International Journal of Education, Modern Management, Applied Science & Social Science (IJEMMASSS)
ISSN : 2581-9925, Impact Factor: 6.340, Volume 03, No. 02(IV), April - June, 2021, pp.60-65

AN EFFICIENT APPROACH BASED ON PARTICLE SWARM OPTIMIZATION FOR CLASSIFICATION

Shruti Tripathi[*]
Ajay Kumar Bharti[**]
Brijesh Tewari[***]
Himanshu Pandey[****]

ABSTRACT

The big quantities of records have created a requirement for new frameworks for processing of data. With the fast advancement of web, the quantity of information that are gathered or created in numerous spaces creates difficulties to mainstream researchers on account of the volume and complications of the records. The need of investigation techniques that pullout helpful data for decision making has been getting more consideration with the end goal for specialists to get a scalable answer for conventional method. In this paper, we proposed a scalable plan and execution of a Parallel Particle Swarm Optimization approach that depends on the Apache Spark system. The fundamental thought of the of this technique is to get the ideal centroid for each target name utilizing molecule swarm streamlining and afterward find out unlabeled information focuses to the nearest cancroids. Two variations of approach Proposed Algorithm-F1 and Proposed Algorithm-F2 were proposed dependent on various fitness functions that can be proficiently parallelized utilizing the Apache Spark structure in future.

Keywords: *Particle Swarm Optimization, Big Data, Classification, Centroid, Apache Spark System*.

Introduction

Data mining is a multidisciplinary research region that consolidates AI, insights, and database exploration and means to examine authentic data to extricate important data [1]. With the continually developing amount of records, the need for frameworks to process these records is growing. In 2014 IDC was anticipated that with the end of 2020, the digital universe might be 10 times as large because it was in 2013, totaling an mind-blowing 44 zettabytes [2]. Big Data isn't always only a big amount of facts, however a new paradigm and set of technology which could save and process these facts. In this context, a set of new frameworks required on storing and processing big volumes of records have emerged.

The Swarm intelligence (SI) strategy is a model that emulates the social conduct of a gathering of people and how they connect with one another furthermore, with their current circumstance to accomplish a specific objective, eg, tracking down the best food source. In the gathering of people, there is no pioneer to direct the people; every individual in some way or other acts somewhat haphazardly inside the pursuit space and offer data with its neighbors until the best wellspring of food in a specific region is found. Since the 1990s, a considerable lot of the SI techniques have been proposed which are propelled by different frameworks. For example, Ant colony optimization (ACO) is roused by the scrounging conduct of a subterranean insect province, which mimics the conduct of subterranean insects looking for the ideal way between their home and a food source. Essentially, particle swarm optimization (PSO) mimics the movement of bird running when they are looking for food in a specific region. These SI strategies were essentially proposed to advance single-objective also; multi-target works and takes care of NP-difficult issues. In any case, during the last decade, SI strategies have been effectively applied to address information mining issues like arrangement or bunching or to upgrade the underlying boundaries of an order calculation, for example, support vector machine.

[*] Department of Computer Science, Maharishi University of Information Technology, Lucknow, U.P., India.
[**] Department of Computer Science, Maharishi University of Information Technology, Lucknow, U.P., India.
[***] National Informatics Center, Ministry of Communication and IT, Government of India, India.
[****] Faculty of Engineering & Technology, Lucknow University, Lucknow, U.P., India.

Literature Review

In Zhao et al's work[3] the authors suggested a parallel version of the clustering algorithm for the K-means using the Map Reduce system. In their proposed method, the K-means algorithm for finding cluster centroid points is formulated as a Map Reduce task. The Map function assigns each data point to the nearest centroid in the Map Reduce job while the Reduce function updates the centroids. The Map Reduce job is repeated until it reaches the stop condition. The experimental results revealed that the algorithm can efficiently process Big Data on a node cluster.

In Ludwig 's work, the parallelization as well as scalability of a common and efficient fuzzy clustering algorithm called the Fuzzy C-Means (FCM) algorithm was proposed.[4] Using the Map Reduce framework, the algorithm was paralleled. A validity analysis was performed to prove that the implementation works correctly, obtaining results of competitive consistency compared with state-of-the-art clustering algorithms. In addition, a scalability analysis was carried out to demonstrate the efficiency of the parallel implementation of FCM in order to increase the number of computing nodes used.

The authors proposed in Wang et al's work[5] a parallel K-means clustering algorithm based on the Apache Spark system to address the limitations of the K-means algorithm given by the Spark MLIB library. The algorithm's main function is to perform the distance calculation on the worker nodes between data points and centroids, and the centroid update on the master node. The algorithm was tested with real data sets, and the results showed that the parallel K-means algorithm was accurate and successful.

