Volume 2, No. 3, May-June 2011

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Modified FPF algorithm for clustering the web

Dr. M. Hanumanthappa*
Reader,
Department of Computer Science & Applications,
Bangalore University,
Bangalore,India
hanu6572@hotmail.com

B R Prakash
Research Scholar,
Department of Computer Science & Applications,
Bangalore University,
Bangalore,India
brp.tmk@gmail.com

Manish Kumar
Research Scholar,
Department of Computer Science & Applications,
Bangalore University,
Bangalore,India
manishkumarjsr@yahoo.com

Abstract: Clustering is a widely used technique to partition data in homogeneous groups. It finds applications to Web text and video information retrieval. The main goal of clustering algorithms is to discover the hidden structure of data and group them without any a-priori knowledge of the data domain. Clustering is often used for exploratory tasks. In this paper we survey the principal strategies for clustering, the main clustering objective functions and related algorithms, first we discussed FPF algorithm for the k-center problem then we improved the Furthest-point-first algorithm in terms of speed and quality.

Keywords : Clustering, k-center, k-means, FPF algorithm

I. INTRODUCTION TO CLUSTERING

Clustering is a technique to split a set of objects in groups such that *similar* objects are grouped together, while objects that are not similar fall in different clusters. The choice of the notion of similarity (or distance) among objects that are needed to be clustered is of crucial importance for the final result. Clustering algorithms have no a-priori knowledge about the data domain, its hidden structure and also the number of hidden classes in which data are divided is unknown[1]. For this characteristic, clustering is often referred as un-supervised learning in contrast to classification (or supervised learning) in which the number of classes is known and for each class a certain number of examples are given. The independence of clustering algorithms from the data domain is at the same time the secret of its success and its main drawback. In fact since clustering does not need any a-priori knowledge of the data domain, it can be applied to a widerange of problems in different application areas Dealing with text documents is one of the foremost issues in information retrieval. In this context, clustering plays a strategic role. Large text document corpora have become popular with the growth of the Internet and the decrease of price of disk storage space and connection band-width[2].

A. Clustering Strategy

Clustering algorithms can be classified according with many different characteristics. One of the most important is the strategy used by the algorithm to partition the space [3]:

B. Partitional Clustering:

given a set **O = {O**1**, ... O**n**}** of n data objects, the goal is to create a partition **C = {C$_1$, . . . , C$_k$}** such that:
$\forall$ i $\in$ [1, k] Ci $\neq \emptyset$

When the data representation and the distance function *d* have been chosen, partitional clustering reduces to a problem of minimization of a given target function. The most widely used:

K-center minimizes the maximum cluster radius

$$\min_{j} \max_{x \in c_j} d(x, C_j)$$

C. FPF Algorithm for the k-Center Problem

One of the possible goals for partitional clustering is the minimization of the largest cluster diameter solving the k-center problem[4]. More formally the problem is defined as:

Definition 1. The k-centers problem: Given a set O of points in a metric space endowed with a metric distance function d, and given a desired number k of resulting clusters, partition O into non-overlapping clusters C*1*, . . . ,C*k* and determine their "centers" c*1*, . . . , c*k* $\in$ O so that max*j* max x$\in$C*j* d(x, c*j*) (i.e. the radius of the widest cluster) is minimized.

II. BASIC ALGORITHM

Given a set O of n points, FPF increasingly computes the set of centers c1 $\subset$ . . . $\subset$ ck $\subseteq$ O, where Ck is the solution to the k-center problem and C1 = {c1} is the starting set, built by randomly choosing c1 in O. At a generic iteration 1 < i ≤ k, the algorithm knows the set of centers Ci−1 (computed at the previous iteration) and a mapping μ that associates, to each point p O, its closest center μ(p) Ci−1. Iteration i consist of the following two steps:

A. Find the point p O for which the distance to its closest center, d(p, μ(p)),is maximum; make p a new center ci and let Ci = C*i−1* {ci}.
B. Compute the distance of c*i* to all points in O \ C *i−1* to update the mapping μ of points to their closest center.

After k iterations, the set of centers C*k* = {c*1*, . . . , c*k*} and the mapping μ define the clustering. Cluster C*i* is the set of points {p O \ C*k* such that μ(p) = c*i*}, for i [1, k]. Each iteration can be done in time O(n), hence the overall cost of the algorithm is O(kn)[5].

