

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Special Issue: International Conference on Advanced Research Methodology Held by The International Association for
Scientific Research & Engineering Technology, India

An Efficient Supervised Document Clustering Approach Using Seeds Affinity Propagation

P.Yogendra Prasad¹PG Student,
Dept. of CSE, CREC, Tirupathi,**J.Naga Munciah²**Associate Professor,
Dept. of CSE, CREC, Tirupathi

Abstract: The objective is to find among all partitions of the data set, best publishing according to some quality measure. Affinity propagation is a low error, high speed, flexible, and remarkably simple clustering algorithm that may be used in forming teams of participants for business simulations and experiential exercises, and in organizing participant's preferences for the parameters of simulations. This paper proposes an efficient Affinity Propagation algorithm that guarantees the same clustering result as the original algorithm after convergence. The heart of our approach is (1) to prune unnecessary message exchanges in the iterations and (2) to compute the convergence values of pruned messages after the iterations to determine clusters.

I. INTRODUCTION

The problem of clustering has been studied widely in the database and statistics literature in the context of a wide variety of data mining tasks. The clustering problem is defined to be that of finding groups of similar objects in the data. The similarity between the objects is measured with the use of a similarity function. The problem of clustering can be very useful in the text domain, where the objects to be clusters can be of different granularities such as documents, paragraphs, sentences or terms. Clustering is especially useful for organizing documents to improve retrieval and support browsing. There are so many feature selection and transformation methods.

1.1 Document Frequency-based Selection

The simplest possible method for feature selection in document clustering is that of the use of *document frequency* to filter out irrelevant features. While the use of inverse document frequencies reduces the importance of such words, this may not alone be sufficient to reduce the noise effects of very frequent words. In other words, words which are too frequent in the corpus can be removed because they are typically common words such as “a”, “an”, “the”, or “of” which are not discriminative from a clustering perspective. Such words are also referred to as *stop words*. Typically commonly available stop word lists of about 300 to 400 words are used for the retrieval process. In addition, words which occur extremely infrequently can also be removed from the collection. This is because such words do not add anything to the similarity computations which are used in most clustering methods.

In some cases, such words may be misspellings or typographical errors in documents. Noisy text collections which are derived from the web, blogs or social networks are more likely to contain such terms. We note that some lines of research define document frequency based selection purely on the basis of very infrequent terms, because these terms contribute the least to the similarity calculations. However, it should be emphasized that very frequent words should also be removed, especially if they are not discriminative between clusters. Note that the TF-IDF weighting method can also naturally filter out very common words in a “soft” way. Clearly, the standard set of stop words provide a valid set of words to prune. Nevertheless, we would like

a way of quantifying the importance of a term directly to the clustering process, which is essential for more aggressive pruning. We will discuss a number of such methods below.

1.2. Term Strength

The core idea of this approach is to extend techniques which are used in supervised learning to The unsupervised case. The term strength is essentially used to measure how informative a word is for identifying two related documents. For example, for two related documents x and y , the term strength $s(t)$ of term t is defined in terms of the following probability:

$$s(t) = \frac{P(t \text{ in } x \text{ and } y)}{P(t \text{ in } x)} \quad (1)$$

Clearly, the main issue is how one might define the document x and y as related. One possibility is to use manual (or user) feedback to define when a pair of document are related. This is essentially equivalent to utilizing supervision in the feature selection process, and may be practical in situations in which predefined categories of documents are available. On the other hand, it is not practical to manually create related pairs in large collections in a comprehensive way. It is therefore desirable to use an automated and purely unsupervised way to define the concept of when a pair of documents is related. A pair of documents are defined to be related if their cosine similarity is above a user-defined threshold. In such cases, the term strength $s(t)$ can be defined by randomly sampling a number of pairs of such related documents as follows:

$$s(t) = \frac{\text{Number of pairs in which } t \text{ occurs in both}}{\text{Number of pairs in which } t \text{ occurs in the first of the pair}} \quad (2)$$

Here, the first document of the pair may simply be picked randomly. In order to prune features, the term strength may be compared to the expected strength of a term which is randomly distributed in the training documents with the same frequency. If the term strength of t is not at least two standard deviations greater than that of the random word, then it is removed from the collection.

