Pattern Recognition ■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Subspace clustering using affinity propagation

Guojun Gan [a,*], Michael Kwok-Po Ng [b]

[a] Department of Mathematics, University of Connecticut, 196 Auditorium Rd U-3009, Storrs, CT 06269, USA
[b] Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong

ARTICLE INFO

Article history:
Received 13 June 2014
Received in revised form
3 September 2014
Accepted 1 November 2014

Keywords:
Data clustering
Subspace clustering
Affinity propagation
Attribute weighting

ABSTRACT

This paper proposes a subspace clustering algorithm by introducing attribute weights in the affinity propagation algorithm. A new step is introduced to the affinity propagation process to iteratively update the attribute weights based on the current partition of the data. The relative magnitude of the attribute weights can be used to identify the subspaces in which clusters are embedded. Experiments on both synthetic data and real data show that the new algorithm outperforms the affinity propagation algorithm in recovering clusters from data.

1. Introduction

Data clustering is a fundamental tool for data analysis that aims to identify some inherent structure present in a set of objects [1,2]. Data clustering is also a key task in data mining and knowledge discovery, which focus on extracting non-trivial or hidden patterns from a set of objects [3,4]. In data clustering, we use clustering algorithms to automatically divide a set of objects or data points is into groups such that objects in the same group are similar to each other, while objects from different groups are distinct [5].

The *k*-means algorithm is a popular clustering algorithm that was developed about 60 years ago [6,7]. The number of clusters is a required input for the *k*-means algorithm. Given a dataset and a number *k* of clusters, the *k*-means algorithm starts by choosing *k* data points from the dataset as initial cluster centers and then repeats updating the cluster memberships and the cluster centers until some stop criterion is met [8,9]. It is highly possible that different initial cluster centers lead to different clustering results. Cluster center initialization may also affect the speed of convergence. As a result, several methods [10–18] have been developed to initialize cluster centers for the *k*-means algorithm.

A key challenge to most conventional clustering algorithms, including the *k*-means algorithm, is that they are not efficient to deal with high-dimensional data because clusters are embedded in subspaces of the high-dimensional data space and different clusters are associated with different subsets of the attributes. To address this problem, subspace clustering algorithms have been developed to identify clusters embedded in subspaces of the original data space. Examples of subspace clustering algorithms include [19–33], to just name a few. In particular, Favaro et al. [26] treated subspace clustering as a rank minimization problem and proposed an efficient way to solve the problem. Soltanolkotabi et al. [31] proposed a subspace clustering algorithm based Sparse Subspace Clustering [29] to cluster noisy data.

Soft or fuzzy subspace clustering algorithms are a type of subspace clustering algorithms that use weights to determine the importance of an attribute to a particular cluster. For example, the clustering algorithms proposed in [21–23,34,25,35] are soft subspace clustering algorithms. The subspace clustering algorithms proposed in [21,22] are similar to the *k*-means algorithm except that weights are used in the distance calculations. As a result, those subspace clustering algorithms also suffer from the cluster center initialization problem mentioned above.

In this paper, we develop a subspace clustering algorithm based on affinity propagation [36] to address the cluster center initialization problem. We call the new algorithm SAP (Subspace Affinity Propagation). The idea behind the SAP algorithm is to combine the power of attribute weighting and the affinity propagation method. Similar to the affinity propagation method, the SAP algorithm simultaneously considers all data points as initial cluster centers and thus does not have the cluster center initialization problem.

The remaining of this paper is organized as follows. Section 2 gives a brief review of the affinity propagation algorithm. Section 3 introduces the SAP algorithm. Section 4 presents numerical results based on synthetic data and real data to demonstrate the performance

* Corresponding author. Tel.: +1 860 486 3919; fax: +1 860 486 4238.
E-mail addresses: Guojun.Gan@gmail.com (G. Gan), mng@math.hkbu.edu.hk (M.-P. Ng).

http://dx.doi.org/10.1016/j.patcog.2014.11.003
0031-3203/

of the SAP algorithm. Section 5 concludes the paper and points out some areas for future research.

2. Affinity propagation

Affinity propagation is an efficient clustering method developed by Frey and Dueck [36]. This method starts with the similarity measures between pairs of data points and keeps passing real-valued messages between data points until a high-quality set of representative points (i.e., exemplars) and corresponding clusters are found. Unlike the k-means algorithm [8], which chooses an initial subset of data points as cluster centers, the affinity propagation method considers simultaneously all data points as cluster centers and thus is independent of the quality of the initial set of cluster centers.

In affinity propagation, two types of messages are exchanged between data points: responsibility and availability. The responsibility $r(i,k)$ is sent from data point i to candidate exemplar point k and reflects how well-suited it would be for point k to be the exemplar of point i. Here an exemplar [36,37] means a cluster center. The availability $a(i,k)$ is sent from data point candidate exemplar point k to data point i and reflects how appropriate it would be for data point i to choose candidate exemplar k as its exemplar.

Mathematically, the responsibility $r(i,k)$ and the availability $a(i,k)$ are updated as follows [36]:

$$r^{new}(i,k)=\lambda r^{old}(i,k)+(1-\lambda)\left(s(i,k)-\max_{j,j\neq k}\{a(i,j)+s(i,j)\}\right), \tag{1}$$

$$a^{new}(i,k)=\lambda a^{old}(i,k)+(1-\lambda)\left(\min\left\{0,r(k,k)+\sum_{j,j\notin\{i,k\}}\max\{0,r(j,k)\}\right\}\right),\quad i\neq k, \tag{2}$$

$$a^{new}(k,k)=\lambda a^{old}(k,k)+(1-\lambda)\left(\sum_{j,j\neq k}\max\{0,r(j,k)\}\right), \tag{3}$$

where λ is the damping factor between 0 and 1, and $s(i,j)$ is the similarity between points i and j for $i\neq j$. For example, $s(i,j)$ can be the negative squared Euclidean distance between points i and j, i.e.,

$$s(i,j)=-\sum_{l=1}^{d}(x_{il}-x_{jl})^2,$$

where x_{il} and x_{jl} are the lth attribute of point $\mathbf{x}_i$ and point $\mathbf{x}_j$, respectively. Note that $s(k,k)$ is an input value called "preference." The larger the value of $s(k,k)$, the more likely the point k is to be chosen as an exemplar.

The responsibilities and the availabilities are updated repeatedly until some stop criterion is met. For example, the iterative process can be terminated after a fixed number of iterations. At any step of the iterative process, responsibilities and availabilities can be combined to identify clusters and their members. For data point i, let k be the value that maximizes $a(i,k)+r(i,k)$, i.e.,

$$k=\arg\max_j\{a(i,j)+r(i,j)\}.$$

If $k=i$, then point i is an exemplar or cluster center. If $k\neq i$, then point k is an exemplar for point i.

Algorithm 1. The basic affinity propagation algorithm.

