Attribute Reduction:An Implementation of Heuristic Algorithm using Apache Spark

Pratik Ghodke[1], Vishesh Shah[2], Sani Asnain Mulla[3], Aman Pawar[4],Prof. Bhushan Pawar[5]

[1,2,3,4] Students, [5]Assistant Professor
Department of Computer Engineering, Savitribai Phule Pune University, India

***

Abstract - *Most weather event occurs in the troposphere, the lowest level of atmosphere. Weather describe the degree of the condition as hot or cold, minimum and maximum temperature, clear or cloudy, atmospheric pressure. This degree changes at instant time due to atmospheric condition. Each degree can be considered as attribute in this data which is important, but many attributes within this data are redundant and unnecessary. The structure of Weather Forecast data is complex and consists of structured data, unstructured data and semi-structured data. However, many attribute reduction algorithms are time-consuming.*

To solve this problem, we are using the methodology of rough sets to build Weather Forecast knowledge representation system. By utilizing good benefits of in-memory computing, an attribute reduction algorithm for weather forecast using Spark is proposed. In this algorithm, we are using a heuristic formula for measuring the importance of the attribute to reduce search space, and a suitable algorithm for simplifying weather forecast decision table, which improves the computation power.

Key Words: Big Data, Spark, Resilient Distributed Data, Rough Set Theory

1. INTRODUCTION

Weather forecast data plays very important role in the development of the nation economy that based on natural resources.

As the weather data is collected from various sources, the attributes in the data may be redundant and also may not be useful to predict the correct climate of a particular area at instant time. Such attributes may increase the running cost and decreases the performance of the data mining algorithms. So to make the algorithm more powerful it is necessary to remove unimportant attributes and select only effective attribute.

To solve this problem many attribute reduction algorithms are introduced but are time consuming. Rough set theory is used from extracting the vital information. The features and advantages of the Spark are useful to determine attribute reduction algorithms to compress the data by removing redundant attributes which will be useful in decision making and reduce the time complexity for processing the algorithm. The heuristic function used studies the importance of attribute to reduce search space of algorithm.

2. THEORETICAL BACKGROUND

[1] Pawlak in 1982 proposed Rough Set Theory, the most high-powered mathematical technique applied to handle the blurriness and ambiguity of data. It demonstrates that the rough set technique can be used for blend and examine the approximations in the distributed context.

[2] M. Zaharia proposed that Resilient Distributed Datasets is used for describing a programming interface that efficiently provides fault tolerance.

[3] M. R. Anderson and M. Cafarella proposed a data-centric system called Zombie, which speed ups feature engineering with the help of quick-witted input selection, which optimize the inner loop of the process.

[4] J. Zhang et.al. have proposed three different parallel matrix-based techniques to actions on large-scale, imperfect data. They are constructed on MapReduce and performed on Twister, a lightweight runtime system.

[5] J. Zhang et.al. Proposed a parallel approach for large-scale attribute reduction based on Hadoop MapReduce. This method analysis shows that these parallel modules are powerful and efficient for big data. The classification accuracy is enhanced by this module and processing is done faster.

2.1 SPARK

A lightning-fast cluster computing technology known as Spark is used for fast computation. It computes interactive queries and stream processing using extended MapReduce. In-memory cluster computing, is the main trait of Spark speed-ups the processing of an application. Main Components of Spark are GraphX, MLib, Spark Streaming and Spark SQL.

The fundamental data structure of Spark is Resilient Distributed Datasets (RDD). RDD is a fault-tolerant assembly of elements, which is works in parallel. It is inexhaustible. Each dataset is divided and computed on different cluster nodes. RDDs support Python, Java or Scala objects, also for user-defined classes.

3. ALGORITHM

Based on notation given by Rough Set Theory, a weather forecast decision table can be illustrated as W = (U, A, V, f).

U: is the non-empty set of all weather forecast data and can be depicted as U= {w_1, w_2, w_3,……, w_n}.

A: is the non-empty set of attributes associated with the forecast data and can be depicted as A= C U D, where the elements in C are conditional attributes and elements belongs to D are decision making attributes for the records of weather forecast data.

V: it gives the value of the attributes present in A.

f: it is a mapping function used to get a value V associated with a single record from set U.

POS: is the positive region used to classify new elements from the previously classified elements of set U

Module 1

Input: Weather Forecast Decision table.

Output: A consistent weather forecast table which can be mathematically visualized as

W'= (U', A', V', f').

