Pattern Recognition 48 (2015) 1455–1464

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
Available online 11 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