Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition Letters 28 (2007) 965–973
Pattern Recognition Letters
www.elsevier.com/locate/patrec

A method for initialising the *K*-means clustering algorithm using *kd*-trees

Stephen J. Redmond *, Conor Heneghan

Department of Electronic Engineering, University College Dublin, Belfield, Dublin 4, Ireland

Received 28 October 2005; received in revised form 29 July 2006
Available online 14 January 2007

Communicated by A. Fred

Abstract

We present a method for initialising the *K*-means clustering algorithm. Our method hinges on the use of a *kd*-tree to perform a density estimation of the data at various locations. We then use a modification of Katsavounidis' algorithm, which incorporates this density information, to choose *K* seeds for the *K*-means algorithm. We test our algorithm on 36 synthetic datasets, and 2 datasets from the UCI Machine Learning Repository, and compare with 15 runs of Forgy's random initialisation method, Katsavounidis' algorithm, and Bradley and Fayyad's method.

Keywords: Clustering; *K*-means algorithm; *Kd*-tree; Initialisation, Density estimation

1. Introduction and review

Clustering has long been the basis of many knowledge discovery tasks such as machine learning, statistics, data mining, and pattern recognition. There are two main branches of clustering; *Hierarchical* and *Partitional* (Jain et al., 1999). In this paper we concentrate on partitional clustering, and in particular a popular partitional clustering method called *K*-means clustering.

Partitional clustering involves partitioning a given set of data points into a number of distinct groups, termed *clusters*. Throughout this paper, by *data* we will be referring to n data points spanning an m dimensional space and we denote K as the number of clusters the data will be partitioned into. Partitional clustering attempts to partition the data points into clusters such that the similarity between the points in each cluster is maximal by some global measure. That is, if the points in each of the K clusters are maximally similar then changing a point's membership to another cluster will only serve to reduce the chosen global measure. Finding the optimum partition for a given global measure is known to be an NP-complete problem. However, there exist several suboptimal search strategies which find very competitive solutions within a reasonable amount of time. We describe one of the more popular of these methods next, the *K*-means algorithm.

1.1. K-means clustering algorithm

The algorithm of choice for many clustering tasks is the *K*-means algorithm (MacQueen, 1967). The *K*-means algorithm attempts to find the cluster centres, $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$, such that the sum of the squared distances (this sum of squared distances is termed the *Distortion, D*) of each data point $(\boldsymbol{x}_i)$ to its nearest cluster centre $(\boldsymbol{c}_k)$ is minimised, as shown in Eq. (1), where d is some distance function. Typically d is chosen as the Euclidean distance. A pseudo-code for the *K*-means algorithm is shown in Algorithm 1

* Corresponding author. Tel.: +353 87 9035170; fax: +353 1 2830921.
E-mail addresses: Stephen.Redmond@ee.ucd.ie (S.J. Redmond), Conor.Heneghan@ucd.ie (C. Heneghan).

0167-8655/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2007.01.001

$$D = \sum_{i=1}^{n} \left[\min_{k=(1\ldots K)} d(\boldsymbol{x}_i, \boldsymbol{c}_k) \right]^2 \quad (1)$$

Unfortunately the K-means algorithm is a local optimisation strategy and as such is sensitive to the choice of the initial positions of the cluster centres, $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$ (Pena et al., 1999). These initial centre locations are often termed the *seeds* for the K-means algorithm. The accepted initialisation technique has been to complete repeated runs of Forgy's algorithm (c.f. Section 1.2) and choose the final clustering which results from the run that returns the smallest value of the Distortion. However, while repeated runs of the Forgy method appears to be the *de facto* approach for initialising the K-means algorithm, many other techniques have been proposed. We now provide a brief summary of some of these techniques in Section 1.2.

