Partition Affinity Propagation for Clustering Large Scale of Data in Digital Library

Xuqing Zhang, Fei Wu, Dingyin Xia, and Yueting Zhuang

College of Computer Science, Zhejiang University, Hangzhou, 310027, P.R.China

***Abstract* —** Data clustering is very useful in helping users visit the large scale of data in digit library. In this paper, we present an improved algorithm for clustering large scale of data set with dense relationship based on Affinity Propagation. First, the input data are divided into several groups and Affinity Propagation is applied to them respectively. Results from first step are grouped together in some way, and Affinity Propagation is implemented to them. Experimental results show that our algorithm, referred to as Partition Affinity Propagation, brings an encouraging effect for speeding up Affinity Propagation in clustering dense data set, while clustering accuracy are almost kept or even better.

***Index Terms* — Algorithms, Affinity Propagation, Clustering methods, Dense Data, Experimentation, Performance.**

I. Introduction

In the era of information explosion, we are often helpless in the face of massive increasing multimedia data. The scale of some digital library has reached TB level, such as China-America Digital Academic Library (CADAL, www.cadal.zju.edu.cn). Users will get 369 results from CADAL when they query a “medicine” keyword. If they input the corresponding Chinese word “药” instead, they will get more than ten thousand results. It’s hard for users to quickly get what they want from those results.

As a result, information organization and knowledge discovery are becoming more and more important. The motivation of these methods is to mine implicit relationship among massive data and provide more convenient way for users to browse massive data. Data clustering is such a useful technique for us to analyze the semantic distribution of a large database.

The aim of data clustering is to classify the large scale of data into different groups, or more precisely, to partition a data set into subsets (clusters), so that the data in each subset (ideally) share some common features, that is to say the data in the same subset is more similar than the data belong to different subsets according to some defined distance measurement.

If the distance measurement between pairs of data points is already known, one of the most common approaches used for data clustering is K-means [2] clustering technique. The K-means clustering algorithm begins with an initial set of exemplars which are randomly selected and then iteratively refine them so as to minimize the sum of squared errors between data points and their exemplars. But it is sensitive to the initial exemplar set. So it is often rerun many times with different initial exemplar set in order to gain a good result. But with the increase of prospected number of clusters, the possibility of gaining a good initial exemplar by random selection decreases.

Recently proposed Affinity Propagation (AP) [1] by Frey is a quite different method. It takes the similarity between pairs of data points as input. Real-valued messages are exchanged between data points until a high-quality set of exemplars and corresponding clusters gradually emerge [1]. That means you need not choose the initial exemplars and you need not prespecify the number of clusters, either. Besides, the similarity between pairs of data points can be asymmetry.

For data set with dense relationship, that is to say the similarity between every two data points in the data set is finite such as calligraphy image data, the message-communication process of AP is not efficient for such dense matrix. Experimental results show that the time spent increases rapidly when the size of the data set increases, see Fig. 1.

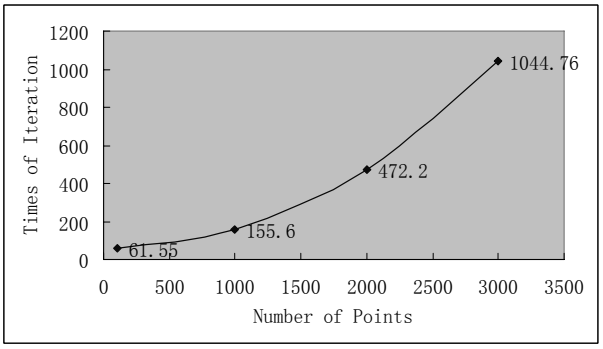

Fig. 1. Relationship between number of data points and number of iterations. Test by applying AP on random generated two dimensional data sets of size 100, 1000, 2000, 3000, each 20 groups.

For the above disadvantage, we propose an improved algorithm called partition affinity propagation (PAP) in this paper to speed up AP applied to large scale of dense data set. The aim of PAP is to reduce the computation cost spent in AP while mostly keeping the original clustering accuracy.

Section II briefly introduces the affinity propagation algorithm. Section III gives out our speed-up algorithm,

partition affinity propagation. Some experimental comparison and analysis is given at Section IV.

II. AFFINITY PROPAGATION (AP)

Affinity propagation (AP) can be viewed as a method that searches for minima of an energy function:

$$E(\mathrm{c}) = -\sum_{i=1}^{N} s(i, c_i) \tag{1}$$

that depends on a set of hidden labels, $c_1 \cdots c_N$, corresponding to the N data points. Each label c_i indicates the exemplar of the data point *i*, while $s(i, c_i)$ is the similarity between data point *i* and its exemplar. As the goal of AP is to minimize the energy function $E(\boldsymbol{c})$, $s(i, c_i)$ must be non-positive.