Yang and Li[6] suggested a parallel clustering algorithm to ant-colony using the Map Reduce system. Using the ant colony algorithm heterogeneous Big Data are automatically decomposed into clusters in their work. The algorithm has been checked with large data sets and the experimental results show that the algorithm achieves high precision with good efficiency.

Particle Swarm Optimization Algorithm

The well renowned PSO algorithm has been introduced by Kennedy and Eberhart, a stochastic research approach motivated by the social natural behavior of a group of birds in pursuit of the best food source in the region. In this algorithm every particle is a possible solution inside a solution space that moves through a search space in search of the best solution. The Inertia Weight (*w*), Cognitive Learning (c1) and Social Learning (c2) are the three factors affecting the direction of movement of a particle in terms of controlling the mobility of the particle from its current position, determining how well the particle is pursuing the best location it has so far found and controlling the extent to which the particle accompany the best position which was searched by the whole group respectively.

Algorithm 1 presents "the PSO Algorithm pseudocode. A vector at random velocity v = {v1, v2, v3, ..., vn}. The aptitude function measures the aptitude for each particle in each PSO iteration by measuring the particle's location to determine how close this particle is to achieving its target (the best solution). The best personal position (PBest) and best global position (GBest) will be updated according to the physical state. PBest is the best location so far discovered by a single particle, and GBest is the best location so far discovered by any particle in the entire swarm. The current vector velocity for each particle is then modified using PBest and GBest as follows:"

$$v_j^{(t+1)} = wv_j^t + r_1c_1\left(PBest - p_j^t\right) + r_2c_2\left(GBest - p_j^t\right) \quad (1)$$

Here, in above equation p_j^t and v_j^t are "the current position vector and the current particle vector j at iteration t, respectively; r1 and r2 are random vectors and c1 (cognitive learning) and c2 (social learning) are constant user-specified coefficients. The inertia weight value in our research is decreasing linearly with increasing numbers of iterations determined using Equation (2) at each iteration. $t.^{25}$ v_j^{t+1} is a new velocity vector of particle *j*, where the new value at each dimension of the velocity vector is clamped within the range [v_{min}, v_{max}] "

$$w(t) = w_{max} - \left((w_{max} - w_{min})\frac{t}{T_{max}}\right) \quad (2)$$

Then, "the new position of the particle $j\,(p_i^{(t+1)})$ is computed using the current particle's position (p_i^t) and the new velocity vector $(v_j^{(t+1)}$ as follows:"

$$p_j^{(t+1)} = p_j^t + v_j^{(t+1)}. \quad (3)$$

All previous operations are repeated until the stopping criterion is satisfied. ”

Algorithm 1 PSO algorithm

```
for each particle do
    Randomly initialize particle's position and velocity
end for
repeat
    for each particle do
        Compute fitness (FV)
        if Current fitness is better than the best fitness in particle's memory then
            Take current particle position as personal best position (PBest)
        end if
    end for
    Take position of particle whose best fitness as global best position (GBest)
    for each particle do
        Update particle velocity using Equation (1)
        Update particle position using Equation (3)
    end for
    Update the inertia weight using on Equation (2)
until stopping criterion is satisfied
```

Proposed Approach

The proposed algorithms are based on PSO for finding the optimum centroid in a data set for all target groups. In Proposed Algorithm the position and velocity of each particle are encoded as vectors as follows:”

$$\vec{p}_j = \left\{ p_j^{c_1}, p_j^{c_2}, \ldots, p_j^{c_i} \right\} \quad (4)$$

$$\vec{v}_j = \left\{ v_j^{c_1}, v_j^{c_2}, \ldots, v_j^{c_i} \right\} \quad (5)$$

Here in equation (4) and equation (5), $p_j^{c_i}$ and $v_j^{c_i}$ are “the position and the velocity vector for particle j’s class label c_i, respectively, which are represented in an N-dimensional space as follows:”

$$p_j^{c_i} = \{p_1, p_2, \ldots, p_n\} \quad (6)$$

$$v_j^{c_i} = \{v_1, v_2, \ldots, v_n\} \quad (7)$$

Here in equation (6) and equation (7), p_n and v_n are “the real values of the position and velocity, respectively, for dimension n.

In addition, “each particle has the following attributes:”

- First is Particle Identification Number (ParticleID)
- Second is Current Fitness (FV)
- Third is the Best Centroids Vector which has been discovered till now during the journey of particle (PBest)
- Fourth and last is the Best fitness which has been discovered till now during the journey of particle.

Furthermore, “our proposed algorithm keeps track of the best centroids vector (GBest) and the best group fitness which are achieved by any particle.

