Pattern Recognition 43 (2010) 1306–1319

Contents lists available at ScienceDirect

Pattern Recognition

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

A new clustering algorithm for coordinate-free data

Alejo Hausner

University of New Hampshire, Durham, NH 603-862-1237, USA

ARTICLE INFO

Article history:
Received 18 July 2008
Received in revised form
1 August 2009
Accepted 28 October 2009

MSC:
05C15
62H30
54E35

Keywords:
Cluster analysis
Graph coloring
Metric space
Partition

ABSTRACT

This paper presents the colored farthest-neighbor graph (CFNG), a new method for finding clusters of similar objects. The method is useful because it works for both objects with coordinates and for objects without coordinates. The only requirement is that the distance between any two objects be computable. In other words, the objects must belong to a metric space. The CFNG uses graph coloring to improve on an existing technique by Rovetta and Masulli. Just as with their technique, it uses recursive partitioning to build a hierarchy of clusters. In recursive partitioning, clusters are sometimes split prematurely, and one of the contributions of this paper is a way to reduce the occurrence of such premature splits, which also result when other partition methods are used to find clusters.

1. Introduction

This paper presents a new method for cluster analysis: the colored farthest-neighbor graph (CFNG). Cluster analysis studies the problem of finding groups of similar objects. Given a set of objects, this problem amounts to splitting the set into clusters or groups, such that the objects in each group are more similar to each other than to the objects in other groups.

A basic assumption of cluster analysis is that little is known about the data initially. The rules or characteristics that will define a group are not known *a priori*. Since clusters are identified based on similarity between objects, often the only assumption made in cluster analysis is that, given any two objects in the data, we can compute a positive number which expresses their degree of similarity. This degree of similarity is usually expressed as a distance: if two objects are very similar the distance between them is small, while two less-similar objects will have a larger distance between them.

Knowing the degree of similarity between objects does not reveal what caused clusters of objects to occur in the first place. We do not expect to learn everything there is to know about the data. Most often, identifying clusters in the data is simply a *first step* to discovering useful information about the data itself.

For example, the data may be genetic samples: here, each object is a version of the same gene, found in several different individuals of the same species, or from individuals in several related species. The information in each object is a sequence of DNA base pairs. The distance between two such genes is defined as the number of places where the base pairs are different. Such differences are due to mutations, and more mutations imply that more generations have passed since the two individuals descended from a hypothetical common ancestor. Conversely, fewer mutations means the two genes are more similar, and that they descended from a more recent common ancestor. With this definition of similarity, cluster analysis aims at finding groups of related individuals. Notice that the analysis may not yield any more information beyond discovering such groups. Actually discovering geographic, historical, environmental or other causes for the observed groups remains a task for a specialist in the domain. As stated above, cluster analysis simply provides a first stage in knowledge discovery.

1.1. Metric spaces and embedding

If we can compute degrees of similarity between objects, we are stating that we know the objects belong to a *metric space*. However, sometimes we also know that the objects are *embedded*. Before proceeding, let us define these two terms more carefully.

A metric space consists of a set of objects S, and a function $d(x, y)$ called the *metric* which takes two objects in the set and

E-mail addresses: ah@cs.unh.edu, hausner@dwc.edu.
URLS: http://www.cs.unh.edu/~ah, http://faculty.dwc.edu/hausner.

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.10.018

yields a positive real number. This number is the distance between the two objects. If two objects x and y are similar, their distance $d(x,y)$ will be small, and if the objects are very different, it will be large.

The metric function $d(x,y)$ usually has the following properties:

1. $d(x,y) > 0$ iff $x \neq y$ (non-negativity).
2. $d(x,y) = d(y,x)$ (symmetry).
3. $d(x,y) + d(y,z) > d(x,z)$ (triangle inequality).

The CFNG method works even for metrics $g(x,y)$ that do not obey the triangle inequality; such metrics are called *semimetrics*.

A set of objects is said to be embedded if each object has coordinates. For example, 2D points on the plane and 3D points in space are embedded; the number of coordinates attached to each object is the *dimension* of the space. Coordinates give the objects position, and from them, distances can be obtained. Not all objects are embedded. For example, words are sequences of characters, and web pages can be identified as collections of words, and neither of these has natural coordinates. This means that words and web pages do *not* have position, *i.e.*, are not embedded.

In two dimensions, the distance between two points can be computed using Pythagoras' theorem, and this theorem generalizes to the Euclidean distance used in higher dimensions. This implies that embedded objects also belong to a metric space. However, the converse does not hold: many interesting data sets exist which are in a metric space, but are not embedded. This distinction is important for the CFNG method. The method's only requirement is that the objects being clustered belong to a metric space. This differs from many popular methods, which rely on embeddedness: they often need to create a synthetic object that represents the "center" of a cluster, even though this center is often not itself an object in the original input set. For non-embedded data, such approaches cannot be used. They must be adapted in ways that render the methods less efficient.