Require: Similarity matrix, preference, λ, *conviter*, *maxiter*
  numiter ← 1
  changes ← 0
  while true do
    Calculate responsibilities according to Eq. (1)
    Calculate availabilities according to Eqs. (2) and (3)
    if Exemplars changed **then**
      changes ← 0
    else
      changes ← *changes* + 1
    end if
    if *numiter* > = *maxiter* or *changes* > = *conviter* **then**
      break
    end if
    numiter ← *numiter* + 1
  end while
  Find the exemplars and form clusters by assigning every data point to its nearest exemplar

Algorithm 1 shows the pseudo-code of the affinity propagation algorithm. The affinity propagation algorithm requires several inputs: a similarity matrix, the preference value, λ, *conviter*, and *maxiter*. The preference value controls the number of clusters. Usually a higher preference value leads to more number of exemplars. The parameter λ is the damping factor that controls the robustness of the iterative process. A default value for λ is 0.9 as suggested by [36]. The parameters *conviter* and *maxiter* control when the iterative process will be terminated. In particular, the iterative process terminates if the exemplars do not change for *conviter* consecutive iterations. The iterative process also terminates if the number of iterations reaches *maxiter*. The default values for *conviter* and *maxiter* are 10 and 1000, respectively.

Since the publication of the affinity propagation algorithm in 2007, many improvements to the algorithm have been proposed. For example, Yu et al. [38] introduced a space vector model to calculate similarities.

3. Subspace affinity propagation

Antony [39] utilized affinity propagation to improve the Density Conscious Subspace clustering algorithm (DENCOS) [40]. In the approach proposed in [39], affinity propagation is used to detect the local densities for a dataset in order to select a small number of final representative exemplars, which will be partitioned by the DENCOS algorithm.

In this paper, we use affinity propagation to find subspace clusters in a different way by utilizing variable weights [21] or fuzzy subspace clustering [22,34]. The SAP (Subspace Affinity Propagation) algorithm is similar to the original affinity propagation algorithm except that we add a component to determine the importance of each dimension or attribute to the exemplar.

Suppose that the underlying dataset consists of n data points: $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n$, each of which is described by d numerical attributes. For each point i, we introduce a vector of weights, $\mathbf{w}_i=(w_{i1},w_{i2},\ldots,w_{id})^T$, such that

$$\sum_{l=1}^{d}w_{il}=1 \tag{4}$$

and

$$w_{il}\geq 0,\quad l=1,2,\ldots,d.$$

For $i\neq k$, the similarity between points i and k with the attribute weights is calculated as

$$s(i,k)=-\sum_{l=1}^{d}w_{kl}^{\alpha}(x_{il}-x_{kl})^2, \tag{5}$$

where $\alpha>1$ is a constant, and x_{il} and x_{kl} are the lth component of points $\mathbf{x}_i$ and $\mathbf{x}_k$, respectively. Note that the similarity is not symmetric because we use the weights associated with point k

Please cite this article as: G. Gan, M.K.-P. Ng, Subspace clustering using affinity propagation, Pattern Recognition (2014), http://dx.doi.org/10.1016/j.patcog.2014.11.003

to calculate the similarity between points i and k. The reason for calculating similarity in this way is that the responsibility $r(i,k)$ (see Eq. (1)) that reflects how well-suited for point k to serve as an exemplar for point i is dependent on $s(i,k)$.

At the beginning of the SAP algorithm, we initialize all the weights to be equal, i.e., $w_{il}=1/d$ for $i=1,2,\ldots,n$ and $l=1,2,\ldots,d$. Once responsibilities and availabilities are updated according to Eqs. (1), (2), and (3), we will find all the exemplar points and update the weights of every exemplar point and then update the attribute weights as follows:

$$w_{kl}=\frac{1}{\sum_{h=1}^{d}\left(\frac{V_{kl}+\epsilon}{V_{kh}+\epsilon}\right)^{1/(\alpha-1)}},\quad k=1,2,\ldots,n,\ l=1,2,\ldots,d, \tag{6}$$

where $\alpha>1$ is a constant and ϵ is a small positive constant used to prevent dividing by zero. The derivation of Eq. (6) can be found in the appendix of this paper.

As we can see from the above description of the SAP algorithm, the only difference between the SAP algorithm and the original affinity propagation method is the attribute weight update. During the iterative process, we update the attribute weights for all exemplars identified at that time and recalculate the similarities between every exemplar and other points. We do not need to recalculate the similarities between non-exemplars and other points because the weights of non-exemplars are not updated. We can use the similarities between non-exemplars and other points calculated before. This can save the computation time significantly if there are only a few exemplars.

To save the computation time further and make the iterative process more stable, we can update the attribute weights and the similarities between exemplars and other points less frequently. That is, we do not need to update the attribute weights and recalculate the similarities at every iterative step. We can do so every *freq* iterative steps.

Algorithm 2. The SAP algorithm.

Require: Dataset, preference, λ, *conviter*, *maxiter*, *freq*, α, ϵ
  numiter $\leftarrow 1$
  changes $\leftarrow 0$
  Calculate the similarity matrix based on equal weights
  while true do
    Calculate responsibilities according to Eq. (1)
    Calculate availabilities according to Eqs. (2) and (3)
  if *numiter* mod *freq* $=0$ **then**
    Update attribute weights according to Eq. (6)
    Update the similarities between every exemplar and other data points
  end if
  if Exemplars changed **then**
    changes $\leftarrow 0$
  else
    changes $\leftarrow$ *changes* $+1$
  end if
  if *numiter* $>=$ *maxiter* or *changes* $>=$ *conviter* **then**
    break
  end if
  numiter $\leftarrow$ *numiter* $+1$
end while
Find the exemplars and their corresponding attribute weights
Form clusters by assigning every data point to its nearest exemplar

The pseudo-code of the SAP algorithm is shown in Algorithm 2. The SAP algorithm requires several parameters. Since the SAP algorithm will calculate the similarities with attribute weights, we need to input a dataset to the algorithm. The preference parameter and the damping factor are inherited from the original affinity propagation method. These two parameters control the number of clusters and the stability of the iterative process, respectively. The parameters *conviter* and *maxiter* tell when to terminate the iterative process. The parameter *freq* controls the frequency of weight and similarity update. The other two parameters α and ϵ are used in distance calculation. The default values for these parameters are given in Table 1.

As we can see from Eqs. (5) and (6), the parameter α affects the similarity and the attribute weight. If we choose a larger α, the similarity will be smaller and the attribute weights will be more homogeneous. Suppose that the number of attribute is 10, for example. Changing α from 2 to 3 will change the similarities to one 10th of their values. As a result, α will impact the choice of a preference value.

There is a tradeoff to choose a value for the attribute weight update frequency. Since it takes several iterations for the affinity propagation process to produce stable partitions, updating the attribute weight too frequently makes the iterative process unstable. Also updating the attribute weight too frequently increases the runtime. On the other hand, updating the attribute weight infrequently will not capture the subspace information. In practice, we found that updating the attribute weight every 10 iterations works well. Note that when we set *freq* to a number that is larger than *maxiter*, the attribute weight will not get updated during the iterative process. In such cases, the SAP algorithm becomes the ordinary affinity propagation algorithm we introduced before.

