A novel PSO based algorithm to find initial seeds for the k-means clustering algorithm

Conference Paper · November 2016

DOI: 10.1201/9781315364094-30

3 authors, including:

Lavika Goel
Birla Institute of Technology and Science Pilani
36 PUBLICATIONS **194** CITATIONS
SEE PROFILE

Shivin Srivastava
Birla Institute of Technology and Science Pilani
2 PUBLICATIONS **10** CITATIONS
SEE PROFILE

A novel PSO based algorithm to find initial seeds for the k-means clustering algorithm

Lavika Goel, Nilay Jain, Shivin Srivastava

Abstract— **Clustering is a very fundamental problem in machine learning and data mining. Many algorithms have been proposed but none is more popular than the k-means algorithm developed by Lloyd around 50 years ago. K-means is a fast and simple algorithm but often gives a sub-optimal clustering. This is due to the random initialization that is employed in the simple K-means algorithm. K-means++ provides a new way to seed the k-means algorithm that is O(log k)-competitive with optimal clustering. In this paper we use Particle Swarm Optimization to find the initial seeds of k-means algorithm. Our approach basically tries to select the k centroid points as far away as possible from each other so that good clusters are generated.**

I. INTRODUCTION

Swarm intelligence (SI) is the collective behavior of decentralized, self-organized systems, natural or artificial. This concept is employed in work in artificial intelligence. SI systems consist typically of a population of simple agents or bodies interacting locally with one another and with their environment. The inspiration often comes from nature, especially biological systems.[ref] The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local, and to a certain degree random, interactions between such agents lead to the emergence of "intelligent" global behavior, unknown to the individual agents. Swarm Algorithms can be applied to a variety of problems like optimization problems, scheduling problems, clustering problems etc. In this paper we propose to apply swarm algorithms to data clustering and cluster analysis problems.

Cluster analysis falls into unsupervised learning category of machine learning. Here we do not have any prior information about the category labels of the data points. The task is to cluster the data points in such a manner that, data points falling into the same cluster are related to each other. One metric that we use to specify good clustering is that the intra cluster distance between the data points should be small and the inter cluster distance must be large. Clustering has a lot of applications in the scientific fields ranging from Computational Biology and Medical Imaging to Market Research, etc.

One of the most popular clustering algorithms is K-means. This algorithm is still widely used despite being proposed more than 50 years ago. There have been many more clustering algorithms invented since then, but the simplicity and scalability of k-means algorithm makes it one of the most widely used clustering algorithm even today.

K-means++ algorithm is an addition to the k-means algorithm to give us an even better clustering on the data. More precisely it gives a (logk) competitive bound with the optimal clustering. The strategy that k-means++ algorithm employs is in the initialization of the clusters. k-means++ tries to select k initial cluster points in such a manner that each cluster point i is chosen at random from the weighted probability distribution of the distances of point i from the other i-1 points. This initialization strategy makes k-means++ give a (logk) competitive bound with the optimal clustering.[3]

II. RELATED WORK

The clustering problem and k-means algorithm have a very rich history. Because k-means algorithm is very simple and has a good observed speed, it is one of the most widely used clustering algorithms ever since being introduced by Lloyd in 1955 [1]. The simplicity and scalability of k-means algorithm implied that it was very widely adopted peer reviewed in the computer science community.

K-means begins with k centers that are chosen randomly from the data points. Each point is assigned to a centroid, and then we compute the

centroids again, to the mean of all the points assigned to that centroid. This process is done repeatedly until the algorithm converges.[Lloyd ref] The problems that came to be associated with k-means was the random initialization step. The optimal clustering is given by k-means algorithm only when each initial point was part of one cluster. This means the number of iterations for k-means initialization should be increased when the number of clusters were more [2][5]. But the probability that each initial cluster centroid was part of one cluster in the final clustering was still small. Hence increasing the number of iterations still not helped and we were left with suboptimal clustering.

K-means++ algorithm solved the problem of suboptimal clustering by focusing on the initialization technique of k-means. Instead of randomly initializing the k-cluster points, David Arthur et. al. proposed in their remarkable paper that initializing k particles based on the probability of distance of the new cluster from the existing cluster points gives a more ”optimal” clustering. The larger the distance of particle from currently chosen points, the greater is its probability of getting selected as a cluster point [3][4]. This paper analyses mathematically the benefits of using such an initialization, and they arrive at the conclusion that using such an initialization gives us an O(log k) competitive algorithm to optimal clustering.