1.3. Similarity Measures

Before clustering, a similarity/distance measure must be determined. The measure reflects the degree of closeness or separation of the target objects and should correspond to the characteristics that are believed to distinguish the clusters embedded in the data. In many cases, these characteristics are dependent on the data or the problem context at hand, and there is no measure that is universally best for all kinds of clustering problems. Moreover, choosing an appropriate similarity measure is also crucial for cluster analysis, especially for a particular type of clustering algorithms. Density-based clustering finds clusters as dense areas in the data set, and the density of a given point is in turn estimated as the closeness of the corresponding data object to its neighboring objects. Recalling that closeness is quantified as the distance/similarity value, we can see that large number of distance/similarity computations are required for finding dense areas and estimate cluster assignment of new data objects. Therefore, understanding the effectiveness of different measures is of great importance in helping to choose the best one.

Euclidean distance is a standard metric for geometrical problems. It is the ordinary distance between two points and can be easily measured with a ruler in two- or three-dimensional space. Euclidean distance is widely used in clustering problems, including clustering text. It satisfies all the above four conditions and therefore is a true metric. It is also the default distance measure used with the K-means algorithm. Measuring distance between text documents, given two documents d_a and d_b represented by their term vectors and t_i respectively, the Euclidean distance of the two documents is defined as

Where the term set is $T = \{t_1, \dots, t_m\}$. As mentioned previously, we use the t_i idf value as term weights.

Affinity Propagation is derived as an application of the max-sum algorithm in a factor graph ,i.e., it searches for the minima of an energy function on the basis of message passing between data points [7]. The clustering performance depends on the similarity measure and message updating frequency. For its simplicity, general applicability, and good performance, AP has

already been used in text clustering. By using AP to preprocess texts, Maet al. developed an incremental method [11] for text clustering. Wang et al. combined AP with a parallel strategy for e-learning resources clustering [12]. However, they used AP only as an unsupervised algorithm and did not consider any structural information derived from the specific documents. For text mining tasks, the majority of state-of-the-art frameworks employ the vector space model (VSM), which treats a document as a bag of words and uses plain language words as features [13], [14].

This model can represent the text mining problems easily and directly. However, with the increase of data set size, the vector space becomes high dimensional, sparse, and the computational complexity grows exponentially. Moreover, in many practical applications, completely unsupervised learning is lacking relevant information. On the other hand, supervised learning needs an initial large number of class label information, which requires expensive human labor and time [15], [16]. Therefore, in recent years, semi supervised learning has captured a great deal of attentions [17], [18], [19], [20], [21]. Semi supervised learning is a machine learning paradigm in which the model is constructed using both labeled and unlabeled data for training—typically a small amount of labeled data and a large amount of unlabeled data [16], [22]. To examine the effectiveness of the proposed method, we have applied it to the benchmark data set Reuters-21578. In order to analyze the behavior of the new algorithm (and also the impact of the two individual proposed contributions), we have performed a detail comparison with four clustering methods on the same data set, namely,

1. K-Medoids Approach
2. Fast Algorithm of Affinity Propagation Approach.

II. RELATED WORK

2.1. Affinity Propagation

Affinity Propagation is a clustering algorithm that identifies a set of 'exemplars' that represents the dataset [Frey and Dueck, 2007]. The input of Affinity Propagation is the pair-wise similarities between each pair of data points, $s[i, j]$ ($i, j = 1, 2, \dots, N$). Any type of similarities is acceptable, e.g. negative Euclidean distance for real valued data and Jaccard coefficient for non-metric data, thus Affinity Propagation is widely applicable. Given similarity matrix $s[i, j]$, Affinity Propagation attempts to find the exemplars that maximize the net similarity, i.e. the overall sum of similarities between all exemplars and their member data points.

The process of Affinity Propagation can be viewed as a message passing process with two kinds of messages exchanged among data points: responsibility and availability. Responsibility, $r[i, j]$, is a message from data point i to j that reflects the accumulated evidence for how well-suited data point j is to serve as the exemplar for data point i . Availability, $a[i, j]$, is a message from data point j to i that reflects the accumulated evidence for how appropriate it would be for data point i to choose data point j as its exemplar. All responsibilities and availabilities are set to 0 initially, and their values are iteratively updated as follows to compute convergence values:

$$\begin{aligned} r[i, j] &= (1 - \lambda)\rho[i, j] + \lambda r[i, j] \\ a[i, j] &= (1 - \lambda)\alpha[i, j] + \lambda a[i, j] \end{aligned} \quad (2)$$

Where λ is a damping factor introduced to avoid numerical oscillations, and $\rho[i, j]$ and $\alpha[i, j]$ are, we call, propagating responsibility and propagating availability, respectively. $\rho[i, j]$ and $\alpha[i, j]$ are computed.