Choosing a preference value for the SAP algorithm is similar to choosing a preference value for the original affinity propagation algorithm. That is, we use the median of the similarities as a starting point. However, we need to consider the subspace dimensions when calculating the median of the similarities. At the beginning of the SAP algorithm, all the weights are set to $1/d$ and the similarity between two points is calculated as

$$s(i,k)=-\frac{1}{d^{\alpha}}\sum_{l=1}^{d}(x_{il}-x_{kl})^2.$$

Suppose that the average subspace dimensions of the clusters is d'. Then d' weights will be significantly larger than the remaining $d-d'$ weights, which are close to zero. The similarity between two points will approximately be

$$s(i,k)=-\frac{1}{(d')^{\alpha}}\sum_{j=1}^{d'}(x_{il_j}-x_{kl_j})^2.$$

Since we do not know the d' subspace dimensions at the beginning, we can use the average of the full distance to estimate the similarity between two points as

$$s(i,k)=-\frac{1}{(d')^{\alpha}}\frac{d'}{d}\sum_{l=1}^{d}(x_{il}-x_{kl})^2. \tag{7}$$

For the SAP algorithm, we can choose the median of the $n(n-1)/2$ distinct similarities calculated by Eq. (7) as a starting point.

Table 1
Default values for some parameters required by the SAP algorithm.

Parameter	Default value	Parameter	Default value
conviter	10	α	2
maxiter	1000	ϵ	10^{-6}
freq	10		

Please cite this article as: G. Gan, M.K.-P. Ng, Subspace clustering using affinity propagation, Pattern Recognition (2014), http://dx.doi.org/10.1016/j.patcog.2014.11.003

4. Experiments

In this section, we present experimental results to demonstrate the performance of the SAP algorithm in terms of discovering subspace clusters and identifying the significant attributes associated with them. We use both synthetic data and real data in these experiments.

4.1. Evaluation method

Since the labels of the synthetic data and the real data are known, we use the corrected Rand index [41] to evaluate the performance of the clustering algorithms. The corrected Rand index takes values between -1 and 1. A corrected Rand index of 1 indicates a perfect agreement between the known partition and the found partition; while a negative corrected Rand index indicates agreement by chance.

To define the corrected Rand index, we let $\mathcal{U}=\{U_1,U_2,\ldots,U_{k_1}\}$ be the known partition and let $\mathcal{V}=\{V_1,V_2,\ldots,V_{k_2}\}$ be the partition found by a clustering algorithm. Then the corrected Rand index is calculated as

$$R=\frac{\binom{n}{2}\sum_{i=1}^{k_1}\sum_{j=1}^{k_2}\binom{n_{ij}}{2}-\sum_{i=1}^{k_1}\binom{n_{i\cdot}}{2}\sum_{j=1}^{k_2}\binom{n_{\cdot j}}{2}}{\frac{1}{2}\binom{n}{2}\left[\sum_{i=1}^{k_1}\binom{n_{i\cdot}}{2}+\sum_{j=1}^{k_2}\binom{n_{\cdot j}}{2}\right]-\sum_{i=1}^{k_1}\binom{n_{i\cdot}}{2}\sum_{j=1}^{k_2}\binom{n_{\cdot j}}{2}}, \tag{8}$$

where $n_{ij}=|U_i\cap V_j|$, $n_{i\cdot}=|U_i|$, $n_{\cdot j}=|V_j|$, and n is the total number of data points.

4.2. Experiments on synthetic data

Our purpose of developing the SAP algorithm is twofold: first, we want to apply the affinity propagation method to high-

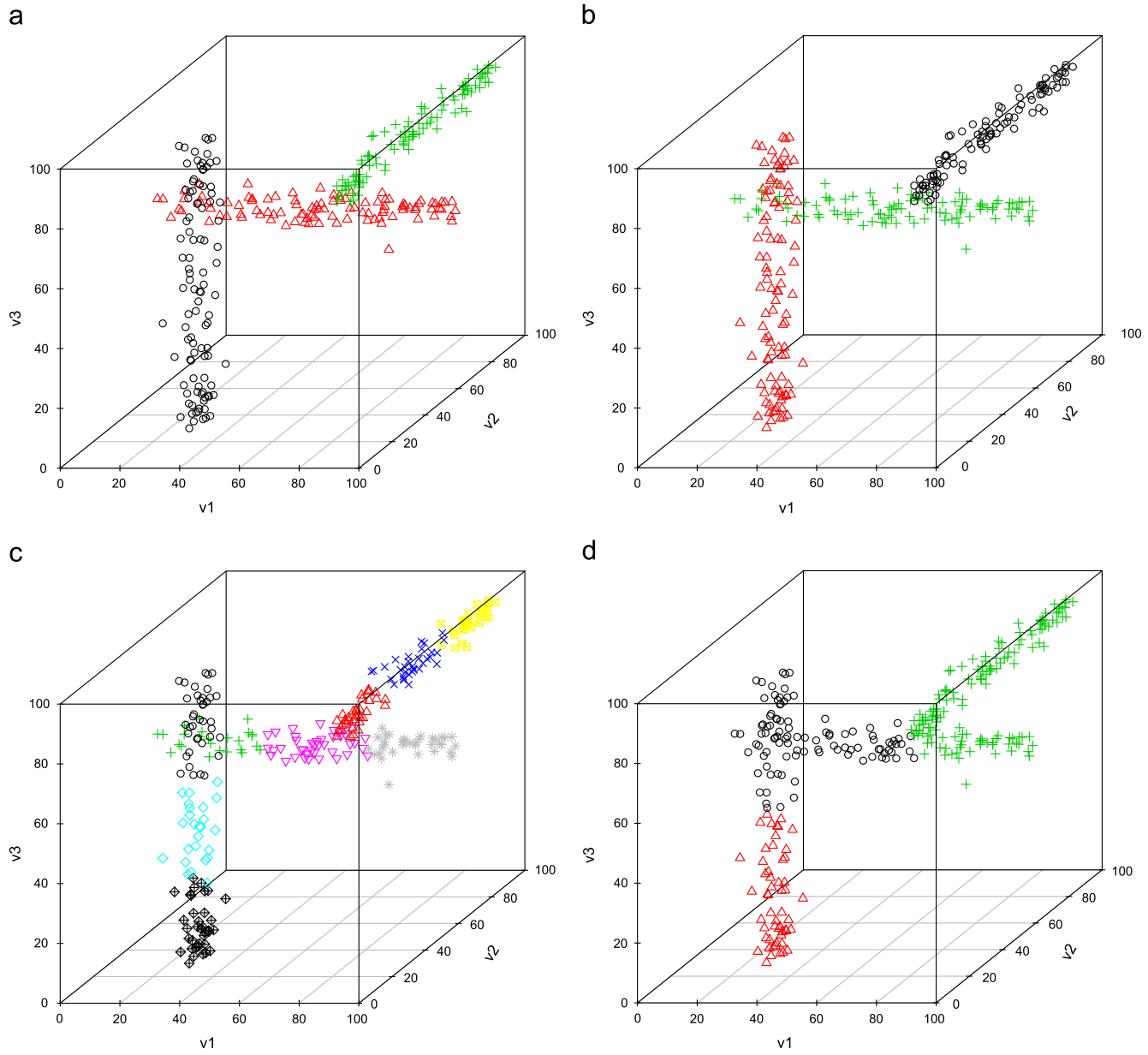

Fig. 1. A 3-dimensional data set and its partitions. (a) The 3-dimensional dataset. (b) Partition obtained by SAP with preference -500, *freq*$=10$. (c) Partition obtained by SAP with preference -500, *freq*$=1001$. (d) Partition obtained by SAP with preference -8000, *freq*$=1001$.