The k-means++ initialization technique, is an NP hard problem known as the k-center problem in the computer science literature. The k-center problem is defined as following: Given n cities with specified distances, we want to build k warehouses in different cities such that the sum of maximum distance of a city to a warehouse is minimum. Dasgupta[6] analyses the approximate methods to solve the k-center problem using Farthest First traversal and Covering Numbers. The initialization technique used by David Arthur et. al. is a probabilistic variant of the greedy method Farthest first traversal.

K-means++ has a further advantage in that it can be made extremely scalable by parallelizing it as referenced in the paper by Sergei Vassilvitskii et. al. [9][10].

We plan to apply Particle Swarm Optimization technique, to find the solution of the k-center problem and use it to initialize the k-means algorithm. Particle Swarm Optimization (PSO) is a novel algorithm developed by Kennedy et. al. [7]. PSO was introduced to optimize continuous nonlinear functions. Though now there are many variants of PSO that can work in different settings like dynamic environment of PSO, Multi-objective optimization and Discrete PSO. PSO was discovered by simulating behaviours in a social setting. In PSO we allow particles to wander in a search space. PSO tries to optimize the given fitness function. Given this function, particles remember their personal best and global best position values, on the basis of which a converging point for all particles is obtained which is the solution to the equation which was described by the fitness function.

PSO has been applied in data clustering by van der merwe et. al. [8]. They use PSO to initialize the k-centroids and then extend this method to apply k-means.

III. A BRIEF REVIEW OF K-MEANS++ AND HYBRID K-MEANS/PSO TECHNIQUES

This section provides a brief review of the k-means++ algorithm, Hybrid PSO with k-means and other techniques that have been used in literature up until now.

A. k-means++ algorithm

The k-means++ algorithm starts by choosing the first centroid arbitrarily.[ref]. Let D(x) denote the shortest distance of a data point to the cluster we have already chosen, then k-means++ initialization is defined as follows[3]:

1) Choose an initial center c_1 uniformly at random from X.
2) Choose the next center c_i , selecting $c_i = x \in X$ with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$.
3) Repeat Step 1b until we have chosen a total of k centers.

B. Hybrid k-means PSO

In the Hybrid k-means PSO technique, the strategy adopted while performing PSO is to concatenate the dimensions of the n data points to get

into a n*d dimensional space. Then the optimal centroid point is found using PSO with the sum of squared distance error as the fitness function metric which represents the optimal solution. Here is the algorithm[8]:

1) Initialize each particle to contain N, randomly selected cluster centroids.
2) For t = 1 to $t_{max} do$
   3) For each particle i do
   4) For each data vector $z_p Calculate the Euclidean distance d(z, m_i) to all cluster centroids C_{ij}$
5) Assign $z_p to cluster C_{ij} such that d(z_i, m_{ij}) = min Calculate the fitness using equation (8)$
6) Update the global best and local best positions
8) Update the cluster centroids using equations (3) and (4).

But the idea of concatenating so many dimensions gives rise to an inefficient algorithm and we plan to improve on this strategy with our method.

IV. Proposed Methodology

In this section we describe our main algorithm.

A. Challenges

Since the standard PSO works on the continuous space and our problem involves choosing k points from the set of n points, we needed to find a suitable mapping from the continuous space to the discrete space. We decided to map the n points which were originally in the continuous space to a binary ceil(lg n) dimensioned hypercube whose vertices encode the data points in the original euclidean space. This restricts the movement of search agents to the n points only.

Next was the problem of adapting the PSO algorithm for this modified search space. The optimum point to which the PSO search agents must converge to represents the configuration of the k particles. So ideally this required us to move to another higher dimensional search space where each point represents a configuration of the k points, as seen in [8]. Since this would be a costly affair both in terms of time and space, we decided to use the hypercube as search space only.

In our modified approach the k particles play a dual role. They are search agents of the PSO algorithm which not only explore the search space but whose final configuration in the hypercube represents the position the k selected points. This deviation from the original PSO required us to modify the optimizing function so that we could make sense of the particles positions. Two different optimising functions were used i.e. one for calculating global best and one for calculating local best of the particles. Our selection of the functions is crucial as the algorithm finally needs the local and global best positions to converge.

Also, the meaning of velocity and distance between particles needs to redefined for particles in a hypercube because the particle is constrained to move only at the corners of the hypercube and nowhere else. We define the distance or separation between the particles as their edit distance, as described in the next subsection.