FPF:

```
Data: Let O be the input set, k the number of clusters
Result: C, k-partition of O
C = x such that x is an arbitrary element of O;
for i = 0; i < k; i + + do
    Pick the element x of O \ C furthest from the closest element in C;
    Ci = Ci = x;
end
forall x  O \ C do
    Let i such that d(ci, x) < d(cj , x), j ≠ i Ci append (x);
end
```

Algorithm 1*: The furthest point first algorithm for the k-center problem.*

III. K-MEANS

The k-means algorithm is probably the most widely used in the literature. Its success comes from the fact it is simple to implement, enough fast for relatively small datasets and it achieves a good quality. The k-means algorithm can be seen as an iterative cluster quality booster. It takes as input a rough k-clustering (or, more precisely, k candidate centroids) and produces as output another k-clustering, hopefully of better quality. K-means, as objective function, attempts to minimize the sum of the squares of the inter-cluster point-to-center distances[6]. More precisely, this corresponds to partition, at every iteration, the input points into non-overlapping clusters C1, . . . ,Ck and determining their centroids μ1, . . . , μk so that

$$\sum_{j=1}^{k} \sum_{x \in C_j} (d(x, \mu_j))^2$$

Is minimized.

It has been shown that by using the sum of squared Euclidean distances as objective function, the procedure converges to a local minimum for the objective function within a finite number of iterations. The main building blocks of k-means are [7]:

A. The Generation of the Initial k Candidate Centroids:

In this phase an initial choice of candidate centroids must be done. This choice in critic because both the final clustering quality and the number of iterations needed to converge are strongly related to this choice.

B. The main Iteration Loop:

In the main iteration loop, given a set of k centroids, each input point is associated to its closest centroid, and the collection of points associated to a centroid is considered as a cluster. For each cluster, a new centroid that is a (weighted) linear combination of the points belonging to the cluster is recomputed, and a new iteration starts.

C. The termination condition:

Several termination conditions are possible; e.g. the loop can be terminated after a predetermined number of iterations, or when the variation that the centroids have undergone in the last iteration is below a predetermined threshold. The use of k-means has the advantage that the clustering quality is steadily enough good in different settings and with different data. This makes k-means the most used clustering algorithm.

D. Initialize k-Means:

Essentially k-means accepts as input an initial clustering that can be made with any clustering algorithm. It is well-known that the quality of the initialization (i.e. the choice of the initial k centroids) has a deep impact on the resulting accuracy. Several methods for initializing k-means are:

RC The simplest initialization for k-means is the one in which the initial centroids are Randomly Chosen among the input points and the remaining points are assigned to the closest centroid. The resulting clustering is often referred as *random clustering*.

RP In the Random Perturbation, for each dimension dj of the space, the distribution of the projections on dj of the data points is computed, along with its mean μj and its standard deviation σj ; the k initial centroids are obtained through k perturbations, driven by the μj's and σj's, of the centroid of all data points.

MQ MacQueen's proposed a variant of k-means: the initial centroids are randomly chosen among the input points, and the remaining points are assigned one at a time to the nearest centroid, and each such assignment causes the immediate recomputation of the centroid involved. Then k-means is initialized with the resulting clustering

IV. IMPROVING THE FPF ALGORITHM FOR THE K-CENTER PROBLEM

In this paper we devoted to improve the Furthest Point First algorithm from both the computational cost point of view and the output clustering quality. Since theoretically the FPF algorithm as proposed by Gonzalez [8] is optimal (unless P = NP), only heuristics can be used to obtain better results and, in the worst case, it is not possible to go behind the theoretical bounds. We profiled FPF and analyzed the most computational expensive parts of the algorithm. We found that most of the distance computations are devoted to find the next furthest point. We observed that there are cases such that some distance computations can be avoided without changing the final clustering algorithm. FPF clustering quality can be improved modifying part of the clustering schema. We describe approaches that use the random sampling technique to improve clustering output quality, we call this algorithm M-FPF. Another crucial shortcoming of FPF is that it selects a set of centers not representative of the clusters. This phenomenon must be imputed to the fact that, when FPF creates a new center, it selects the furthest point from the previous selected centers and thus the new center can likely be close to a boundary of the subspace containing the data set. To overcome this problem we modify M-FPF to use *medoids* instead of centers.

A. Exploiting the Triangular Inequality to Improve the FPF Speed

We observed that most of the running time of the FPF algorithm is devoted to compute distances for finding the closest center to each point. More precisely at a generic iteration $1 < i \leq k$, after finding the center μk, n − k distances must be computed to decide whether or not to assign a point to the new center. If this is done in a straightforward manner it takes O(n) time per iteration, thus the total computational cost of the algorithm is O(nk).

Exploiting the triangular inequality, in certain conditions we can avoid to compute the distances among all the points in a cluster and the new center being sure that they are closer to their center. Unfortunately the worst case time complexity still remain O(nk) because the number of saved distance computations depends on data distribution and thus, it can not be predicted in advance.