To evaluate “the fitness of each particle, two fitness functions are used. The first fitness function (F1) computes the fitness of particle j using. ”

Equation (8) by getting "the average of the sum of all Euclidean distances (Equation (9)) between the current centroid vector of class label c_i ($\vec{p}_j^{\,c_i}$) and the data instances that belong to class label c_i according to the training data set. "

$$F1(j) = \frac{1}{D}\sum_{i=1}^{D} d\left(\vec{x}_i, \vec{p}_j^{\,c_i}\right) \tag{8}$$

Here in equation (8) D is "the number of data instances in a training data set, and $\vec{x}_i$ is Data instance in the Training Dataset. It should be noted that each instance of data for each dimension (function) is standardized within [0.0, 1.0] using standardization and min-max that the sum of such distances is divided by N (total number of characteristics in the data series), and thus the measured distance falls within [0.0, 1.0]. "

$$d(\vec{a}, \vec{b}) = \sqrt{\sum_{k=1}^{n}(a_k - b_k)^2} \tag{9}$$

For "the second fitness function F2, particle fitness j, which is a linear combination of two fitness values, is determined as follows"

$$F2(j) = \frac{1}{2}(F1(j) + F\psi(j)). \tag{10}$$

The "first physical condition is determined using the first function of physical condition F1 (Equation (8)) and the second physical condition is measured using the physical function F premises. In two steps, F calculates fitness. During the first step the nearest centroid is allocated to all data instances in a training data set. The second stage measures the percentage of instances of a training data set incorrectly rated"

$$F\psi(j) = \frac{1}{D}\sum_{i=1}^{D}\delta(\vec{x}_i) \tag{11}$$

$$\delta(\vec{x}_i) = \begin{cases} 1 & \text{if } c_{\text{predicted}}(\vec{x}_i) \neq c_{\text{actual}}(\vec{x}_i) \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

Where *D* is "the number of data instances in a training data set, c_{actual} is the actual outcome of *xi*, and $c_{\text{predicted}}$ is the predicted outcome of *xi*.

To apply the classification function in a broad data set, it is important to adjust updating the location of the particles and evaluating the aptitude in all iteration. These two operations are conducted in two stages in Proposed Algorithm. The controller program sends the tasks to the performers in the first step. Additionally, all particles are sent to the performers through the cluster administrator via a diffusion vector. Then, each executor reads a portion of the encapsulated data records in an RDD. The executor's pseudocode for evaluating the physical functions F1 and F2 is shown in algorithms 2 and 3, respectively. It should be remembered here that the executors read only once a part of the data instances and cache them for subsequent iterations."

Algorithm 2 Fitness function 1 evaluation

```
for each data instance x in RDD do
    for each particle j in Swarm do
        - Extract centroid vector p from particle j which belongs to the same class label of x
        - Compute the distance between x and p using Equation (9)
        - Add ParticleID and the distance to accumulator Accumulator_F1
    end for
end for
```

In Algorithm 2, "the executor extracts the data instance vector RDD x and the position vector p that belongs to the same class mark as the particle x. Then the distance from p to x is determined using equation (9). Next, the accumulator variable AccumulatorF1 is added to ParticleID and calculated distance. In all RDD instances this cycle is replicated. "

```
Algorithm 3 Fitness function 2 evaluation
  for each data instance x in RDD do
    for each particle j in Swarm do
      // FV1
      - Extract centroid vector p from particle j which belongs to same class label of x
      - Compute the distance between x and p using Equation (9)
      - Add ParticleID and distance to accumulator Accumulator_F1

      // FV2
      - Assign x to closest centroid
      if Assigned_Class(x) != Actual_Class(x) then
        - Add 1.0 and ParticleID to accumulator Accumulator_F2
      end if
    end for
  end for
```

In Algorithm 3, "the executor extracts the data instance vector x from the RDD and calculates its suitability in two steps. The first step is the same as the previous one, and the results are added to the accumulator vector AccumulatorF1. The data instance x is allocated in the second step to the class tag whose location is closest to the data instance and then the predicted class tag is compared to the x class tag. If the predicted class tag is not equivalent to the real class tag, the value is 1.0 and the AccumulatorF2 accumulator variable is applied to ParticleID. In all RDD instances this cycle is replicated."

After the performers have finished their work, the second stage begins by sending variables of the AccumulatorF1 and/or AccumulatorF2 to the controller program to update the physical fitness FV and the particle position vector p.

```
Algorithm 4 Driver program
  Find number of instances in training data set D
  for each particle j in Swarm do
    if Fitness_Function = F1 then
      - Extract a value V of particle j from Accumulator_F1
      - Update FV of the particle j using Equation (8)
      - Update PBest if necessary
    else if Fitness_Function = F2 then
      - Extract a value V1 of particle j from Accumulator_F1
      - Extract a value V2 of particle j from Accumulator_F2
      - Update FV of particle j using Equation (10)
      - Update PBest if necessary
    end if
  end for
  - Take position vector of particle whose best FV value as GBest
  for each particle j in Swarm do
    - Update particle velocity using Equation (1)
    - Update particle position using Equation (3)
  end for
```

The controller pseudocode part is shown in Algorithm 4.