The process of AP can be viewed as a message communication process on a factor graph [3].

There are two kinds of messages exchanged between each pair of data points, responsibility and availability. The "responsibility" $r(i,k)$ is sent from data point *i* to candidate exemplar point *k* and reflects the accumulated evidence for how well-suited point *k* is to serve as the exemplar for point *i*, taking into account other potential exemplars for point *i*. The "availability" $a(i,k)$ is sent from candidate exemplar point *k* to point *i* and reflects the accumulated evidence for how appropriate it would be for point *i* to choose point *k* as its exemplar, taking into account the support from other points that point *k* should be an exemplar [1].

The algorithmic procedure is stated below:

ALGORITHM 1. (AP)

Suppose we are given the similarity among N data points denoted by matrix $S_{N \times N}$ where the diagonal of the matrix is the preferences of each data point. The initial value of availability matrix $A_{N \times N}$ is zero.

Step 1. Updating all responsibilities given the availabilities using the following updating rule:

$$r(i,k) \leftarrow s(i,k) - \max_{k' s.t. k' \neq k} \{a(i,k') + s(i,k')\} \tag{2}$$

Step 2. Updating all availabilities given the responsibilities using the following updating rule:

$$a(i,k) \leftarrow \min\Big\{0, r(k,k) + \sum_{i' s.t. i' \notin \{i,k\}} \max\{0, r(i',k)\}\Big\} \tag{3}$$

$$a(k,k) \leftarrow \sum_{i' s.t. i' \neq k} \max\{0, r(i',k)\} \tag{4}$$

Step 3. Combining availabilities and responsibilities to monitor the exemplar decisions. For point *i*, the value of *k* that maximizes $a(i,k) + r(i,k)$ either identifies point *i* as an exemplar if $k = i$, or identifies the data point that is the exemplar for point *i*.

Step 4. Terminate the algorithm when these decisions did not change for a certain number of iterations or maximal number of iterations reaches.

$s(i,i)$ is the input preference for data point *i*, indicating how suitable data point *i* can be the exemplar. The possibility of data point *i* as an exemplar partly depends on $s(i,i)$. The larger the $s(i,i)$, the bigger the possibility of data point *i* as an exemplar. Besides, input preferences influence the number of clusters, too.

However, the clustering result depends not only on input preferences, but also on message passing procedure.

When updating the messages, numerical oscillations must be taken into consideration. As a result, each message is set to λ times its value from the previous iteration plus $1-\lambda$ times its prescribed updated value. λ should be larger or equal than 0.5 and less than 1. If λ is very close to 1, numerical oscillation may be avoided. But no guarantee is given to the convergence of the algorithm [4]. Therefore, a maximal number of iterations is set to avoid infinite iteration.

III. PARTITION AFFINITY PROPAGATION (PAP)

If the similarity between each pair of data points in the data set is finite, we say this data set is of dense relationship such as calligraphy image data. For data set with dense relationship the message-communication process of AP is done in a dense matrix. The computation overload spent is in the direct ratio to the number of iterations.

For an iteration of AP, each element in the responsibility matrix $r(i,k)$ must be calculated for once and each calculation must be applied on $N-1$ elements, where *N* is the size of the input similarity matrix, according to (2). The calculation of each element in the availability matrix is alike.

The algorithm we proposed would shorten the time spent by decreasing the number of iterations given the $N \times N$ similarity matrix.

The procedure is stated below:

ALGORITHM 2. (PAP)

Suppose we are given the similarity of N data points denoted by matrix $S_{N \times N}$ which is the same as the input of AP.

Step 1. Partition the matrix $S_{N \times N}$ into *k* parts averagely as following:

$$S = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1k} \\ S_{21} & S_{22} & & S_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ S_{k1} & S_{k2} & \cdots & S_{kk} \end{bmatrix}$$

k must be larger than 1 and less than $\lfloor N/(4\times C)\rfloor$, where C is the maximal number of clusters prospected.

So the sub-matrix $S_{11}, S_{22}, \cdots S_{kk}$ are all square matrices, and the size of S_{11} to $S_{k-1,k-1}$ is $\lfloor N/k \rfloor$ which is an integer less than or equal to N/k, the size of S_{kk} is $N-(k-1)*\lfloor N/k \rfloor$.

Step 2. Take sub-matrix $S_{11}, S_{22}, \cdots S_{kk}$ as the input of AP, and implement AP on them respectively. Then we get k availability matrices $A_{11}, A_{22}, \cdots A_{kk}$.

Step 3. Merge $A_{11}, A_{22}, \cdots A_{kk}$ together using the following rule and form the availability matrix of the whole data set:

$$A' = \begin{bmatrix} A_{11} & & & \\ & A_{22} & & \\ & & \ddots & \\ & & & A_{kk} \end{bmatrix}$$

The value of remaining part of matrix A is set to zero.