That is, messages between data points are computed from the corresponding propagating messages. The exemplar of data Point i is finally defined as:

$$\text{argmax}\{r[i, j] + a[i, j] : j=1, 2, \dots, N\} \quad (3)$$

As described above, the original algorithm requires $O(N^2T)$ time to update messages, where N and T are the number of data points and the number of iterations, respectively. This incurs excessive CPU time, especially when the number of data points is large. Therefore, a fast Affinity Propagation algorithm is demanded as pointed out in [Jia et al.,2008]. In the existing algorithms for the K-means problem, we find that AP performs at least as well as the competing algorithms in terms of quality. However, due to a memory footprint of $O(N^2)$, the algorithm cannot be applied on datasets where the number of data points N is large. Another reason why AP is not very suited for large N is its $O(N^2)$ scaling of the runtime per iteration.

The K-means algorithm and deterministic annealing (DA) have a runtime that scales with $O(NKD)$. Therefore, when the dimension D and number of clusters K is small, DA and K-means have a much lower runtime. We observe, that AP's runtime is mostly independent of the dimension D and the number of clusters K . That means, when K and D is large, e.g $K = 50$ and $D = 100$, AP can be much faster than K-means algorithm and DA.

Also, the K-means algorithm is not only slow for large K but has severe problems to find good solutions. Hence, AP works well in settings where the K-means algorithm has problems. Compared to hierarchical clustering algorithms, e.g. Ward's method, AP generally runs much slower. When clusters are well-defined and there is only little noise in the dataset, the performance is comparable. If that is not the case, AP finds better solutions.

III. PROPOSED SYSTEM

3.1. K-Nearest neighbourhood algorithm

The theorem presented in the last section shows sufficient conditions under which clustering can be performed consistently. Now we want to present a generic algorithm which can be used to minimize arbitrary clustering objective functions. With help of Theorem 1 we can then prove the consistency of its results for a large variety of clustering objective functions.

We have seen that the key to obtain consistent clustering schemes is to work with an appropriate Function class. But of course, given quality functions Q and Q_n , the question is how such a function space can be constructed in practice. Essentially, three requirements have to be satisfied:

- » The function space F_n has to be "small". Ideally, it should only contain polynomial functions.
- » The function space F_n should be "rich enough". In the limit $n \rightarrow \infty$, we would like to be able to approximate any (reasonable) measurable function.
- » We need to be able to solve the optimization problem $\text{argmin}_{f \in F_n} Q_n(f)$.

This sounds trivial at first glance, but in practice is far from easy. One rather straightforward way to achieve all requirements is to use a function space of piecewise constant functions. Given a partitioning of the data space in small cells, we only look at clustering's which are constant on each cell (that is, the clustering never splits a cell). If we make sure that the number of cells is only of the order $\log(n)$, then we know that the number of clustering is at most $K \log(n) = n \log(K)$, which is polynomial in n . In the following we will introduce a data-dependent random partition of the space which turns out to be very convenient.

We will construct a function class F_n as follows. Given a finite sample $X_1, \dots, X_n \in \mathbb{R}^d$, the number K of clusters to construct, and a number $m \in \mathbb{N}$ with $K \leq m \ll n$, randomly pick a subset of m "seed points" X_{s1}, \dots, X_{sm} . Assign all other data points to their closest seed points, that is for all $j = 1, \dots, m$ define the set Z_j as the subset of data points whose nearest seed point is X_{sj} . In other words, the sets Z_1, \dots, Z_m are the Verona cells induced by the seeds X_{s1}, \dots, X_{sm} . Then consider all partitions of X_n which are constant on all the sets Z_1, \dots, Z_m . More formally, for given seeds we define the set F_n as the set of all functions.

Obviously, the function class F_n contains K_m functions, which is polynomial in n if the number m of seeds satisfies $m \in O(\log n)$. Given F_n , the most simple polynomial-time optimization algorithm is then to evaluate $Q_n(f)$ for all $f \in F_n$ and choose

the solution $f_n = \operatorname{argmin}_{f \in F_n} Q_n(f)$. We call the resulting clustering the nearest neighbor clustering and denote it by $NNC(Q_n)$. The entire algorithm is summarized in Figure 1. We have already published results on the empirical performance.

3.2. Nearest Neighbor Clustering $NNC(Q_n)$, naive implementation

Parameters: number K of clusters to construct, number $m \in \mathbb{N}$ of seed points to use (with $K \leq m \ll n$), clustering quality function Q_n

Input: data set $X_n = \{X_1, \dots, X_n\}$, distances $d_{ij} = d(X_i, X_j)$

- » Subsample m seed points from the data points, without replacement.
- » Build the Voronoi decomposition Z_1, \dots, Z_m of X_n based on the distances d_{ij} using the seed points as centers.
- » Define $F_n := \{f: X_n \rightarrow \{1, \dots, K\} \mid f \text{ constant on all cells } Z_j\}$
- » For all $f \in F_n$ evaluate $Q_n(f)$.

