties are described in detail later in this section.

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 [6]. If the underlying clustering algorithm is centroid-based (say KMeans), then its measure of distance (Euclidean) is used for computing the pair-wise distances between prototypes. For instance, the corresponding Bregman divergence is used for Bregman Hard Clustering [1]. For non-centroid based algorithms, pair-wise distances between prototypes can be defined in terms of the Tanimoto coefficient (Intersection/Union) between rows of C. Then 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 [8].

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; the sum of the data-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, this process helps eliminate many clusters that formed solely of noisy data-points and outliers.

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.

Rationale Behind CLUMP

Our approach tries to mitigate the ill-effects of the greediness and computational complexity of 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 better demarcated than in the original data. This would ensure that greedy agglomerative 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 $k \approx 2$ to 3 times true number of clusters) 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 relatively spread out in the non-dense regions. Therefore, KMeans centroids serve our requirements of prototypes of the data very well. Prototypes are then grouped into meta-clusters using the single-link agglomerative algorithm which helps our approach find arbitrarily complex shapes 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 represent 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 in practice it is advisable to 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. The second parameter is the number of clusters for each KMeans run. We show in Section 4 how these parameters can be set.

3. Related Work

Fred and Jain [4] introduced the Evidence Accumulation (EA) framework where, like our approach, they over-cluster the dataset multiple times to obtain a series of clusterings. But unlike our approach, the ensemble of clusterings are used to induce a pairwise similarity space between objects, which are then clustered agglomeratively. This makes EA impractical for large datasets. Gap Statistic [10] 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.

Other notable efforts on scalable and robust clustering that bear some resemblance to our approach are Zhang *et al's* BIRCH [11], Guha *et al's* CURE [5], Strehl and Ghosh's MCLA [9], and Bradley and Fayyad's work [2]. For lack of space, readers are refered to [7] and the original references for detailed descriptions of the algorithms.

4. Experiments on Real Datasets

In this section we compare CLUMP to Evidence Accumulation (EA) [4], and KMeans (with k guessed using the Gap Statistic [10]) on a variety of datasets.

Evaluation Measures

Our approach seeks to discover natural clusters in the data. Many researchers use extrinsic information 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 multiple 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 which will give more insight into the quality of clusterings found.

Number of Clusters (k) reports the average number of clusters detected by the algorithms over multiple runs. Comparing this to the true number of clusters can indicate the accuracy of the clustering obtained. **Classification via Clustering (CVC)** is simply the classification accuracy assuming all members of a cluster were predicted to be members of the majority class in that cluster. 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)** of a clustering *w.r.t* the class labels was introduced by Strehl and Ghosh [9] and is frequently used for evaluating clusterings. Detailed explanations of these measures are given in [7].

Datasets Used

To objectively evaluate our approach against EA and KMeans+Gap we used 10 datasets with widely differing properties to reduce any bias because of dataset selection. Seven of these datasets (Iris, Wine, Glass, Pendigits, Wisconsin Breast Cancer, Waveform, and Vowel) are available at the UCI Machine Learning repository [1], while 8D5K was used in [9] and can be obtained from `www.strehl.com`. For text-data experiments we used different subsets of the 20Newsgroup dataset [2]. EA and KMeans+Gap were not evaluated on text-data, because of computational costs. Properties of these datasets are described in detail in [7].

Comparison with EA and KMeans+Gap

Table 1 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 $r = 200$, $k^{(q)} = \lfloor \sqrt{n} \rfloor$ and $t = 0.5$ as recommended in [4]. 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 1 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.

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

The results in Table 1 show that CLUMP was overall the best of the three techniques compared. On all datasets, except Pendigits, CLUMP performed statistically better than EA. On comparison with the KMeans+Gap, CLUMP performed statistically better on 6 out of 8 datasets. This might be because KMeans' underlying modeling assumption of spherical Gaussians does 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.

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. Figure 1(a) and 1(b) report results on two datasets. Similar trends were seen in other datasets as well. According to the graphs as we increase the number of noisy points in the data, the accuracy of EA drops drastically, while CLUMP's accuracy remains unaffected. These results confirm our intuition that because CLUMP clusters

[1] `www.ics.uci.edu/\~{}mlearn/MLRepository.html`

[2] `http://kdd.ics.uci.edu/`

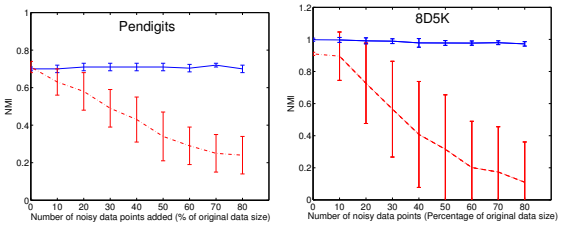

Figure 1. Plot of NMI with increasing levels of noise in the dataset.

Datasets		Sp-KMeans Best	Sp-KMeans Avg./Std Dev.	CLUMP Avg./Std Dev.
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 2. Results are averaged over 100 runs. Sp-KMeans was provided $k = 3$ and its best result over of 100 runs is also reported.

prototypes instead of the original data-points, it is more resilient to noise in the dataset.

Parameters Values

The parameter r controls the number of times KMeans is run to obtain the prototypes. For most datasets the change in the value of NMI with r 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. We observe that using a value of $k^{(q)}$ that is roughly 2 to 3 times the true value of k helps us get the best clusterings. The data based on which these observations are made is omitted here due to lack of space and is presented in [7].

Experiments with Text

For these experiments we used subsets of the 20-Newsgroup dataset. 20News-diff contains 1000 documents from 3 classes which are very different in their content, while documents in the 3 classes in 20News-sim are similar in content. The details about these datasets can be obtained from [7].

In order to effectively obtain cluster prototypes in such high-dimensional space, we used Spherical KMeans (Sp-KMeans) [3]. We used the same distance measure as Sp-KMeans to obtain pair-wise distance between the prototypes for agglomerative clustering. In these experiments, Sp-KMeans is provided with the true number of clusters $k = 3$ while CLUMP tried to discover it. For both datasets, CLUMP was run with parameters $k^{(q)} = 10$ and $r = 15$.

As seen in Table 2 The average number of clusters found by CLUMP over 100 runs is equal to the correct number of clusters for the 20News-diff dataset which is remarkable since CLUMP summarized 3000 documents with 150 prototypes. The slightly lower average accuracy as compared to Sp-KMeans is because in certain runs CLUMP was unable to identify the true number of clusters while Sp-KMeans was provided with $k = 3$ for every run. The20News-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 in terms of NMI and CVC. 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.

Acknowledgments: This work was supported by NSF Grant IIS-0307792.

References

[1] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. In *SIAM International Conference on Data Mining (SDM)*, 2004.

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

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

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

[5] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *SIGMOD '98*, pages 73–84, 1998.

[6] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall Inc., 1988.

[7] K. Punera and J. Ghosh. Clump: A scalable and robust framework for structure discovery. Technical report, Electrical and Computer Engineering, University of Texas at Austin, [`www.lans.ece.utexas.edu/~kunal/papers/icdm05-clump-long.pdf`], 2005.

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

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

[10] 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.

[11] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In *SIGMOD '96*, pages 103–114, 1996.