B. Approach

We aim to use PSO to disperse the 'k' centroids as far away from each as possible. The standard PSO algorithm[7] leaves a fixed number of agents in the search space which follow a particular heuristic rule to explore and exploit the search space. Since we want to select k points as initial centroids our metric of choosing a good solution is the sum of inter particle distances.

To find the optimum configuration of the particles, we modify our PSO algorithm so that instead of the particles converging at a single point representing the optimum configuration in a higher dimensional space (as seen in [8]), the configuration of the particles in the search space itself becomes the optimum configuration. The PSO is performed in the discretized space in the hypercube, where each data point is at the corner of the hypercube. This constraints the PSO particles to move only inside the hypercube. The changed "interpretations" of position and velocity inside the hypercube were described as follows: Let key of any data point be denoted by k. Position of particle inside the hypercube is defined as the binary representation of key k. Velocity of particle inside the hypercube is a integer number n. To move the particle at position p with velocity v, we flip randomly v bits of the particle from the binary position vector p.

The global fitness function maximizes the distance of PSO agents from each other. [equation for the global fitness function]. The local fitness function maximizes the fitness of one particular particle. [equation for the local fitness function]. We prove in the following theorem that the global and local fitness functions achieve the same objective and that they complement each other.

C. Flowchart

D. Psuedocode

Algorithm 1 k-means PSO

1: **procedure** PSO INITIALIZATION(data matrix X, k)
2: $n \leftarrow$ number of data points
3: *Map data points to hypercube with ceil(lg(n)) dimensions*
4: *Define the global and local fitness functions:*
   1) $J(X) = \sum_{i=1}^{k} \sum_{j=i+1}^{k} ||x_i - x_j||^2$
   2) $H(x_p) = \sum_{i=1}^{k} ||x_i - x_p||^2$
5: $pbest \leftarrow$ *matrix of size* : $k \cdot ceil(lgn)$
6: $gbest \leftarrow$ *matrix of size* : $k \cdot ceil(lgn)$
7: $velocity \leftarrow$ *matrix of size* : $k \cdot ceil(lgn)$
8: $position \leftarrow$ *matrix of size* : $k \cdot ceil(lgn)$
9: *PSO Equation*:
   1) $v[i] = v[i] + c1 * rand(-1, 1) * (pbest[i] - position[i]) + c2 * rand(-1, 1) * [gbest[i] - position[i])$
   2) $present[i] = present[i] + v[i]$

Now we apply PSO and store the last 10 global best positions of the particles. We apply k-means algorithm 10 times with one global best configuration being input as an initial seed at a time. We evaluate the results and performance of our algorithm in the next section.