B. We modified the Algorithm as Follows:

Consider, in the FPF algorithm, any center ci and its associated set of closest points Ci. Store Ci as a ranked list, in order of decreasing distance to ci. When a new center cj is selected, scan Ci in decreasing order of distance, and stop scanning when, for a point p Ci, it is the case that d(p, ci) ≤ ½ d(cj , ci). By the triangular inequality, any point p that satisfies this condition cannot be closer to cj than to ci. This rule filters out from the scan points whose neighbor cannot possibly be cj , thus significantly speeding up the identification of neighbors.

C. Using a Random Sample

The efficiency of the algorithm is further improved by applying FPF algorithm not to the whole data set but only to a random sample of size $n' = \sqrt{nk}$ of the input points [9]. Note that given that $k \leq n$, it is always true that $n' \leq n$. Then we add the remaining $(n - n')$ points to the cluster of their closest center, one by one.

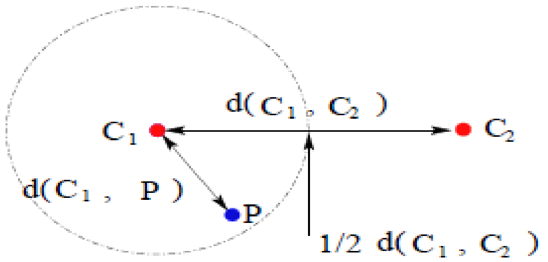

Figure1: Exploiting the triangular inequality.

Also in the operation of insertion of the $(n - n')$ remaining points, the bottleneck is the time spent computing distances to the point to the closest center.

This operation can be made more efficiently [] exploiting the triangular inequality even if the worst case running time does not change. Consider to have available the distances between all the pairs of centers of the clustering. Let p be the new point to be inserted in the clustering, by the triangular inequality if ½ d(ci, cj) > d(ci, p) then d(ci, p) < d(cj , p). It means that the computation of the distance d(cj , p) can be safely avoided.

```
μ-FPF:
Data: Let O be the input set, k the number of desired clusters
Result: C: a k-partition of O
Initialize R with a random sample of size √|O|k elements of O;
C = FPF(R, k);
forall Ci  C do
μi = getCenter (Ci);
end
forall p in O \ R do
assign p to cluster Ci such that d(p, μi) < d(p, μj), j 6= i;
end
```

***Algorithm 2**: M-FPF.*

D. Using Medoids as Centers

The concept of medoid was introduced by Kaufman and Rousseeuw [10]. Medoids have two main advantages with respect to centroids:

First of all, they are elements of the input and not "artificial" objects. This make medoids available also in those environments in which the concept of centroid is not well defined or results artificious. evertheless, in many environments (i.e texts) centroids tends to become dense objects with a high number of features more of which of poor meaning. This makes centroids to lose representativeness and compute distances with them becomes more expensive with respect to distances between "normal" objects.

The main drawback of the original definition is that the clustering algorithm (Partition around Medoids) and the computation of medoids is expensive to overcome this disadvantage many different re-definitions of medoids were introduced in the literature.

In the context of the Furthest Point First heuristic where some input points are elected as cluster centers and are used to determinate which input points belong to the cluster, the restrictions of the use of centroids are not present. However, we observed that, although the objects selected from FPF as centers determine the points belonging to the cluster, they are not "centers" in the sense suggested by the human intuition. The computation of the medoid is quadratic in the number of points of O. In fact, one should compute the distance between all the possible pairs of objects of the input in order to find the diametral points. Following [11] it is possible to find a good approximation a and b in linear time using the following search schema:

[a] Select a random point p Є O
[b] In O (n) find the furthest point from p and call it a
[c] In O (n) find the furthest point from a and call it b

A further approximation of the medoid computation is still possible. Although it reduces drastically the cost during the update procedure, it is quite rough and should be used only in those online contexts where computational time makes the difference or in those environments where there is a huge amount of redundant data. After the first time in which we find a, b and the medoid m, when a new point p is inserted in the cluster, the update of the medoid can be done using the following procedure:

i. If d(p, a) > d(a, b) □ d(p, a) > d(p, b) discard b and replace it with p
ii. If d(p, b) > d(a, b) □ d(p, b) > d(p, a) discard a and replace it with p
iii. If d(a, b) > d(p, a) □ d(a, b) > d(p, b):
If |d(p, a) − d(p, b)| + |d(p, a) + d(p, b) − d(a, b)| < |d(m, a) −
d (m, b)| + |d(m, a) + d(m, b) − d(a, b)| discard m and p become the new medoid
– Otherwise discard p

After the first initialization, this procedure requires only the computation of two distances