Please cite this article as: G. Gan, M.K.-P. Ng, Subspace clustering using affinity propagation, Pattern Recognition (2014), http://dx.doi.org/10.1016/j.patcog.2014.11.003

dimensional data; second, we want to address the problem of cluster center initialization suffered by some subspace clustering algorithms such as the FSC algorithm [34]. In this subsection, we shall use synthetic data to show that the SAP algorithm can perform as expected.

4.2.1. Synthetic data generation

We follow the method introduced in [42] to generate synthetic data with subspace clusters. Here we give a brief description of the data generation method we used to create our synthetic data.

Let d be the intended dimension and k be the intended number clusters. Let n_i be the intended size of cluster i for $i=1,2,\ldots,k$. Let $S_i \in \{1,2,\ldots,k\}$ be the subset of attributes associated with cluster i for $i=1,2,\ldots,k$. The data generation algorithm consists of two steps. First, the algorithm generates k anchor points or cluster centers. To simplify the data generation process, we generate the cluster centers such that all the coordinates of a center have the same value. In other words, we generate the cluster centers $\mathbf{z}_i$ as follows:

$$z_{ij}=\frac{90i}{k}, \quad i=1,2,\ldots,k,\ j=1,2,\ldots,d.$$

Second, the algorithm generates the required data points for all clusters. For a given $i=1,2,\ldots,k$, the algorithm generate a data point $\mathbf{x}$ for cluster i as follows. If $j \in S_i$, then the jth attribute value is generated from a normal distribution with standard deviation of $rU[1,s]$, where $U[1,s]$ denotes a uniform random number generated from $[1,s]$. Here s and r are fixed separate parameters. If $j \notin S_i$, the jth attribute value is generated as $U[0,100]$. Mathematically, the jth attribute value is generated as

$$x_j=\begin{cases} z_{ij}+N(0,(rU[1,s])^2) & \text{if } j \in S_i \\ U[0,100] & \text{if } j \notin S_i, \end{cases}$$

where $N(0,(rU[1,s])^2)$ denotes a random number generated from a normal distribution with mean 0 and standard deviation $rU[1,s]$. We used $r=s=2$ to generate our synthetic datasets.

4.2.2. Results

We used the method described above to generate two datasets. The first dataset contains 300 points, each of which is described by 3 attributes. The dataset is shown in Fig. 1(a), from which we see that this dataset have 3 subspace clusters. Each cluster is embedded in a 2-dimensional plane.

Fig. 2 shows the number of clusters obtained by applying the SAP algorithm to this dataset with different preference values. If we set the preference value to 10 percentile of the similarities calculated from Eq. (7) with $d'=2$, we obtained 9 clusters. In other cases, the SAP algorithm produced 3 clusters. In general, a higher preference value leads to more number of clusters.

We apply the SAP algorithm to the 3-dimensional dataset with different preference and weight update frequency. The results are shown in Fig. 1. In particular, Fig. 1(b) shows the 3 clusters found by the SAP algorithm with a preference value of -500 and a frequency value of 10. If we set the frequency of updating weights and similarities to 1001, which is larger than the maximum number of iterations, then the SAP algorithm becomes the ordinary affinity propagation algorithm. Fig. 1(c) and (d) shows that the ordinary affinity propagation algorithm is not able to recover correctly the clusters embedded in subspaces.

Table 2 shows the number of iterations, the run-time, and the accuracy of the SAP algorithm under various settings. From the table we see that the SAP algorithm converges in less number of iterations when attribute weights and similarities are updated during the iterative process.

Table 3 shows the attribute weights of the 3 clusters found by the SAP algorithm when the preference and frequency were set to -500 and 10, respectively. The relative magnitude of weights in the table tells us the subspace information of each cluster. For example, the weight of the first attribute for Cluster 3 is 0.0041, which is much smaller than the other two attribute weights. This information tells us that Cluster 3 is embedded in the subspace formed by the second and third attributes.

We also applied the FSC algorithm [34] to cluster this 3-dimensional dataset into 3 clusters 1000 times with random cluster center initialization. Fig. 3 shows the corrected Rand indices of the 1000 runs of the FSC algorithm. From the figure we see that the FSC algorithm failed to recover the 3 clusters in two out of the 1000 runs.

The second synthetic dataset contains 2000 100-dimensional points, which are belong to four subspace clusters. Table 4 shows sizes of the clusters and the subspace dimensions associated with these clusters. The clusters are associated with subspaces of different dimensions.

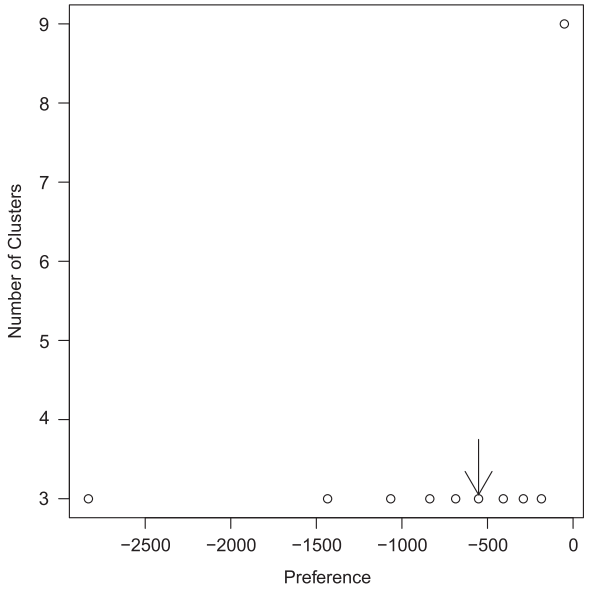

Fig. 2. The number of clusters obtained by applying the SAP algorithm to the 3-dimensional dataset with 10, 20, …, 100 percentiles of the similarities calculated from Eq. (7) with $d'=2$. The point pointed by the arrow is the number of clusters with the preference value set to the median.

Table 2
Speed and accuracy of the SAP algorithm applied to the 3-dimensional dataset with different preference and weight update frequency.

Preference	*freq*	Number of iterations	Run-time (s)	Corrected Rand index
−500	10	46	1.514	1
−500	1001	46	1.494	0.4022
−8000	1001	77	2.126	0.4144

Table 3
Attribute weights associated with 3 clusters of the 3-dimensional dataset.