The operations above shall be replicated until the full number of iterations is reached. Every instance of data in a test dataset is then assigned to the class tag whose centroid is nearest by equation (9).

Conclusion

In this work, we have planned and carried out a variant of particle swarm optimization that can works in parallel utilizing the Apache Spark structure to conquer the failure of PSO classification when dealing with large scale data sets. The Proposed Algorithm utilizes the essential PSO technique to get the ideal centroid for each target label in a dataset and afterward allots each unlabeled data set in a testing dataset to the nearest centroid. Two variations of Proposed Algorithm, Proposed Algorithm - F1 and Proposed Algorithm - F2, have been proposed depending on distinct fitness functions.

Our future work aims to implement the proposed algorithms on Apache Spark framework and after that we will try to apply the Proposed Algorithms on the real-world applications with very large data sets (terabyte size) using hundreds of nodes and conduct comprehensive experiments to investigate the utilization of the resources used.

References

1. Gao, S., Li, L., Li, W., Janowicz, K. and Zhang, Y., 'Constructing gazetteers from volunteered big geo-data based on hadoop', Computers, Environment and Urban Systems 61, pp.172–186, 2017.
2. Kaba´c, M., Consel, C. and Volanschi, N., 'Designing parallel data processing for ˇ enabling large-scale sensor applications', Personal and Ubiquitous Computing, pp. 1– 17,2017.
3. Zhao W, Ma H, He Q. Parallel *K*-means clustering based on MapReduce. Paper presented at: IEEE International Conference on Cloud Computing; 2009; Bangalore, India.
4. Ludwig SA. MapReduce-based fuzzy c-means clustering algorithm: implementation and scalability. *Int J Mach Learn Cybern*., pp. 923-934, 2015. https://doi.org/10.1007/s13042-015-0367-0.
5. Wang B, Yin J, Hua Q,Wu Z, Cao J. Parallelizing k-means-based clustering on Spark Paper presented at: International Conference on Advanced Cloud and Big Data (CBD); 2016; Chengdu, China.
6. Yang J, Li X. MapReduce based method for big data semantic clustering. Paper presented at: IEEE International Conference on Systems Man, and Cybernetics; 2013; Manchester, UK.
7. Banharnsakun AA. MapReduce-based artificial bee colony for large-scale data clustering. Pattern Recogn Lett., pp. 8-84, 2017. https://doi.org/10. 1016/j.patrec.2016.07.027
8. Tripathi AK, Sharma K, BalaM. A novel clustering method using enhanced grey wolf optimizer and MapReduce. Big Data Res., pp. 93-100, 2018, https:// doi.org/10.1016/j.bdr.2018.05.002.
9. Aggarwal CC. Data Mining: The Textbook. Berlin, Germany: Springer; 2015.
10. Wu X, Zhu X, Wu GQ, Ding W. Data mining with big data. IEEE Trans Knowl Data Eng. 2014;26(1):97–107.
11. Laney D. 3D Data Management: Controlling Data Volume, Velocity and Variety. 2001.http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-management-Controlling-Data-Volume-Velocity-and-Variety.pdf. Accessed July 2015.
12. Fernández A, del Río S, López V, Bawakid A, del Jesús MJ, Benítez JM, et al. Big data with cloud computing: an insight on the computing environment, mapreduce, and programming frameworks. Wiley Interdisc Rew Data Min Knowl Discov., pp. 380–409, 2014
13. Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters. In: OSDI 2004. San Francisco, CA; pp. 137–150, 2004.
14. White T. Hadoop, The Definitive Guide. Sebastopol: O'Reilly Media, Inc; 2012.
15. Apache Hadoop Project. Apache Hadoop. 2015. http://hadoop.apache.org/. Accessed December 2015.
16. Lin J. Mapreduce is good enough? if all we have is a hammer, throw away everything that's not a nail! Big Data, pp. 28–37, 2012.
17. Karau H, Konwinski A, Wendell P, Zaharia M. Learning Spark: Lightning-Fast Big Data Analytics. Sebastopol: O'Reilly Media; 2015.
18. Spark A. Apache Spark: Lightning-fast cluster computing. https://spark.apache.org/. Accessed December 2015.
19. Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI'12. San Jose; pp. 15–28, 2012.
20. InfoWorld. Apache Flink: New Hadoop contender squares off against Spark. 2015. http://www.infoworld.com/article/2919602/hadoop/flink-hadoops-new-contender-for-apreduce-spark.html. Accessed December 2015.

❖◆❖