µ -FPF-MD*:*

```
Data: Let O be the input set, k the number of desired clusters
Result: C: a k-partition of O
Initialize R with a random sample of size √|O|k elements of O;
C = FPF(R, k);
forall Ci □ C do
    ti = getRandomPoint (Ci);
    ai = ci such that max d(ci, ti) for each ci □ Ci;
    bi = ci such that max d(ci, ai) for each ci □ Ci;
    mi = ci such that
    min |d(ci, ai) − d(ci, bi)| + |d(ci, ai) + d(ci, bi) − d(ai, bi)|;
end
forall p in O \ R do
    assign p to cluster Ci such that d(p,mi) < d(p,mj ), □j ≠ i;
    if d(p, b_i) > d(a_i, b_i) then ai = p ;
    if d(p, a_i) > d(a_i, b_i) then bi = p ;
    if d(p, b_i) > d(a_i, b_i) or d(p, a_i) > d(a_i, b_i) then
        mi = c_i such that
        min |d(c_i, a_i) − d(c_i, b_i)| + |d(c_i, a_i) + d(c_i, b_i) − d(a_i, b_i)|;
    end
end
```

Algorithm 3*: µ -FPF-MD.*

V. CONCLUSION

Clustering based approaches are proposed in the literature K-means is too slow to be applied to on-line contexts like the Web. Clustering is typically not considered as a possible approach due to its computational cost. In this paper we studied the problem of FPF algorithm for the k-center problem We Improving he FPF algorithm for the k-center problem by Exploiting the triangular inequality to improve the FPF speed Using medoids as centers called M-FPF-MD algorithm, further we plan to evaluate the performance of the algorithm for static and dynamic clustering.

VI. REFERENCES

[1] Domenico Cantone, Alfredo Ferro, Alfredo Pulvirenti, Diego Reforgiato Recupero, and Dennis Shasha. Antipole tree indexing to support range search and k-nearest neighbor search in metric spaces. IEEE Transactionson Knowledge and Data Engineering, 17(4):535–550, 2005.

[2] Fazli Can, Ismail Sengor Altingovde, and Engin Demir. Efficiencyand effectiveness of query processing in cluster-based retrieval. Inf.Syst., 29(8):697–717, 2004.

[3] Kenneth L. Clarkson. Nearest-neighbor searching and metric space dimensions. In Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors, Nearest-Neighbor Methods for Learning and Vision: Theory and Practice, pages 15–59. MIT Press, 2006.

[4] Charles Elkan. Using the triangle inequality to accelerate k-means.In ICML, pages 147–153, 2003.

[5] Mayank Bawa, Tyson Condie, and Prasanna Ganesan. Lshforest: self-tuning indexes for similarity search. In Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005, pages 651–660, 2005.

[6] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In KDD '01: Proceedings of the seventh ACM SIGKDD international conferenceon Knowledge discovery and data mining, pages 245–250, New York, NY, USA, 2001. ACM.

[7] E. Ch´avez and G. Navarro. An effective clustering algorithm to index high dimensional metric spaces. In Proceedings of the6th International Symposium on String Processing and Information Retrieval (SPIRE'2000), pages 75–86. IEEE CS Press, 2000.

[8] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. Theoretical Computer Science, 38(2/3):293–306, 1985.

[9] Piotr Indyk. Sublinear time algorithms for metric space problems. In Proceedings of STOC-99, ACM Symposium on Theory of Computing, pages 428–434, 1999.

[10] Leonard Kaufman and Peter J. Rousseeuw.Finding groups in data: an introduction to cluster analysis. Wiley, 1990. A Wiley-Interscience publication.

[11] Flavio Chierichetti, Alessandro Panconesi, Prabhakar Raghavan, Mauro Sozio, Alessandro Tiberi, and Eli Upfal. Finding near neighbors through cluster pruning. In Proceedings of ACM PODS, 2007. To appear.

[12] Paolo Ferragina and Antonio Gulli. A personalized search engine based on Web-snippet hierarchical clustering. In Special InterestTracks and Poster Proceedings of WWW-05, 14th International Conference on the World Wide Web, pages 801–810, Chiba, JP, 2005.

[13] Karina Figueroa, Edgar Ch´avez, Gonzalo Navarro, and Rodrigo Paredes. On the least cost for proximity searching in metric spaces. In 5th International Workshop on Experimental Algorithms (WEA), volume 4007 of Lecture Notes in Computer Science, pages 279–290. Springer, 2006.

[14] M. Furini, F. Geraci, M. Montangero, and M. Pellegrini.VISTO: VIsual Storyboard forWeb Video Browsing. In CIVR '07: Proceedingsof the ACM International Conference on Image and Video Retrieval, July 2007.

[15] Filippo Geraci Ph.D Thesis Fast Clustering For Web Information Retrieval Universit`A Degli Studi Di Siena Facolt´A Di Ingegneria Dipartimento Di Ingegneria Dell'informazione Anno Accademico 2007-2008

[16] M. Furini. On ameliorating the perceived playout quality in chunkdriven p2p media streaming systems. In ICC ’07: Proceedings of the IEEEInternational Conference on Communications,2007.