Cluster	v1 weight	v2 weight	v3 weight
Cluster 1	0.4206	0.0019	0.5775
Cluster 2	0.3774	0.6199	0.0027
Cluster 3	0.0041	0.7260	0.2699

Please cite this article as: G. Gan, M.K.-P. Ng, Subspace clustering using affinity propagation, Pattern Recognition (2014), http://dx.doi.org/10.1016/j.patcog.2014.11.003

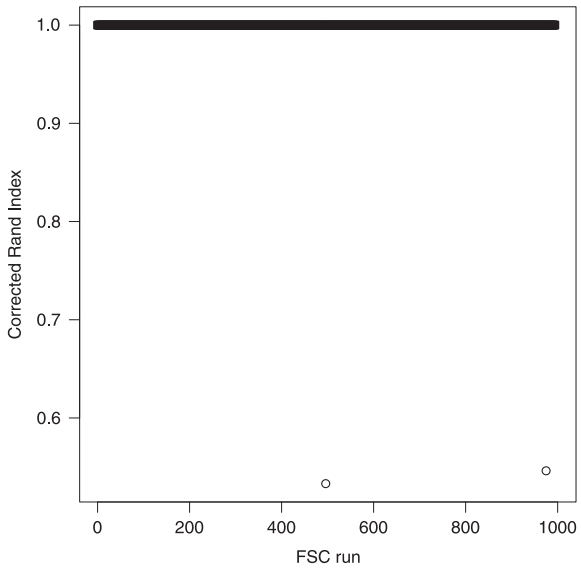

Fig. 3. The corrected Rand indices produced by running the FSC algorithm 1000 times on the 3-dimensional dataset.

Table 4
A 100-dimensional dataset with 4 subspace clusters.

Cluster	Number of points	Subspace dimensions
A	500	10,15,70
B	300	20,30,80,85
C	500	30,40,70,90,95
D	700	40,45,50,55,60,80

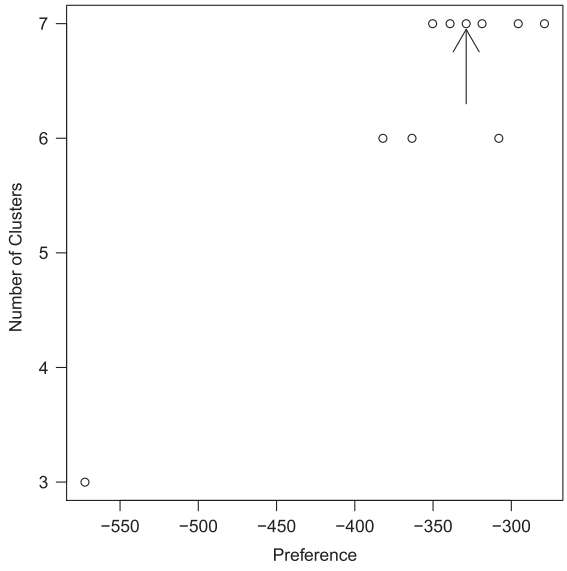

Fig. 4. The number of clusters obtained by applying the SAP algorithm to the 100-dimensional dataset with 10, 20, …, 100 percentiles of the similarities calculated from Eq. (7) with $d' = 5$. The point pointed by the arrow is the number of clusters with the preference value set to the median.

Fig. 4 shows the number of clusters obtained by applying the SAP algorithm to the 100-dimensional dataset with different preference values. If we set the preference value to 10 percentile of the similarities calculated from Eq. (7) with $d' = 5$, we obtained 7 clusters. In other cases, the SAP algorithm produced 3 or 6 clusters. Unlike the similarities of the 3-dimensional dataset, the similarities of the 100-dimensional dataset are uniform and have a smaller range.

Table 5
The confusion matrix produced by applying the SAP algorithm to the 100-dimensional dataset with preference −500 and *freq*10.

	A	B	C	D
1	0	0	500	0
2	0	300	0	0
3	500	0	0	1
4	0	0	0	699

We applied the SAP algorithm to cluster the 100-dimensional data with a preference of −500 and a weight update frequency of 10. The confusion matrix of this test case is shown in Table 5. From the table we see that only one point was clustered incorrectly.

Fig. 5 shows the attribute weights associated with the four clusters identified by the SAP algorithm with a preference value of −500 and a *freq* of 10. The relative magnitudes of the attribute weights show the subspace dimensions associated with these clusters.

To test the original affinity propagation method on this 100-dimensional dataset, we applied the SAP algorithm with a weight update frequency of 1001, which is larger than the maximum number of iterations. Table 6(a) and (b) shows the confusion matrices of the clusters identified by the SAP algorithm without attribute weight update. From the confusion matrices we see that the data points are almost uniformly distributed in the clusters.

Table 7 shows the number of iterations, the number of iterations, and the accuracy of the SAP algorithm under several settings of preference and weight update frequency. With attribute weight update every 10 iterations, the SAP algorithm terminated in 92 iterations and finished in about 5.5 min. Without attribute weight update, the SAP algorithm becomes the ordinary affinity propagation algorithm. The run-times show that the SAP algorithm with weight update is not faster than the ordinary affinity propagation algorithm because the attribute weight update is slow for data in high dimensional spaces.

To test the impact of cluster center initialization on the FSC algorithm, we applied the FSC algorithm to cluster the 100-dimensional dataset 100 times with random cluster center initialization. Fig. 6 shows the corrected Rand indices of the 100 runs. From the figure we see that in 12 out of the 100 runs, the partition produced by the FSC algorithm is not close to the true partition.

4.3. Experiments on real data

To test the SAP algorithm on real data, we obtain two gene expression datasets from [41][1]: the gene expression data from human liver cancers and the gene expression data from breast tumors and colon tumors. Both datasets have known labels. Table 8 shows the information of the two real datasets. Since different attributes of the real datasets have different ranges we use the *z*-score method to normalize all the attributes to prevent an individual attribute from dominating the distance.

We applied the SAP algorithm to the two real datasets with different preferences and attribute weight update frequency. Given a weight update frequency, we tuned the preference value so that the SAP algorithm produces two clusters. Since the maximum number of iterations is 1000, a weight update frequency of 1001

[1] The two datasets are available at http://bioinformatics.rutgers.edu/Static/Supplements/CompCancer/datasets.html.