Output: $f_n := \operatorname{argmin}_{f \in F_n} Q_n(f)$

a) Fast Seeds Affinity Propagation

For resolving the computation time issue of the original Affinity Propagation algorithm, Jia et al. recently proposed FSAP[Jia et al., 2008]. One promising idea for improving the speed of Affinity Propagation is to reduce the number of message values that need to be computed. FSAP aims to reflect this idea as follows. The first stage of FSAP constructs a K -nearest neighbor graph. If data point i is among the K data points that have the largest similarity with data point j , then data point i and j are connected by an edge, otherwise not. Since FSAP performs message transmissions on the K nearest neighbor graph, too many exemplars (at least N/K) might be generated. Therefore, in order to merge multiple exemplars into one cluster, the second stage adds further edges based on the following three criteria:

1. If data point i is the exemplar of data point j , then data point i and j are connected by an edge;
2. For two data points i and j , if there exists two data points m and n that take data point i and j as their exemplar, respectively, and data point m and n are K -nearest neighbor to each other, and so data point i and j are connected by an edge; and
3. For two data points i and j , if they are connected by criterion 2, then all data points that choose data point i as exemplar are connected to data point j , and vice versa. After convergence, the exemplar of data point i is finally determined by Equation (5). They showed that their approach is much faster than the original algorithm described in Section 2. However, FSAP is based on heuristic ideas, i.e., the linked edges are determined based on K -nearest neighbor approximation and heuristic criteria. Therefore, FSAP does not guarantee the same result as the original Affinity Propagation algorithm. Our algorithm presented in this paper is faster than FSAP while it still theoretically guarantees the exactness of the clustering results after convergence.

Algorithm:

Input: pair-wise similarities

Output: exemplars of each data point

- 1: for each data point pair $[i, j]$ do
- 2: compute $a[i, j]$, $r[i, j]$, and $a[i, j]$ by Equation (5-7);
- 3: end for

```

4: for each data point pair [i, j] do
5: if  $r[i, j] \geq 0$  or  $a[i, j] + s[i, j] \geq \max_{k=j} \{a[i, k] + s[i, k]\}$ ; then
6: link data point pair [i, j];
7: end if
8: end for
9: for t = 1 to T do
10: for each linked data point pair [i, j] do
11: update  $r[i, j]$  and  $a[i, j]$  by Equation (1);
12: end for
13: end for
14: for each unlinked data point pair [i, j] do
15: compute  $r[i, j] = \rho[i, j]$  and  $a[i, j] = \alpha[i, j]$ ;
16: end for
17: for each data point i do
18: compute exemplar by Equation (4);
19: end for

```

IV. CONCLUSION

Our Affinity propagation approach improves clustering process. We used K-nearest neighborhood method to calculate the distances between the words and then fast affinity propagation used for clustering. It reduces unnecessary message exchanges in the iterations, and reduces the convergence values of reduced messages from those of un-pruned messages. Experiments show that our algorithm can achieve efficient clustering without sacrificing clustering accuracy. Affinity propagation is a low error, high speed, flexible, and an easy-to-code clustering algorithm that identifies clusters, exemplars, and outliers.