2. Background and related work

Before presenting the CGNG method in detail, it is useful to survey popular approaches to cluster analysis. This survey will necessarily be incomplete: cluster analysis is a very well-researched area. In fact, early work predates the invention of digital computers. Many excellent reviews of clustering techniques have been written, including a recent book by Mirkin [21], and papers by Jain et al. [12] and Berkhin [4]. We cannot cover all previous work in the brief overview below. Hence we will focus on major techniques, and on specific methods that relate to the CFNG method. The interested reader is encouraged to read the reviews given above to obtain a broader outlook.

Popular methods for cluster analysis can be grouped into three main categories: k-means, agglomeration hierarchies, and partition hierarchies. The CFNG method, presented later in the paper, builds a partition hierarchy. Before proceeding further, we will briefly describe the three classes of clustering methods, to better understand where the CFNG method fits in. The first class is k-means.

2.1. k-Means

Given a set of objects, the k-means method finds clusters by initially guessing k initial cluster centers (see Fig. 1a, b, where $k = 2$). The initial cluster centers are chosen arbitrarily, usually randomly. The position of each center is then repeatedly adjusted

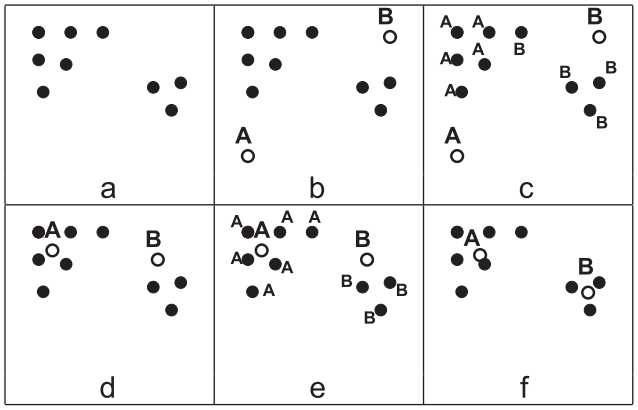

Fig. 1. The k-means algorithm: (a) shows a set of points on the plane; (b) two randomly placed cluster centers are chosen, and then (c) each point is given the label of its nearest cluster center. (d) The average (mean, or centroid) of each class of points is computed, and the each center is moved to its mean, at which time (e) the points are labeled once more, (f) centroids are re-computed, and the centers moved again. The algorithm has converged.

by classifying and averaging. For each iteration, the objects are first classified: each object is matched to the one center it is closest to (Fig. 1c). Then, the objects are averaged: for each center, the average position of all its matching objects is obtained (a total of k means are obtained, hence the method's name). Then, each center is moved to its corresponding mean position (Fig. 1d). Classification and averaging are repeated, until the centers stop moving (Fig. 1f), at which time the final classification of objects groups them into k different clusters.

The k-means method was first described by McQueen [18]. Running m iterations on a set of n objects costs $\Theta(mkn)$ time, so the method is fast. However, its simplicity leads to some limitations. First of all, it requires that k be equal to the number of clusters actually present in the data, a quantity that is by definition unknown at first. In other words, the k-means method simply assumes that there are k clusters, and then tries to estimate their *positions*. The iterative anomalous cluster [20] is a modification of k-means that avoids this assumption. This method begins with a single cluster, then adds new ones, each centered at the object farthest from existing cluster centers. However, such approaches require multiple iterations, and do not scale well to large data sets.

Another limitation is that the positions thus found depend on the initial choice of center positions: a different initial random choice for the k positions will often yield a different final set of cluster centers. This problem can be mitigated by re-running the algorithm several times, and stopping when the same set of centers is found several times. Unfortunately such a solution also loses some of the speed advantage of the k-means method.

For our purposes, a more serious limitation is that k-means requires embedded data: using an average to compute the center of a group of objects can only be done if the objects have coordinates. For non-embedded objects, this problem can be overcome by finding a *medioid object* instead of *center position*. The k-medioids or partitioning around medioids (PAM) approach was introduced by Kaufman and Rousseeuw [16]. For a set of objects in a metric space, the medioid is the object in the set that is nearest the "middle" of the set: it is the object x with the smallest total distance $\sum d(x,y)$ to all other objects y in the set. For a set of n objects, computing this medioid costs $\Theta(n^2)$ time. As a result, PAM is too slow for large data sets. Even for non-embedded data such as strings and graphs, it is possible to obtain a "mean" object for a set of objects, but the problem of obtaining such a representative is known to be NP-hard [13].