Data Compression and Prediction Using Machine Learning for Industrial IoT

Junmin Park* Hyunjae Park and Young-June Choi
Department of Software*, Computer Engineering, Ajou University
Suwon, Republic of Korea
jun.m.park0@gmail.com*,{estancia, choiyj}@ajou.ac.kr

***Abstract*—Industrial IoT generates big data that is useful for getting insight from data analysis but storing all the data is a burden. To resolve it, we propose to compress the industrial data using neural network regression into a representative vector with lossy compression. For efficiency of the compression, we use the divide-and-conquer method such that the industrial data can be handled by the chunk size of data. Through our experiments, we verify that industrial data is represented by a function and predicted with high accuracy.**

Keywords—industrial data; big data; data compression; machine learning; regression

I. INTRODUCTION

In industrial Internet-of-Things (IoT), each sensor produces a large amount of data requiring huge storage to store. Even though the price of storage decreases, keeping all the data is not an appropriate policy. To resolve this problem, compressing the data into a single equation can be a solution to achieve the efficiency of memory and of data processing. In addition, if data can be represented only by one formula, future data can be predicted according to the input value. This is possible because of the characteristic of industrial data where the data is produced periodically over time. This is also because data of another period can be predicted by regression of data coming out periodically. We adopt an approach of using neural network regression [4] as compression and use some combining techniques to achieve more accurate data representation and prediction.

To do so, we propose to vectorize the industrial data through the neural network regression. However, it is inefficient to vectorize the whole data. We hence suggest a new method that exploit the trend of data through vectorization in compressing data by dividing the whole data according to a certain range, vectorizing the divided chunks and combining them. The neural network regression is not enough to compress the industrial data generated on an hourly basis. So we compress the data using the divide-and-conquer method. We divide the data generated by the divide process into time units, and apply neural regression to each of them. In conquer, we apply various machine learning techniques and choose the method with the highest accuracy. In addition, we apply additional techniques with the highest degree of similarity to the original formula.

II. BACKGROUND

A. Loss function

This is a function to obtain the error rate through the RMSE method. In the same process as the existing RMSE method, all the distances between the points and the straight line are obtained, and the sum of the squares is added and the square root is added.

B. Coefficient Averaging

This is a simple calculation method that averages all the coefficients and determines the coefficients of the new equation. In order to generate a new vector through multiple vectors, we use a method to add all the vector components and divide them by the number to determine the components of the new vector.

C. Euclidean distance

If only the entire linear regression is performed once, a lot of outliers are generated to represent the data, and many trade-offs occur to represent the entire data. Therefore, it is necessary to perform the linear regression separately for each range after cropping by dividing the range and combining them. We have tried various techniques for combining these techniques, one of which is Euclidean distance. The formula for calculating Euclidean distance is as follows:

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

where $P=(p_1, p_2, \ldots, p_n)$, $Q=(q_1, q_2, \ldots, q_n)$ are two points.

Using this formula to determine the distance between points in the results of linear regression, the processor can find the distance and proceed to minimize it. After then, this distance is minimized for each point of the raw data. The combination of formulas can represent the entire data. However, this method has a disadvantage in that only the size value is considered and the direction value is not considered.

D. Cosine Similarity

In our study, cosine similarity is used in two places. First, it is used as one of the machine-learning techniques to combine data expressions in the conquer process. Second, it is used as a technique to compare the original formula with the final formula vector to determine the similarity of the original formula with the final formula after all the divide and conquer procedures. It

Authorized licensed use limited to: University of Dayton Libraries. Downloaded on July 20,2020 at 21:18:04 UTC from IEEE Xplore. Restrictions apply.

is based on the second method, which is used to determine the accuracy of the result for general purpose, because the process is the same but the purpose is different.

Cosine Similarity [3] was used as one of the other ways of joining part-by-part straight lines. This is a formula for finding the similarity between straight lines. The formula is as follows:

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

where A_i, B_i are vector attributes of A and B.

Using this formula, we create a regression line that is most similar to the range-specific representation lines. We use machine learning for the resulting straight line to minimize cosine similarity to all linear regressed lines. In contrast to Euclidean distance, there is a disadvantage in that only the direction value of the vector is considered and the size value is not considered.

III. Proposed Data Compression Scheme

In the conventional compression method, the entire data is compressed or maintained by a lossy or lossless function. In the process of vectorizing data, however, a method of vectorizing all the data at once is inefficient, because this would take quite a while. Therefore, we propose a method to vectorize chunks divided into a certain range and combine them into one vector after all of them are vectorized. Through this, the data in the future can be predicted through the tendency of vectors. This is done through the divide-and-conquer method, and the process of combining the range-specific vectors into a single vector is done in the conquer process through machine learning techniques.