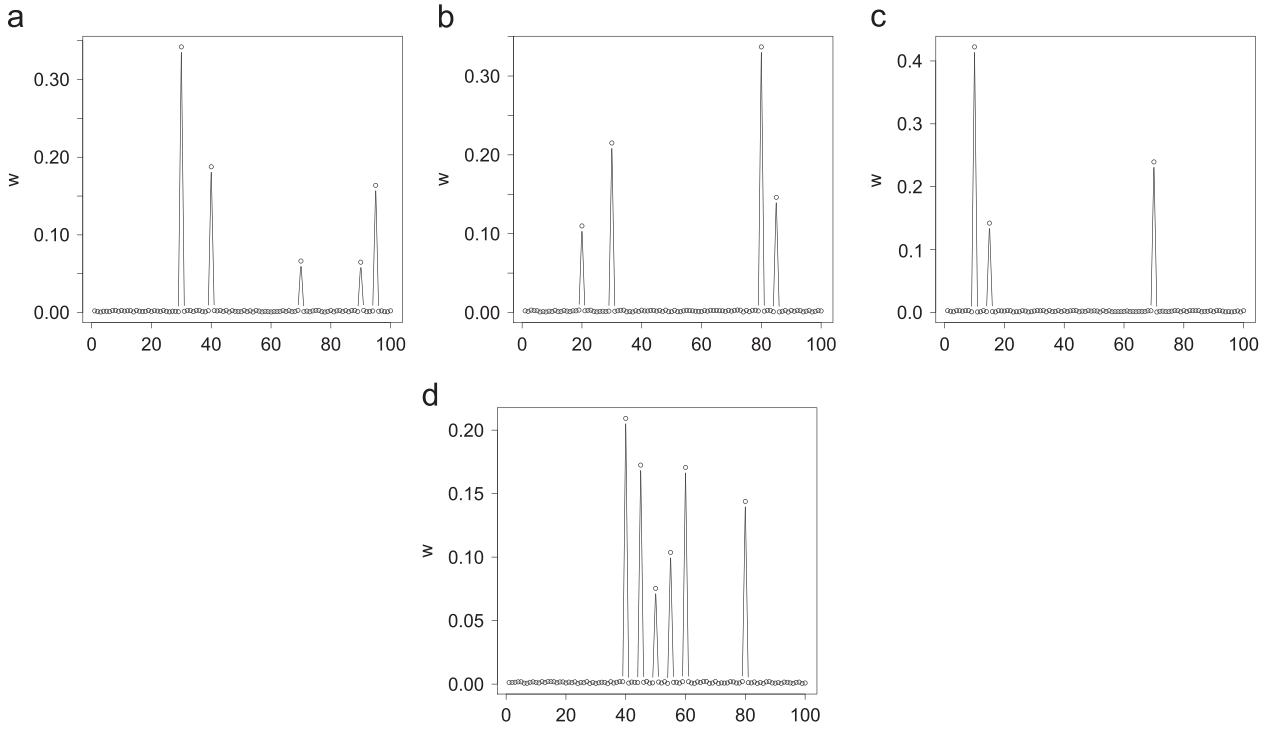

Fig. 5. Attribute weights of the clusters produced by the SAP algorithm on the 100-dimensional dataset.

Table 6
The confusion matrix produced by applying the SAP algorithm to the 100-dimensional dataset when *freq* was set to 1001.

	A	B	C	D
1	158	99	176	181
2	158	109	170	337
3	184	92	154	182

(a) Preference value: -500

	A	B	C	D
1	147	85	141	94
2	152	95	141	202
3	24	41	78	283
4	177	79	140	121

(b) Preference value: -400

Table 7
Speed of the SAP algorithm applied to the 100-dimensional dataset with different preference and weight update frequency.

Preference	*freq*	Number of iterations	Run-time (s)	Corrected Rand index
−500	10	92	332.666	0.99848
−500	1001	83	262.947	0.0133
−400	1001	121	344.854	0.0459

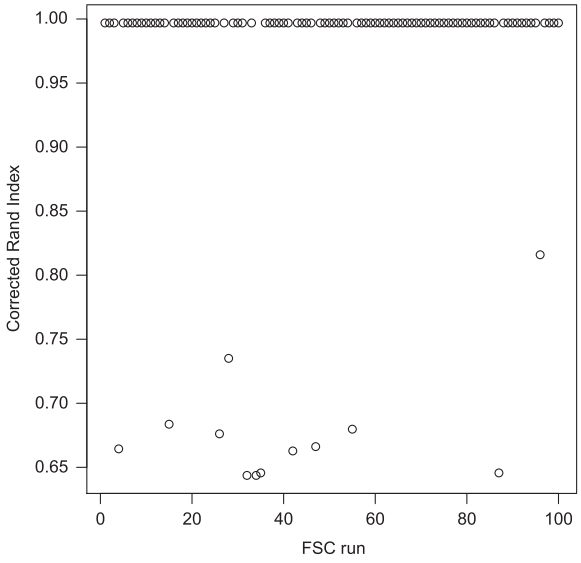

Fig. 6. The corrected Rand indices produced by running the FSC algorithm 100 times on the 100-dimensional dataset.

means that there are no attribute weight updates during the iterative process. Table 9 shows the accuracy of the SAP algorithm under four runs. Since the SAP algorithm without attribute weight update becomes the ordinary affinity propagation algorithm, the corrected Rand indices in Table 9 show that the SAP algorithm with weight update produces more accurate results.

Table 10 shows the confusion matrices corresponding to the four runs in Table 9. Table 10(a) and (b) shows the confusion matrices produced by the SAP algorithm on the human liver cancer dataset with attribute weight update and without attribute weight update, respectively. From the two tables we can see that the ordinary affinity propagation algorithm produces one large cluster and one small cluster. The SAP algorithm with attribute weight update produces two clusters that are similar to the known clusters. For the human liver cancer dataset, the first six significant attributes or genes associated with the first cluster are 3548, 7826, 18395, 6284, 4055, and 21 404; whereas the first six significant

Table 8
Two real gene expression datasets, each of which has two known clusters.

Dataset	Samples	Attributes	Cluster sizes
Human liver cancer	179	85	104,76
Breast and colon tumor	104	182	62,42

Table 9
Accuracy of the SAP algorithm applied to the real datasets with different preference and weight update frequency. Other parameters not shown in the table use default values from Table 1.

Dataset	Preference	*freq*	Corrected Rand index
Human liver cancer	−0.5	10	0.5664
Human liver cancer	−0.3	1001	−0.0043
Breast and colon tumor	−0.05	10	0.8151
Breast and colon tumor	−0.11	1001	0.0296

Table 10
The confusion matrices produced by applying the SAP algorithm to the real datasets. (a) Human liver cancer with preference −0.5 and *freq*10. (b) Human liver cancer with preference −0.3 and *freq*001. (c) Breast and colon tumor with preference −0.05 and *freq*10. (d) Breast and colon tumor with preference −0.11 and *freq*1001. The symbols "HCC" and "Liver" denote the known class labels of the human liver cancer dataset. The symbols "B" and "C" denote the known class labels of the breast and colon tumor dataset.

	HCC	Liver
1	18	71
2	86	4

(a)

	HCC	Liver
1	12	8
2	92	67

(b)

	B	C
1	58	1
2	4	41

(c)

	B	C
1	0	3
2	62	39

(d)

genes associated with the second cluster are 10646, 16 745, 5613, 16 685, 18 961, and 21 999.

Table 10(c) and (d) also shows that the SAP algorithm with attribute weight update produces more accurate results than the ordinary affinity propagation algorithm. For the breast and colon tumor dataset, the first cluster is associated with nine important genes: 205043_at, 205506_at, 206000_at, 206286_s_at, 206312_at, 206418_at, 206430_at, 207814_at, and 214142_at. The second cluster has five significant genes: 206799_at, 209351_at, 206378_at, 209602_s_at, and 205509_at.

For comparison, we also applied the LRR (Low Rank Representation) based method [43] and the GCEPD (Graph based Clustering technique with Embedded Pattern Detection) [44] to the two real datasets. Table 11 shows the results produced by the LRR based method with two clusters and different noise levels. From the table we see that the LRR based method tends to cluster all genes into one cluster with a high noise level and cluster genes evenly with a low noise level. Table 12 shows the results produced by the GCEPD algorithm with a matching factor of 0.5. Since the GCEPD algorithm does not take the number of clusters as input, it produced 23 clusters for the human liver dataset and 14 clusters for the breast and colon tumor dataset, respectively. The results show that the GCEPD algorithm tends to produce many small clusters.