References

1. Y.J. Li, C. Luo, and S.M. Chung, "Text Clustering with Feature Selection by Using Statistical Data," IEEE Trans. Knowledge and Data Eng., vol. 20, no. 5, pp. 641-652, May 2008.
2. B.J. Frey and D. Dueck, "Clustering by Passing Messages between Data Points," Science, vol. 315, no. 5814, pp. 972-976, Feb. 2007.
3. B.J. Frey and D. Dueck, "Non-Metric Affinity Propagation for Un-Supervised Image Categorization," Proc. 11th IEEE Int'l Conf. Computer Vision (ICCV '07), pp. 1-8, Oct. 2007.
4. L. Michele, Sumedha, and W. Martin, "Clustering by Soft-Constraint Affinity Propagation Applications to Gene-Expression Data," Bioinformatics, vol. 23, no. 20, pp. 2708-2715, Sept. 2007.
5. T.Y. Jiang and A. Tuzhilin, "Dynamic Micro Targeting: Fitness- Based Approach to Predicting Individual Preferences," Proc. Seventh IEEE Int'l Conf. Data Mining (ICDM '07), pp. 173-182, Oct. 2007.
6. H.F. Ma, X.H. Fan, and J. Chen, "An Incremental Chinese Text Classification Algorithm Based on Quick Clustering," Proc. 2008 Int'l Symp. Information Processing (ISIP '08), pp. 308- 312, May 2008.
7. W.H. Wang, H.W. Zhang, F. Wu, and Y.T. Zhuang, "Large Scale of E-Learning Resources Clustering with Parallel Affinity Propagation," Proc. Int'l Conf. Hybrid Learning 2008 (ICHL '08), pp.1-10, Aug. 2008.
8. F. Wang and C.S. Zhang, "Label Propagation through Linear Neighbourhoods," IEEE Trans. Knowledge and Data Eng., vol. 20, no. 1, pp. 55-67, Jan. 2008.
9. Z.H. Zhou and M. Li, "Semi-Supervised Regression with Co-Training Style Algorithms," IEEE Trans. Knowledge and Data Eng., vol. 19, no. 11, pp. 1479-1493, Aug. 2007.
10. S. Yu, B. Krishnapuram, R. Rosales, H. Steck, and R.B. Rao, "Bayesian Co-Training," Advances in Neural Information Processing Systems, vol. 20, pp. 1665-1672, MIT Press, 2008.

11. Z.H. Zhou, D.C. Zhan, and Q. Yang, "Semi-Supervised Learning with Very Few Labeled Training Examples," Proc. 22nd AAAI Conf. Artificial Intelligence, pp. 675-680, 2007.
12. O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. MIT Press, 2006
13. L.P. Jing, M.K. Ng, and J.Z. Huang, "An Entropy Weighting Kmeans Algorithm for Subspace Clustering of High-Dimensional Sparse Data," IEEE Trans. Knowledge and Data Eng., vol. 19, no. 8, pp. 1026-1041, Aug. 2007.
14. Hung-Leng Chen, Kun-Ta Chuang, and Ming-Syan Chen, "On Data Labeling for Clustering Categorical Data", IEEE transactions on knowledge and data engineering, vol. 20, no. 11, November 2008
15. Yongli Liu, Yuanxin Ouyang, Hao Sheng, Zhang Xiong, " An Incremental Algorithm for Clustering Search Results", IEEE International Conference on Signal Image Technology and Internet Based Systems in 2008.
16. Hai-Dong MENG, Yu-Chen SONG, Shu-Ling WANG." An Incremental Clustering Algorithm Based on Subcluster Feature" ,The 1st International Conference on Information Science and Engineering (ICISE2009) in 2009.
17. Renchu Guan, Xiaohu Shi, Maurizio Marchese, Chen Yang and yanchun Liang, "Text Clustering with Seeds Affinity Propagation", IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No.4, April 2011.
18. Sergio M. Savaresi and Daniel L. Boley, "On the performance of bisecting K-means and PDDP", Proceedings of the 1st Sergio M. Savaresi and Daniel L. Boley, "On the performance of bisecting K-means and PDDP", Proceedings of the 1st SIAM ICDM, Chicago, IL, 2001. SIAM ICDM, Chicago, IL, 2001.
19. <http://www.psi.toronto.edu/affinitypropagation/faq.html>, 2010. [20 Z.H. Zhou and M. Li, "Distributional Features for Text Categorization," IEEE Trans. Knowledge and Data Eng., vol. 21, no. 3, pp. 428-442, Mar. 2009.
20. S. Huang, Z. Chen, Y. Yu, and W.Y. Ma, "Multitype Features Coselection for Web Document Clustering," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 4, pp. 448-458, Apr. 2006.
21. X.D. Wu et al., "Top 10 Algorithms in Data Mining," Knowledge and Information Systems, vol. 14, no. 1, pp. 1-37, Jan. 2008.
22. M.J. Brusco and H.F. Kohn, "Comment on 'Clustering by Passing Messages between Data Points,'" Science, vol. 319, no. 5864, p. 726c, Feb. 2008.
23. J. Wu, F. Ding, and Q.L. Xiang, "An Affinity Propagation Based Method for Vector Quantization," Eprint arXiv 0710.2037, <http://arxiv.org/abs/0710.2037v2>, Oct. 2007.
24. K.J. Wang, J.Y. Zhang, D. Li, X.N. Zhang, and T. Guo, "Adaptive Affinity Propagation Clustering," Acta Automatica Sinica, vol. 33, no. 12, pp. 1242-1246, Dec. 2007.