Here are the reasons for using machine learning techniques in the conquer process. As a result of performing regression by range, the data is represented by a straight line for each range. With these alone, it is difficult to represent the data, because they need to have a range for each straight line to represent the data, which is inefficient. Therefore, it is necessary to be able to express all the data in one straight line by combining all of these ranges. We use four techniques as follows.

1. Coefficient Averaging: the coefficients of the straight lines obtained by performing regression are averaged to determine the coefficient of the final straight line. This is simply the most convenient way to sum all the coefficients and divide them by the number, and it does not guarantee the accuracy as much.

2. Euclidean Distance: this is a method of finding the final straight line by determining the degree of similarity and learning by using the distance calculation method between Euclidean distances. The degree of similarity of the regression results by range is distinguished and the machine learning is done by the method of maximizing it.

3. Cosine Similarity: each regression result for each range is straight and therefore has directionality. So we can obtain the cosine similarity by calculating the direction as a vector. We generate a new arbitrary straight line and compute the similarity between the straight line and each regression result. After that, machine learning is performed in order to make the similarities to the greatest, and the final overall straight line is obtained.

4. Re-Learning: this is a way to give machine learning once more to the straight lines where machine learning has progressed. Since we cannot learn straight lines, we have to create new points. Therefore, the data is regenerated by the number of original data on each regression result, to be ready. Finally, we perform regression for all data and extract the final straight line.

With these techniques, the data is represented in a single, final line. Of course, as lossy compression, it cannot recover raw data. However, in the case of industrial data the values of each timestamp is continuously produced and the trends of values are more important than a value of specific time without extraordinary value. So it is useful to exclude outliers as much as possible and to maximize the data according to the original tendency. As a result, all of the data is compressed as one equation, thus reducing memory waste and making predictions of future data.

IV. Experiments

Experiments to find the most efficient expression of the data were conducted in three ways. 1. **Find type of regression:** there are various kinds of regression. Among them, we examined what kind of regression should be performed to perform the regression according to the industrial data characteristics. 2. **Compress the data using the proposed four techniques:** using the four techniques described above, we studied whether data can be represented most accurately when combining regressed formulas by range. 3. **Merge:** we have looked at whether it would be more effective to randomly select attributes for each range rather than just merge them together when combining the data by range.

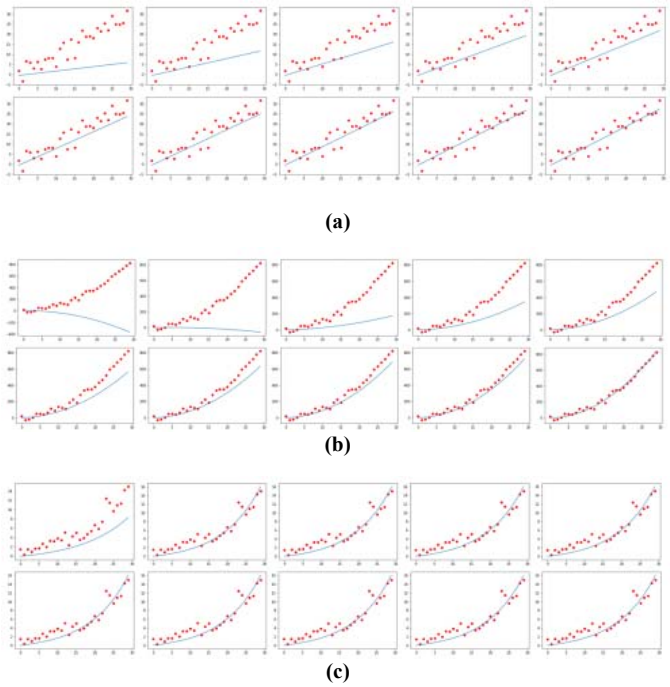

Figure 1. Type of regression: (a) Linear Regression, (b) Polynomial Regression, (c) Exponential Regression. Regression was performed on the sample dataset, and it was learned by gradient descent optimizer method during 1000 steps and output every 100 steps. It shows that linear regression is the best fit for the data.

A. Regression

First, we decided what kind of formulas should be used to express the data formulas. Because we want to use regression to

Authorized licensed use limited to: University of Dayton Libraries. Downloaded on July 20,2020 at 21:18:04 UTC from IEEE Xplore. Restrictions apply.

express it, we need to pre-set the shape of the final formula we want. Therefore, we had to set the number of dimensions and variables. In order to see what shape is most efficient, we tried several regression types: linear, polynomial, and exponential. These three types are the representative types of regression, and the type of expression changes according to this type. The exponential of this experiment was difficult to control the learning rate because the rate of increase was exponential as seen in Figure 1.