Steps:

1. Load the given/considered weather forecast decision table using appropriate function /method /API given by the language. (For e.g. spark .text file () method in Apache Spark).
2. Classify all the elements of U based on conditional attributes of A such that each equivalent class E_i is mapped as a key to a value V_i as an id of records present in U.
3. Find the most frequent value from an equivalent class E and add it to the new Forecast Decision Table W'.
4. Stop.

Module 2

Input: W'= (U', A', V', f').

Output: Resultant reduced attributes.

Steps:

1. Initialize a temporary attribute reduction set.
2. Execute Module 3 to calculate equivalent class E for each attribute a є A'.
3. Execute Module 4 to compute attribute significance and POS of the considered attribute.
4. Select the best significant attribute from A' and add it to the resultant attribute set.
5. Eliminate the record from U' associated with the selected attribute 'a' and update U'.
6. Use Module 1 to transform the updated weather forecast decision table.
7. Check if U' is empty
   If (U' is empty)
      Display the resultant attribute set.
   Else
      Repeat from step 3.
8. Stop.

Module 3

Input: Resultant attribute set.

Output: Equivalent Class E'_i.

Steps:

1. Classify all the elements of U' based on conditional attributes of A' such that each equivalent class E'_i is mapped as a key to a value V'_i as an id of records present in U'.
2. Stop.

Module 4

Input: Equivalent Class of the candidate attribute.

Output: Significance and POS of candidate attribute.

Steps:

1. Check the consistency of the candidate attribute.
   If (candidate attribute is consistent)
      Compute the POS of the attribute
   else
      Remove the equivalent class E'.
2. Compute the significance of the candidate attribute from the calculated POS.
3. Display significance and POS of the candidate attribute.
4. Stop.

4. METHODOLOGY

The rate of increasing data day by day in data age is tremendous. Complex data consists of structured data, un-structured data and semi-structured data generated for diverse system. Each attribute in this dataset is important, but many attributes within it are redundant and unnecessary. To reduce the attributes, algorithm for attribute reduction in Spark is used to minimize the data attributes. In-memory cluster computing, is the main trait of Spark speed-ups the processing of an application.

For better resource minimization and processing speed, Cluster Manager is used. An executor is assigned to each and every application, which is awake till the whole application is performed and multiple threads are used to run tasks. Spark is sceptic to the subordinate cluster manager.

The driver program should listen and accept connections received from executors during its lifetime. The driver program should be network available through the worker nodes.

The architecture first module performs the algorithm to get decision making for rough set theory of equivalence class decision table because attribute reduction cannot be performed on unsteady decision table. Then using module 2 heuristic function, we can reduce the time complexity of equivalence class and get reduce search space.

Module 3 chooses the best candidate from equivalence class and then it is added to attribute reduct. Every time candidate is added, attribute reduct is updated. Module 4 first reckons positive region, the module checks if decision attribute values are inconsistent or consistent. If attribute is inconsistent then the equivalence class will be deleted. Output of the intermediate result is stored.

5. EXPERIMENTAL ENVIRONMENT

To run parallel algorithms, we create a cluster of computers consisting four nodes. One computer serves as a master node, and the other three computers serve as slave nodes. They are connected with each other via an Giga-Ethernet. Detailed information regarding the configuration of software and hardware is described in Table I.

Table 1: Environment of Experiments

Resource	Description
Hardware	2 GHz processor, Network cards, 8GB RAM
Software	Ubuntu 14.04, Scala, Apache Spark

6. PROPOSED OUTCOMES

The outcome of the project will be stored in an equivalence class which can be accessed by the user. Output of the project represents the reduction in the attributes and data processing cost due to Spark which processes in-memory clusters computing.

The state of memory is stored as object which is shareable. The algorithm is capable of better load sharing and handle fault tolerance. The disk has replicated copies of RDD's available at each node.

REFERENCES

[1] Z. Pawlak and A. Skowron, “Rough sets: Some extensions,” Information Sciences, vol. 177, no. 1, pp. 28–40, 2007.

[2] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,” USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012, pp. 2–2.

[3] M. R. Anderson and M. Cafarella, “Input selection for fast feature engineering,” in 2016 IEEE 32nd International Conference on Data Engineering (ICDE). IEEE, 2016, pp. 577–588.

[4] J. Zhang, J.-S. Wong, Y. Pan, and T. Li, “A parallel matrix-based method for computing approximations in incomplete information systems” IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 2, pp. 326–339, 2015.

[5] J. Zhang, T. Li, and Y. Pan, “Plar: Parallel large-scale attribute reduction on cloud systems,” in International Conference on Parallel and Distributed