5. Concluding remarks

In this paper, we proposed a subspace clustering algorithm, called the SAP algorithm, based on affinity propagation developed by [36]. Unlike the *k*-means algorithm, the affinity propagation algorithm uses exemplars, which are data points from the underlying dataset, to represent cluster centers. By introducing attribute weights in the distance calculation, we extended the original affinity propagation algorithm for identifying clusters embedded in subspaces of the data space. The relative magnitude of the attribute weights can be used to identify dimensions of the subspace clusters.

The experiments on both synthetic data and real data have shown that the SAP algorithm outperformed the original affinity propagation algorithm in recovering clusters from data. The experiments on synthetic data have shown that the SAP algorithm is effective in recovering clusters embedded in subspaces and identifying the important attributes. The experiments on real gene expression data have shown that the SAP algorithm produces more accurate clustering results than the original affinity propagation algorithm.

One drawback of the SAP algorithm is that the similarity matrix of the dataset has to be recalculated during the iterative process because the changes of attribute weights affect the similarities between data points. In the original affinity propagation algorithm, the similarity matrix or part of it needs to be calculated once. In future, we would like to investigate ways to improve the SAP algorithm to handle large datasets.

Table 11
The confusion matrices produced by applying the LRR based algorithm to the real datasets. (a), (b), (c) Human liver cancer with noise parameter λ=0.01, 0.05, and 0.1, respectively. (d), (e), (f) Breast and Colon tumor with noise parameter λ=0.01, 0.05, and 0.1, respectively.

	HCC	Liver
1	69	48
2	35	27

(a)

	HCC	Liver
1	12	8
2	92	67

(b)

	HCC	Liver
1	103	75
2	1	0

(c)

	B	C
1	35	7
2	27	35

(d)

	B	C
1	0	1
2	62	41

(e)

	B	C
1	62	41
2	0	1

(f)

Table 12
The confusion matrices produced by applying the GCEPD algorithm to the real datasets. (a) Human liver cancer with a matching factor 0.5. (b) Breast and colon tumor with a matching factor of 0.5.

	HCC	Liver		HCC	Liver
1	19	0	13	1	2
2	17	0	14	0	1
3	17	0	15	9	0
4	1	0	16	1	0
5	1	0	17	2	0
6	13	8	18	2	4
7	8	9	19	2	0
8	1	0	20	1	0
9	0	26	21	1	0
10	0	5	22	1	0
11	0	19	23	0	1
12	7	0			

(a)

	B	C		B	C
1	29	0	8	0	2
2	5	0	9	4	1
3	8	0	10	3	0
4	1	30	11	0	1
5	0	6	12	0	1
6	7	0	13	2	0
7	3	0	14	0	1

(b)

Conflict of interest

None.

Acknowledgements

The authors would like to thank two referees for their insightful comments that greatly improve the organization and quality of the paper.

Appendix A. Attribute weight determination

To update the weight for an exemplar point k, we minimize

$$\sum_{\mathbf{y}\in C_k}\sum_{l=1}^{d} w_{kl}^{\alpha}(y_l-x_{kl})^2+\epsilon\sum_{l=1}^{d} w_{kl}^{\alpha} \tag{A.1}$$

subject to the constraint given in Eq. (4), where C_k is the set of points that choose point k as their exemplar, $\alpha>1$ is a constant, and $\epsilon>0$ is a small constant. This optimization problem is a typical quadratic programming problem and has an analytic solution. To get the weights, we consider

$$\sum_{\mathbf{y}\in C_k}\sum_{l=1}^{d} w_{kl}^{\alpha}(y_l-x_{kl})^2+\epsilon\sum_{l=1}^{d} w_{kl}^{\alpha}-\beta\left(\sum_{l=1}^{d} w_{il}-1\right)$$

and take derivatives of this equation with respect to w_{k1}, w_{k2}, …, w_{kd}, and β, where β is the Lagrange multiplier. Equating those derivatives to zero leads to a linear equation system with $d+1$ unknowns:

$$\alpha w_{kl}^{\alpha-1}(V_{kl}+\epsilon)=\beta,\quad l=1,2,\ldots,d,$$

$$\sum_{l=1}^{d} w_{il}=1,$$

where

$$V_{kl}=\sum_{\mathbf{x}\in C_k}(x_l-x_{kl})^2. \tag{A.2}$$

Solving the above linear equation system gives

$$w_{kl}=\frac{1}{\sum_{h=1}^{d}\left(\frac{V_{kl}+\epsilon}{V_{kh}+\epsilon}\right)^{1/(\alpha-1)}},\quad k=1,2,\ldots,n,\ l=1,2,\ldots,d, \tag{A.3}$$

where V_{kl} is defined in Eq. (A.2). The small positive ϵ can prevent dividing by zero in the weight calculation when all points in C_k have the same value in the lth attribute.