We looked for ways to increase accuracy by using additional techniques before representing data in a single final formula. So we used a little extra technique in the conquer process. In order to combine these data after the data classified by the range is represented by each regression, we have used the corresponding range of formulas in the previous procedures. We have studied whether the result will be higher if the formula of the surrounding range is used instead of the range. When randomly selecting an attribute, it is recommended to select from five ranges in front of and behind the specified range in order to prevent the data from becoming inconsistent due to excessive random selection.

B. Evaluation

We used two evaluation methods: Loss Function, and Cosine Similarity. The Root Mean Square Error (RMSE) was used as a loss function to give more weight to outliers. The difference between the regression line and all data can be squared, summed, and averaged to obtain a loss function. Also, cosine similarity was used to determine how well the data was represented. The accuracy was determined by calculating the cosine similarity between the formula based on the originally produced data and the newly obtained formula.

C. Results and Analysis

TABLE I. TABLE 1 RESULT OF REGRESSION

	Coefficient Averaging	**Euclidean Distance**	**Cosine Similarity**	**Re-Learning**	**Function**
Just Merge (Linear Regression)	0.242980	0.413407	*	0.501171	loss
	0.967321	0.964932	0.961557	0.932541	Similarity
Just Merge (Polynomial Regression)	*	*	*	*	Loss
	0.963415	0.960014	0.962280	0.932541	Similarity
Random Selection (Linear Regression)	0.225185	0.309426	*	*	Loss
	0.960173	0.963117	0.963084		Similarity
Random Selection (Polynomial Regression)	*	*	*	*	Loss
	0.950426	0.958461	0.961966	*	Similarity

** means a value that can not be calculated.*

We excluded exponential regression which is difficult to obtain learning rate and learning ability through previous experiments. As a result, the linear regression and polynomial regression tests were performed. As shown in Table 1, the linear regression performance was better. Using polynomial regression, it was found that learning was harder to predict due to radical occurrence, and that simpler linear regression was more efficient.

In addition, the cosine similarity technique was trained using only a vector, i.e, a direction value, so that even if the constant value is largely deviated, the similarity is correctly displayed but the loss function value cannot be found. Finally, in the re-learning technique, the meaning of selecting an attribute at random is lost. So we did not apply random selection in re-learning. Table 1 shows the result of regressions that applies for all these situations. As a result, sometimes the values come out biased and it is needed to do normalization. However, the necessity of normalization has been eliminated by using a loss function to find the distance between the regressed line and the data and a similarity to calculate only the vector. Because the purpose of normalization is to create a basis, but the straight line itself is the basis for the straight line.

In the divide-and-conquer process, four machine learning techniques were used in the conquer process. The results are shown in Table 1. As a result, it shows that the Euclidean Distance of the overall results is generally high and the applicable range is wide enough to be an efficient compression technique. The other techniques are degraded due to some limitations.

V. CONCLUSION

In this paper, we proposed data compression techniques by machine learning. In particular, we have presented several techniques for compressing and predicting vast amounts of industrial data. As a result, it is an efficient data compression method to represent data by linear regression, use divide-and-conquer method, divide data by time, and use machine learning technology using Euclidean Distance in conquer process. Also, with additional techniques, we tried to randomly select a value around the range for conquer, but it was not affected. Although loss compression cannot be recovered as original data, it is expected that future data is predictable by expressing industrial data as a single equation through this process and utilizing the tendency.

ACKNOWLEDGMENT

This research was supported by the MSIP (Ministry of Science and ICT), Korea, under the National Program for Excellence in SW) supervised by the IITP (Institute for Information & communications Technology Promotion) (2015-0-00908).

VI. REFERENCES

[1] Jin Xu, ShinJae Yoo, John Heiser and Paul Kalb, "Sensor Network Based Solar Forecasting Using a Local Vector Autoregressive Ridge Framework," unpublished.

[2] Anmol Jyot Maan, "Analysis and Comparison of Algorithms for Lossless Data Compression," ISSN 0974-2239 Volume 3, Number 3 (2013), pp. 139-146.

[3] Singhal, Amit (2001). "Modern Information Retrieval: A Brief Overview," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4): 35–43.

[4] Specht, Donald F. "A general regression neural network," IEEE transactions on neural networks 2.6 (1991): 568-576.

[5] Arijit Ukil, Soma Bandyopadhyay and Arpan Pal, "IoT Data Compression: Sensor-agnostic Approach," 2015 Data Compression Conference.

Authorized licensed use limited to: University of Dayton Libraries. Downloaded on July 20,2020 at 21:18:04 UTC from IEEE Xplore. Restrictions apply.