A Study on Efficient Memory Management Using Machine Learning Algorithm

Beom-Joo Park¹, Min-Soo Kang¹, Minho Lee²†, Yong Gyu Jung¹

¹Department of Medical IT Marketing, Eulji University, Korea
²Department of Food and Nutrition, Eulji University, Korea
qoawn9966@naver.com, mskang@eulji.ac.kr, †Minho@eulji.ac.kr, ygjung@eulji.ac.kr

Abstract

As the industry grows, the amount of data grows exponentially, and data analysis using these serves as a predictable solution. As data size increases and processing speed increases, it has begun to be applied to new fields by combining artificial intelligence technology as well as simple big data analysis. In this paper, we propose a method to quickly apply a machine learning based algorithm through efficient resource allocation. The proposed algorithm allocates memory for each attribute. Learning Distinct of Attribute and allocating the right memory. In order to compare the performance of the proposed algorithm, we compared it with the existing K-means algorithm. As a result of measuring the execution time, the speed was improved.

Keywords: Data Mining, Weka, Machine Learning, Clustering, Memory Allocation, Efficient Memory

1. Introduction

The clustering algorithm is an algorithm for grouping similar data points (hereafter referred to as "points") according to the distribution of data, and distributes the given d dimension points to k clusters each, while the points belonging to the same cluster have similar properties and have different clusters. The points to which they belong have different attributes[1]. The K-means algorithm is the most widely known clustering algorithm [2] in which each cluster is represented by a center, the mean value of the points in the cluster. In particular, the first iteration uses the given initial emphasis Cinit [3]. Each iteration assigns each point to the closest midpoint and iteratively performs two steps to compute a new midpoint, which is the average of the points assigned to the cluster[4]. If the point is no longer changed, stop the execution and return the result. Most of the execution time of the K-means algorithm is used to calculate the distance between n points and k points [5, 6].

With the rapid development of computers and the Internet, the amount of information has been explosively increased, and existing machine learning algorithms are suffering from performance degradation in processing such vast amounts of information [7].

In this paper, we propose a method to improve performance degradation in large scale data. Data Distinct
of each attribute is learned, and the number of Distinct is limited by the computer itself, and memory is allocated to each Attribute. The proposed algorithm improves the processing speed of the data compared to the conventional one through efficient memory utilization.

2. Main Subject
The efficient memory allocation process proposed in this paper consists of preprocessing, memory allocation, and applying the existing machine learning algorithm as shown in Figure 1. In this paper, we use the K-means algorithm, which is the most commonly used partitioning clustering algorithm among existing machine learning algorithms. In the preprocessing step, the attribute of data and the ratio of Dist of each attribute are obtained. Allocate memory for each attribute based on the ratio of preprocessed Dist. The existing K-means algorithm is executed with optimized memory for each attribute.

2.1 WEKA API
WEKA is a software for data mining developed in Java language at Waikato University, New Zealand, which enables various machine learning algorithms to be performed [8, 9]. WEKA can be executed by Command Line, GUI environment and user JAVA code. It includes typical algorithms of machine learning such as data preprocessing, regression, clustering, etc [10]. All algorithms in WEKA are public APIs and users can analyze and edit GUI, algorithm using WEKA API. In this paper, clustering processing speed is measured by using Simple K-Means algorithm in WEKA.

2.2 Experimental data
In this paper, we use the data of school companies in 2016. Data has 8 attributes including order, establishment type, school grade, sales amount, operating profit, net profit area distribution, industry type distribution. In this paper, data are clustered into a total of five clusters.

3. Algorithm through efficient memory utilization
3.1 Preprocessing
In the preprocessing step, Distinct is obtained for each attribute of each data. The smaller the Distinct, the smaller the memory required for the task. That is, the size of Distinct is directly proportional to the amount of memory required for the operation. WEKA API to perform the data preprocessing process yields the
results shown in Table 2. The name of each attribute is specified and the type of each attribute is nominal or numeric. The part to be used in this study is the Dist of each property displayed on the right side. Among the experimental data of this study, the no attribute is expected to be the largest Dist, which will require the most memory. Type of establishment attribute and Regional distribution attribute are expected to take the least amount of memory since they have the smallest Dist. To allocate memory according to the Dist of each attribute, preprocessing is performed through the memory allocation process after the predecessor to obtain the Dist.

### Table 1. Data Preprocessing

| Name          | Type | Nom | Int | Real | Dist |
|---------------|------|-----|-----|------|------|
| no            | Num  | 0%  | 100%| 0%   | 195  |
| Typeofestabli | Nom  | 100%| 0%  | 0%   | 2    |
| Schoolgrade   | Nom  | 100%| 0%  | 0%   | 3    |
| SalesAccount  | Num  | 0%  | 98% | 2%   | 181  |
| OperatingProfit| Num | 0%  | 100%| 0%   | 188  |
| NetIncome     | Num  | 0%  | 100%| 0%   | 176  |
| RegionalDistribution | Nom | 100%| 0%  | 0%   | 2    |
| TypeofBusiness| Nom  | 100%| 0%  | 0%   | 9    |

#### 3.2 Memory allocation

Allocate memory for each attribute before executing the existing algorithm. The memory is adjusted to the ratio of the specific attribute among the number of all distincts and assigned to the task. Creates an attribute of the dataset as an object. For each created object, allocate it according to the ratio of Dist based on Max Memory. In this paper, we propose an algorithm that maximizes the performance by reducing the work time by efficiently utilizing the memory of the computer. The proposed algorithm is represented by the pseudo code in Figure 3.