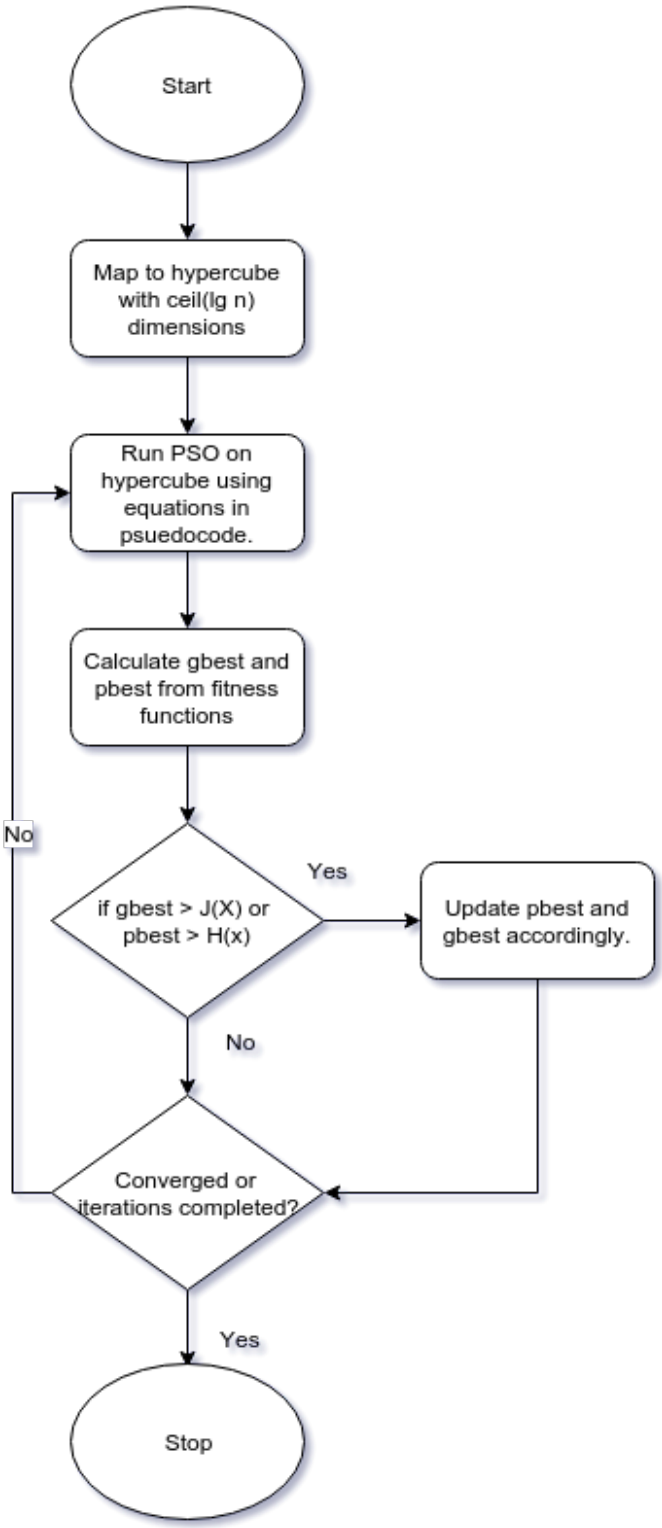

Fig. 1. Moth following the source of light

TABLE I

COMPARATIVE RESULTS ON WINEQUALITY-WHITE DATASET

k	k-means-random	k-means-VanDerMerwe	k-means++	k-means PSO
6	1544344.6209	-	1544327.8491	1544327.8491
9	1146811.0615	-	1141490.8237	1139974.6475
12	941364.00364	-	899171.39676	897445.64313

TABLE II

COMPARATIVE RESULTS ON REDWINE-WHITEWINE-DATASET

k	k-means-random	k-means-VanDerMerwe	k-means++	k-means PSO
2	8589514.6373	-	8589514.6373	8589514.6373
6	2041453.5289	-	2041432.8925	2041680.3178
10	1365612.9213	-	1370097.5526	1364864.6436

TABLE III

COMPARATIVE RESULTS ON IRIS-DATASET

k	k-means-random	k-means-VanDerMerwe	k-means++	k-means PSO
3	78.918808773	-	78.918808773	78.918808773
6	38.882367282	-	38.856319175	38.856232051
9	28.294123636	-	28.150368578	28.383452381

V. Results

We performed k-means clustering using random initialization, using van der merwe's method, k-means++ initialization and pso initialization. Following SSE(sum of squared errors) scores were obtained on the standard datasets as shown in tables 1-3.

VI. Conclusions

In this work we have introduced a novel algorithm to solve the k center problem and applied it for selecting the k centroids for initializing the k means algorithm. We have presented a modified PSO algorithm to select k initial points to fed into the k-means means algorithm as the seed points.

References

[1] Stuart P. Lloyd, Least squares quantization in pcm, IEEE Transactions on Information Theory, 28(2):129 136, 1982.

[2] David Arthur and Sergei Vassilvitskii, How slow is the k-means method?, In SCG 06: Proceedings of the twenty-second annual symposium on computational geometry. ACM Press, 2006.

[3] David Arthur and Sergei Vassilvitskii, k-means++: The Advantages of Careful Seeding, 2007

[4] David Arthur and Sergei Vassilvitskii, k-means++ test code. http://www.stanford.edu/ darthur/kMeansppTest.zip.

[5] Sanjoy Dasgupta, How fast is k-means? In Bernhard Scholkopf and Manfred K. Warmuth, editors, COLT, volume 2777 of Lecture Notes in Computer Science, page 735. Springer, 2003

[6] k center problem, dasgupta, CS291 Geometric Algorithms, lecture 1. http://cseweb.ucsd.edu/ dasgupta/291-geom/kcenter.pdf

[7] J Kennedy, RC Eberhart, Particle Swarm Optimization, Proceedings of the lEEE International Joint Conference on Neural Networks, Vol. 4, pp 1942-1948, 1995.

[8] DW van der merwe, A.W Engelbrecht, Data Clustering using particle swarm Optimization, Evolutionary Computation, CEC, 2003.

[9] Bahmani, Moseley, Vattani, Kumar and Vassilvitskii, Scalable k-means++, Stanford University and Yahoo Research, 2011.

[10] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases, SIGMOD Record, 25:103114, 1996.