References

[1] K.-L. Du, M. Swamy, Clustering I: basic clustering models and algorithms, in: Neural Networks and Statistical Learning, Springer London, 2014, pp. 215–258.
[2] K.-L. Du, M. Swamy, Clustering II: topics in clustering, in: Neural Networks and Statistical Learning, Springer London, 2014, pp. 259–297.
[3] B. Mirkin, Clustering for Data Mining: A Data Recovery Approach, Chapman and Hall/CRC, Boca Raton, FL, 2005.
[4] P. Berkhin, A survey of clustering data mining techniques, in: J. Kogan, C. Nicholas, M. Teboulle (Eds.), Grouping Multidimensional Data, Springer, Berlin Heidelberg, 2006, pp. 25–71.
[5] G. Gan, C. Ma, J. Wu, Data Clustering: Theory, Algorithms, and Applications, of ASA-SIAM Series on Statistics and Applied Probability, Vol. 20, SIAM Press, Philadelphia, PA, 2007.
[6] A. Jain, Data clustering: 50 years beyond k-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[7] J. Wu, Advances in K-means Clustering: A Data Mining Thinking, Springer, New York, NY, 2012.
[8] J. Macqueen, Some methods for classification and analysis of multivariate observations, in: L. Cam, J. Neyman (Eds.), Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics andProbability, vol. 1, University of California Press, Berkely, CA, USA, 1967, pp. 281–297.
[9] J.A. Hartigan, M.A. Wong, Algorithm as 136: a k-means clustering algorithm, J. R. Stat. Soc. Ser. C (Appl. Stat.) 28 (1) (1979) 100–108.
[10] J. Peña, J. Lozano, P. Larrañaga, An empirical comparison of four initialization methods for the k-means algorithm, Pattern Recognit. Lett. 20 (10) (1999) 1027–1040.
[11] S.S. Khan, A. Ahmad, Cluster center initialization algorithm for k-means clustering, Pattern Recognit. Lett. 25 (11) (2004) 1293–1302.
[12] D. Arthur, S. Vassilvitskii, k-means++: the advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM Press, Philadelphia, PA, USA, 2007, pp. 1027–1035.
[13] S.J. Redmond, C. Heneghan, A method for initialising the k-means clustering algorithm using kd-trees, Pattern Recognit. Lett. 28 (8) (2007) 965–973.
[14] J. Lu, J. Tang, Z. Tang, J. Yang, Hierarchical initialization approach for k-means clustering, Pattern Recognit. Lett. 29 (6) (2008) 787–795.
[15] F. Cao, J. Liang, G. Jiang, An initialization method for the k-means algorithm using neighborhood model, Comput. Math. Appl. 58 (3) (2009) 474–483.
[16] M. Erisoglu, N. Calis, S. Sakallioglu, A new algorithm for initial cluster centers in k-means algorithm, Pattern Recognit. Lett. 32 (14) (2011) 1701–1705.
[17] A.M. Bagirov, J. Ugon, D. Webb, Fast modified global k-means algorithm for incremental cluster construction, Pattern Recognit. 44 (4) (2011) 866–876.
[18] R.C. de Amorim, B. Mirkin, Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering, Pattern Recognit. 45 (3) (2012) 1061–1075.
[19] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: SIGMOD Record ACM Special Interest Group on Management of Data, ACM Press, New York, NY, USA, 1998, pp. 94–105.
[20] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: a review, SIGKDD, Newslett. ACM Spec. Interest Group Knowl. Discov. Data Mining 6 (1) (2004) 90–105.
[21] J. Huang, M. Ng, H. Rong, Z. Li, Automated variable weighting in k-means type clustering, IEEE Trans. Pattern Anal. Mach. Intell. 27 (5) (2005) 657–668.
[22] G. Gan, J. Wu, Z. Yang, A fuzzy subspace algorithm for clustering high dimensional data, in: X. Li, S. Wang, Z. Dong (Eds.), Lecture Notes in Artificial Intelligence, vol. 4093, Springer-Verlag, 2006, pp. 271–278.
[23] L. Jing, M. Ng, J. Huang, An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data, IEEE Trans. Knowl. Data Eng. 19 (8) (2007) 1026–1041.
[24] H.-P. Kriegel, P. Kröger, A. Zimek, Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering, ACM Trans. Knowl. Discov. Data 3 (1) (2009) 1:1–1:58.
[25] Z. Deng, K.-S. Choi, F.-L. Chung, S. Wang, Enhanced soft subspace clustering integrating within-cluster and between-cluster information, Pattern Recognit. 43 (3) (2010) 767–781.
[26] P. Favaro, R. Vidal, A. Ravichandran, A closed form solution to robust subspace estimation and clustering, in: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 1801–1807.
[27] E. Müller, I. Assent, S. Günnemann, T. Seidl, Scalable density-based subspace clustering, in: Proceedings of the Twentieth ACM International Conference on Information and Knowledge Management, 2011, pp. 1077–1086.
[28] X. Chen, Y. Ye, X. Xu, J.Z. Huang, A feature group weighting method for subspace clustering of high-dimensional data, Pattern Recognit. 45 (1) (2012) 434–446.
[29] E. Elhamifar, R. Vidal, Sparse subspace clustering: algorithm, theory, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2765–2781.
[30] M.E. Timmerman, E. Ceulemans, K.D. Roover, K.V. Leeuwen, Subspace k-means clustering, Behav. Res. Methods 45 (4) (2013) 1011–1023.
[31] M. Soltanolkotabi, E. Elhamifar, E.J. Candes, Robust subspace clustering, Ann. Stat. 42 (2) (2014) 669–699.
[32] B. Mcwilliams, G. Montana, Subspace clustering of high-dimensional data: a predictive approach, Data Mining Knowl. Discov. 28 (3) (2014) 736–772.
[33] L. Zhu, L. Cao, J. Yang, J. Lei, Evolving soft subspace clustering, Appl. Soft Comput. 14 (2014) 210–228.
[34] G. Gan, J. Wu, A convergence theorem for the fuzzy subspace clustering (FSC) algorithm, Pattern Recognit. 41 (6) (2008) 1939–1947.
[35] G. Guo, S. Chen, L. Chen, Soft subspace clustering with an improved feature weight self-adjustment mechanism, Int. J. Mach. Learn. Cybern. 3 (1) (2012) 39–49.
[36] B.J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (5814) (2007) 972–976.
[37] M. Mézard, Where are the exemplars? Science 315 (5814) (2007) 949–951.
[38] J.-H. Yu, X. ming Bai, J. wei Lv, Affinity propagation clustering based on new similarity (in chinese), J. Chin. Comput. Syst. 34 (3) (2013) 602–605.
[39] J.K. Antony, An efficient and fast density conscious subspace clustering using affinity propagation, Int. J. Recent Trends Eng. Technol. 6 (1) (2011) 140–142.
[40] Y.-H. Chu, J.-W. Huang, K.-T. Chuang, D.-N. Yang, M.-S. Chen, Density conscious subspace clustering for high-dimensional data, IEEE Trans. Knowl. Data Eng. 22 (1) (2010) 16–30.
[41] M. de Souto, I. Costa, D. de Araujo, T. Ludermir, A. Schliep, Clustering cancer gene expression data: a comparative study, BMC Bioinformatics 9 (1) (2008) 1–14.
[42] C. Aggarwal, J. Wolf, P. Yu, C. Procopiuc, J.S. Park, Fast algorithms for projected clustering, in: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, ACM Press, 1999, pp. 61–72.
[43] Y. Cui, C.-H. Zheng, J. Yang, Identifying subspace gene clusters from microarray data using low-rank representation, PLoS ONE 8 (3).
[44] G. Priyadarshini, R. Sarmah, B. Chakraborty, D.K. Bhattacharyya, J.K. Kalita, An effective graph-based clustering technique to identify coherent patterns from gene expression data, Int. J. Bioinform. Res. Appl. 8 (1/2) (2012) 18–37.

Please cite this article as: G. Gan, M.K.-P. Ng, Subspace clustering using affinity propagation, Pattern Recognition (2014), http://dx.doi.org/10.1016/j.patcog.2014.11.003

Guojun Gan received his Ph.D. in applied mathematics in 2007 from York University, Toronto, ON, Canada. In August 2014, Dr. Gan will join the University of Connecticut, Storrs, CT, as an Assistant Professor. Prior to that, he was the Director of Variable Annuity Hedging Research & Development at Manulife Financial. His research is on variable annuity valuation and hedging, open source variable annuity valuation systems, and high dimensional data and large data clustering.

Michael Kwok-Po Ng is a Professor in the Department of Mathematics and Professor (Affiliate) of Department of Computer Science at the Hong Kong Baptist University. He obtained his B.Sc. degree in 1990 and M.Phil. degree in 1992 at the University of Hong Kong, and Ph.D. degree in 1995 at Chinese University of Hong Kong. He was a Research Fellow of Computer Sciences Laboratory at Australian National University (1995–1997), and an Assistant/Associate Professor (1997–2005) of the University of Hong Kong before joining Hong Kong Baptist University.

Please cite this article as: G. Gan, M.K.-P. Ng, Subspace clustering using affinity propagation, Pattern Recognition (2014), http://dx.doi.org/10.1016/j.patcog.2014.11.003