If you apply the algorithm proposed in this paper, the amount of memory in use increases compared with the existing algorithm. By increasing the utilization of memory, work time is reduced.

```plaintext
Memory Allocation Process
1: for (int i = 0; i < dataset.num (); i++) {    // Repeat the size of dataset
2:   distRate = (dataset.DistinctValues(i) / distSum)    // Distinct ratio of each attribute
3:   newAttribute = new byte(maxMemory() * distRate)    // Declare newAttribute by allocating memory for each Distinct ratio
4:   newAttribute = data.attribute(i)    // Insert existing Attribute into declared newAttribute
5:   replaceAttribute (newAttribute, i)    // Replace existing Attribute with newAttribute
6: }
7: buildClusterer(data)    // Create a cluster with data that has undergone memory allocation process
```

#### Figure 2. Memory Allocation Process

#### 3.3 Clustering

And applied to the clustering algorithm using the calculated memory ratio. In order to evaluate the performance of the proposed algorithm, we used K-means, which is the most commonly used partitioning clustering algorithm, to create clusters.

#### 4. Experiment result

Experimental configuration of this paper was tested on Intel Core i5-4210M CPU, 8.00GB Memory and Microsoft Windows 10 Home Premium K 64bit operating system. Experimental environment was tested
with Eclipse Java IDE Mars Release (4.5.0) Version, version 1.8.0_77. Experimental data used data from 2016 school company survey, including 8 attributes and 195 Instances.

This experiment compares the generation time of the K-means clusters generated by the existing data and the K-means clusters generated by the data through the proposed memory allocation process. Considering the influence of external factors, we increased reliability by comparing 1000 cluster creation times, 600 cluster creation times, and 200 cluster creation times in total 10 times.

Table 2. Comparing the time to create a cluster

|                  | Create 200 clusters | Create 600 clusters | Create 1,000 clusters |
|------------------|----------------------|----------------------|-----------------------|
|                  | 1st  | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th | 9th | 10th | Average time |
| Normal Data      | 7.4  | 6.9 | 6.6 | 7.1 | 6.2 | 6.9 | 7.1 | 6.4 | 6.5 | 7.4  | 6.85         |
| Allocated Data   | 6.6  | 6.4 | 6.4 | 6.6 | 6.2 | 7.2 | 5.8 | 5.7 | 6.2 | 6.4  | 6.35         |
|                  | 1st  | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th | 9th | 10th | Average time |
| Normal Data      | 18.65| 18.62|18.01|17.59|18.55|17.99|17.67|18.22|18.04|18.95|18.23        |
| Allocated Data   | 16.88| 16.24|16.25|15.48|15.99|16.17|16.19|16.04|16.04|17.11|16.24        |
|                  | 1st  | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th | 9th | 10th | Average time |
| Normal Data      | 32.6 | 34.7 |33.1 |34.2 |34.4 |33.1 |32.9 |33.8 |34.1 |33.7 |33.66        |
| Allocated Data   | 30.1 | 31.3 |29.8 |30.7 |30.6 |29.8 |30.5 |29.6 |31.4 |30.6 |30.44        |

Table 2 shows the average generation time by measuring the cluster generation rate 10 times through the data with and without memory allocation. As a result of comparing the generation time of each cluster, the average generation speed was slightly changed from 6.85 sec to 6.35 sec with a speed improvement of about 7% in 200 generation. However, when the cluster creation time of 1,000 is compared, the average processing speed of the normal data that is not allocated memory is measured to be 33.66 sec. The average processing speed of the Allocated Data allocated memory according to the Distinct ratio is 30.44 sec Respectively. As a result, the proposed system showed about 10% speed increase. For visual comparison, each cluster creation time is shown in the graph in Figure3.

![Figure 3. Time graph to create cluster](image)

5. Conclusion

In this paper, we propose a method to quickly apply a machine learning based algorithm through efficient resource allocation. For each data, Distinct of Attribute is learned and allocated memory. In order to compare
the performance of the proposed system, we compared it with the existing K-means algorithm. As a result of measuring the execution time of the memory allocated data through the existing data and the proposed system, the speed improvement is about 10%. The algorithm proposed in this paper can be used to improve the speed of a machine learning based algorithm that analyzes and processes a large variety of data.

References

[1] Woosaeng Kim, Sooyoung Kim. (2014.3). Document Clustering Technique by K-means Algorithm and PCA. Journal of the Korea Institute of Information and Communication Engineering, 18(3), 625-630.

[2] indong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. 2007. Top 10 algorithms in data mining. Knowl. Inf. Syst. vol.14, no.1, pp.1-37, Dec. 2007.

[3] S. Lloyd. Least squares quantization in PCM. IEEE Transactions on Information Theory, vol.28, no.2, pp.129-137, 1982.

[4] Tae-Chang Jee, Hyun-Jin Lee, Yill-Byung Lee. (2009.1). Fast K-Means Clustering Algorithm using Prediction Data. JOURNAL OF THE KOREA CONTENTS ASSOCIATION, 9(1), 106-114.

[5] G. Hamerly. Making k-means even faster. In SIAM International Conference on Data Mining (SDM), 2010.

[6] Taesik Yoon, Kyuseok Shim, Journal of KIISE : Computing Practices and Letters 18(1), 2012.1, 55-59 (5 pages).

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, “The WEKA Data Mining Software: An Update,"SIGKDD Explorations, vol. 11, no. 1, pp.10 18 2009.

[8] Williams, N., 2006 , Evaluating machine learning algorithms for automated network application identification.

[9] J. W. Kim, “Improving Artificial Intelligence Lecture using WEKA Tool,"Proceedings of KIIS Fall Conference, vol. 22, no 2, pp. 170 171, 2012.

[10] Ik-Hyun Youn, Kwang-hee Won, Jong-Hoon Youn, Jeremy Scheffler. (2016.3). Wearable Sensor-Based Biometric Gait Classification Algorithm Using WEKA. Journal of information and communication convergence engineering, 14(1), 45-50.