Step 4. Run AP with A as the initial availability matrix.

In step1 and step 2, the whole data set is divided into k groups and AP is run on them respectively. The resulting availability matrices $A_{ii}(i=1,\cdots k)$ are used to construct an intermediate availability matrix A. Then the AP clustering of the whole data set is run using A as the initial availability matrix.

In most cases, clustering result is gained in less number of iterations than running AP clustering on the whole data set from the very beginning.

Further more, message passing procedure is going on among all the data points in the last step, so the clustering result is similar to do the AP clustering on the whole data set from the very beginning, or even better.

In step 2, the size of S_{ii} is about $1/k^2$ of the original one, so one iteration of S_{ii} spends about $1/k^2$ the time of the original one according to (2), (3) and (4). With the decrease of the size of similarity matrix, the average number of iterations taken for convergence also decreases as shown by Fig. 1. When size of input similarity matrix S and number of partition parts k increase to some extent, the time taken in step 2 can be ignored.

According to above analysis, the total time taken for one run of PAP is less than using AP directly on the whole data set from the very beginning.

IV. EXPRIMENTS

In this section we present a set of experiments to verify the speed-up effect of our proposed algorithm.

A. Data Set

The algorithm performance is evaluated on three kinds of data set:

1. Two dimensional data points generated by random sampling.
2. Three dimensional manifold subspace data points: Swiss Roll, by rolling up a randomly sampled plane into a spiral [5].
3. China-America Digital Academic Library [6] calligraphy database. We use the method mentioned in [7] to extract the high-dimensional features of the images in the database which each image contains exactly one Chinese character.

The negative of squared Euclidian distance is used as the similarity between each pair of data points.

B. Evaluation Metric

We utilize number of iterations and the total time used for one run on AP and PAP as the evaluation metric for clustering speed.

One run of AP will terminate only when exemplar decisions keep the same for a certain number of iterations or maximal number of iterations reaches. In the following experiments, we set this "certain number of iterations" as 50.

C. Performance Comparison

1. Two dimensional data sets test

We test two dimensional data set of size 100, 1000, 2000 and 3000, each 20 groups. AP and PAP with k = 8, are run on each group.

Table I shows the average number of iterations of AP and the average number of iterations in step 4 of PAP with k = 8. The final 50 iterations are not included. Table II shows the average time spent by AP and PAP. The speed-up effect is obvious. And we can see from it that when the size of data set reaches certain scale, the time spent on step 2 of PAP can be ignored. When the size of data set reaches 3000, the number of iterations in step 4 of PAP and the total time spent of PAP are almost the same percent of those of AP.

TABLE I
STATISTICS OF NUMBER OF ITERATIONS OF AP AND PAP IN STEP 4

Number	Iteration AP	Iteration PAP	Iteration PAP/AP
100	11.45	9.9	86%
1000	105.6	93.8	89%
2000	422.2	162.9	39%
3000	973.1	559.3	58%

TABLE II
STATISTICS OF ITERATION TIME OF AP AND PAP

number	Time(s) AP	Time(s) PAP	Time PAP/AP
100	0.1	0.2	276%
1000	37.1	35.1	95%
2000	346.3	154.4	45%
3000	1916.4	1139.6	59%

Table III shows the test results of running AP and PAP with k = 2, 4, 8, 16 on the first ten groups of size 3000 which the final 50 iterations are not included. It indicates that the speed-up effect got from PAP is dependent on the value k. If k is well chosen, the average number of iterations in step 4 of PAP is only 28% of AP. However, if not, numerical oscillations might happen such as group 1. PAP with k = 2 does not converge on data set of group 1 after maximal iteration, 1500.

How to choose the most appropriate value k to gain the best speed-up effect is one of our further researching tasks.

TABLE III
STATISTICS OF NUMBER OF ITERATIONS OF AP AND PAP IN STEP 4 RUNNING ON 10 GROUPS OF TWO-DIMENSIONAL DATA POINTS OF SIZE 3000 GENERATED BY RANDOM SAMPLING

	AP	PAP (k=2)	PAP (k=4)	PAP (k=8)	PAP (k=16)	Min (PAP)
1	936	1450	623	597	263	263
2	1118	1450	569	296	1337	296
3	721	624	520	693	1450	520
4	1209	668	1102	462	314	314
5	851	436	967	961	1323	436
6	1450	353	257	321	161	161
7	1081	816	523	270	828	270
8	1450	1450	473	468	195	195
9	894	940	468	626	939	468
10	1158	842	428	554	125	125
Sum	10868	9029	5930	5248	6935	3048
Per-cent	100%	83%	55%	48%	64%	28%

2. Three dimensional manifold subspace data set: Swiss Roll test

We test Swiss Roll data set of size 2000.