Algorithm 1. K-means Algorithm

(1) Initialise K centre locations $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$.
(2) Assign each $\boldsymbol{x}_i$ to its nearest cluster centre $\boldsymbol{c}_k$.
(3) Update each cluster centre $\boldsymbol{c}_k$ as the mean of all $\boldsymbol{x}_i$ that have been assigned as closest to it.
(4) Calculate $D = \sum_{i=1}^{n} [\min_{k=(1\ldots K)} d(\boldsymbol{x}_i, \boldsymbol{c}_k)]^2$.
(5) If the value of D has converged, then return $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$; else go to Step 2.

1.2. A review of some initialisation methods

One of the earliest references to initialising the K-means algorithm was by Forgy in 1965 (see Anderberg, 1973). We shall term this approach the Forgy Approach (FA). Forgy simply suggested that K instances should be chosen from the database at random and used as the seeds. This approach takes advantage of the fact that if we choose points randomly we are more likely to choose a point near a cluster centre by virtue of the fact that this is where the highest density of points is located. However, there is no guarantee that we will not choose two seeds near the centre of the same cluster, or that we will not choose some poorly situated outlying point. In fact, repeated runs of this method is presently the standard method of initialising the K-means algorithm. We note that the primary negative aspect of repeated runs is the increased time taken to obtain a solution.

In 1967, McQueen introduced what was is akin to an online learning strategy to determine a set of cluster seeds (MacQueen, 1967). The first K points in the database are chosen as the preliminary seeds. Then, the next data point in the database is assigned to the cluster represented by the nearest seed. The centroid of that cluster is updated to be equal to the mean of all points assigned to it. The next data point is introduced, and the mean of its nearest cluster is update accordingly. This process is repeated until all data points are assigned to a cluster. The important point to note is that once a data point is assigned to a cluster its membership does not change. Hence, once all data points have been assigned, a point may be assigned to a cluster whose centroid is not the nearest. The K resulting centroids are used to seed the K-means algorithm. In a very large database this method can be somewhat burdensome, as a mean vector must be calculated every time a new instance is added.

Tou and Gonzales (1974) suggest the Simple Cluster-Seeking (SCS) method. Initialise the first seed with the first instance in the database. Calculate the distance between this seed and the next point in the database, if it is greater than some threshold then select it as the second seed, otherwise move to the next instance in the database and repeat. Once the second seed is chosen move to the next instance in the database and calculate the distance between it and the two seeds already chosen, if both these distances are greater than the threshold then select as the third seed. We continue until K seeds are chosen. We note that this method is dependent on the order of the points in the database, and, more critically, we must decide on a user-defined threshold value.

Linde et al. (1980) proposed a Binary Splitting (BS) method which was intended for use in the design of Vector Quantiser codebooks. It calls upon a hierarchical use of the K-means algorithm. The K-means algorithm is first run for $K = 1$. The cluster centre found, $\boldsymbol{c}$, is split into two clusters, $\boldsymbol{c} + \boldsymbol{\epsilon}$ and $\boldsymbol{c} - \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is some small random vector. The K-means algorithm is run again with these two points as seeds. When convergence is reached, the two cluster centres are again split to make four seeds and the K-means algorithm is run again. The cycle of split and converge is repeated until a fixed number of clusters is reached, or until each cluster contains only one point. This method clearly carries increased computational complexity, since after each split the K-means algorithm must be run. In addition, the quality of the clustering after each split depends upon the choice of $\boldsymbol{\epsilon}$, since this determines the direction in which each cluster will be split.

Kaufman and Rousseeuw (1990) suggest that we select the first seed as the most centrally located instance. Next we examine which of the points in the database, which when chosen as the next seed, will produce the greatest reduction in the Distortion, D (c.f. Eq. (1)). Once the second is chosen we choose the third seed in the same fashion and continue until K seeds are chosen. Again the first obvious drawback of this algorithm is the considerable amount of computation involved in choosing each seed. If this algorithm is to be considered useful for large databases a subsample of the instances must be used instead when finding the seeds (He et al., 2004).

Babu and Murty (1993) published a method of near-optimal seed selection using genetic programming. The population consists of various seed selections. The fitness of each seed selection is assessed by running the K-means algorithm until convergence and then calculating the Distortion value. The fittest solutions will then reproduce to