Fig. 2 shows the clustering result of applying AP on the Swiss Roll data set. Fig. 3 shows the clustering result of applying PAP with k = 8 on the same data set. We can see that PAP not only maintains the original clustering accuracy, sometimes it can even improve it. The misclustering on top middle of Fig. 2 disappeared in Fig. 3.

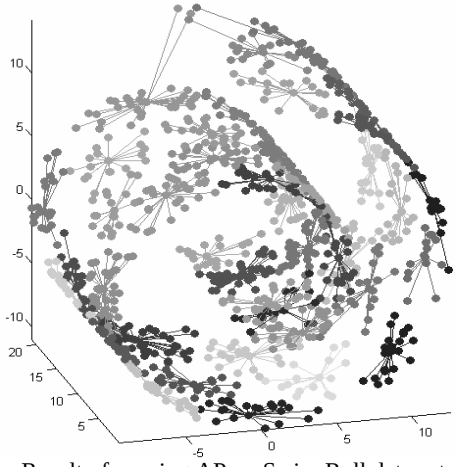

Fig. 2. Result of running AP on Swiss Roll data set

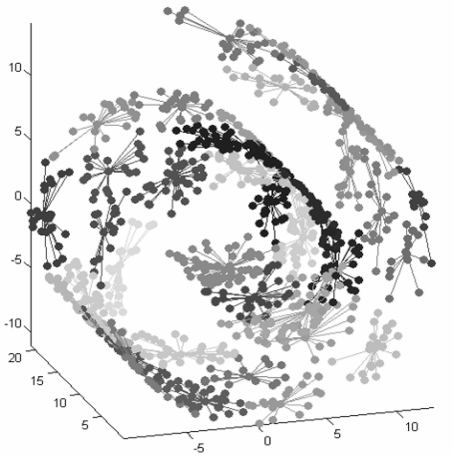

Fig. 3. Result of running PAP with k = 8 on Swiss Roll data set

3. China-America Digital Academic Library calligraphy database test

We select 1024 images from the database, including 38 distinct characters.

Table IV is the statistical figures of the test which the final 50 iterations are not included. It shows the effectiveness of PAP on real-world data set.

TABLE IV
STATISTICS OF RUNNING AP AND PAP ON 1024 IMAGES FROM CHINA-AMERICA DIGITAL ACADEMIC CALLIGRAPHY DATABASE

	cluster	iteration	Percent	error
AP	39	36	100%	-366.674
PAP(k=2)	39	19	53%	-366.799
PAP(k=4)	38	38	106%	-319.016
PAP(k=8)	37	43	113%	-366.674
PAP(k=16)	38	23	61%	-366.445

Fig. 4 shows part of clustering result from running PAP with k = 2. Image belongs to the same cluster is put in the same document.

Fig. 4. Result of running PAP on data set from China-America Digital Academic Calligraphy database

Users of the CADAL calligraphy database may input a calligraphy image to query the similar images. Figures of some Chinese characters, such as the third column of the first row and the fifth column of the first row, are similar. If we can return the clustered results, it would bring much convenience to the users. So they can browse the result data more quickly.

V. Conclusion

Data clustering is very useful in digit library. Affinity propagation is such an algorithm for data clustering. To accelerate affinity propagation algorithm for clustering large scale of data set with dense relationship while maintaining its accuracy, an optimal algorithm, Partition Affinity Propagation, is proposed in this paper. PAP speeds up affinity propagation by decrease the number of iterations on the whole similarity matrix. Experiments verified the effectiveness of the proposed algorithm.

Acknowledgement

This work is supported by National Natural Science Foundation of China (No.60533090, No.60525108), and Technology Project of Zhejiang Province (2005C13032，2006C13097), and program for Changjiang Scholars and Innovative Research Team in University (IRT0652, PCSIRT)

References

[1] Brendan J. Frey and Delbert Dueck, “Clustering by passing messages between data points,” *Science*, 315(5814): 972- 951, 2007

[2] J. MacQueen, in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. Le Cam, J. Neyman, Eds. (university. of California Press, Berkeley, CA, 1967), vol. 1, pp. 281-297.

[3] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss, “Understanding belief propagation and its generalizations,” *International Joint Conference on Artificial Intelligence* in August 2001, TR-2001-22 January 2002

[4] http://vision.ucsd.edu/~kbranson/research/bp_pres.pdf

[5] http://www.math.umn.edu/~wittman/mani/index.html

[6] http://www.cadal.net/

[7] Yueting Zhuang, Xiafen Zhang, Jiangqin Wu, Xiqun Lu, “Retrieval of Chinese Calligraphic Character Image”, *PCM 2004*, 17-24