Majority Rule Approach to Deep Learning for Large Benchmark Data and Real Credit Card Transaction Data

Ayahiko Niimi
Faculty of Systems Information Science
Future University Hakodate
Japan

Abstract

In our previous study, we introduced two major applications for developing advanced deep learning methods that were focused on credit card data analysis. Herein, we validated our proposed methods by comparing benchmark experiments with other machine learning approaches. Through these experiments, we confirmed that deep learning exhibits accuracy similar to that of a Gaussian kernel support vector machine (SVM). Additionally, we validated the proposed methods using a large-scale transaction dataset. Motivated by our experimental results, we recently proposed three distinct quick approaches to deep learning for time-series data at the 2016 World Congress on Internet Security (WorldCIS-2016). The experimental results failed to demonstrate the effectiveness of our proposed method; however, through this work, we identified three key challenges associated with the proposed methods: (1) creating a model that generates an error rate less than 50%, (2) dividing the dataset evenly over time, and (3) considering the class not contained in the divided data. If a method can overcome these problems, then our proposed technique will be effective according to the ensemble learning theory. Herein, we apply our proposed majority rule approach to deep learning for large benchmark data and real credit card transaction data and discuss the experimental results.

1. Introduction

Deep learning is a state-of-the-art research topic in the field of machine learning, with applications that can potentially solve a wide variety of problems [1], [2]. Herein, we investigate the application of deep learning to credit card data analysis.

Credit card data are primarily used for the user and transaction judgments. User judgment determines whether a credit card should be issued to a user by discerning if particular criteria are satisfied, whereas transaction judgment determines whether a particular transaction should be considered valid [3]. Further, transaction data comprise imbalanced data, thereby proving them a difficult subject for classification. Some studies have focused on handling imbalanced data [4], [5]; however, to the best of our knowledge, handling imbalanced data via deep learning has not yet been to be studied. In our previous research, we identified the deep learning processes required to solve each of these problems and proposed appropriate methods for deep learning [6]-[8].

To verify the proposed methods, we compared our methods to benchmark experiments using other machine learning techniques, confirming that the accuracy of our deep learning approaches is similar to that of the Gaussian kernel support vector machine (SVM) method. In addition to describing our experiments and results, herein, we provide suggestions for future deep learning experiments. Although previously we only used a small transaction dataset for the evaluation experiment [9]. In a previous study, the proposed methods were also validated using a large transaction dataset [8]. Because we found that processing large volumes of data required for deep learning is time-consuming, we proposed three distinct approaches to quickly implement deep learning methods for time series data. Herein, we apply our proposed majority rule approach to deep learning for league benchmark data and real credit card transaction data, and discuss the experimental results.

The remainder of this paper is organized as follows. In Section 2, we introduce the characteristics of the credit card dataset. Next, in Section 3, we describe the deep learning approach. In Section 4, we discuss the data-processing infrastructure suitable for credit card data analysis. In Section 5, we present the three proposed methods to quickly implement of deep learning for time-series data. In Section 6 and 7, we describe our experiments and the corresponding results, respectively. Then, Section 8 discusses our considerations. Finally, in Section 9, we present our conclusions and discuss areas for future work.

2. Credit Card Dataset

Credit card information is organized into two distinct datasets:

1. a credit approval dataset and
2. a credit card transaction dataset.

2.1. Credit Approval Dataset

For each user who applies for a new credit card, there is a record of the decision made as to whether to issue a card to the user or reject the application in accordance with general usage trend models based on various user attributes. To reach this decision, it is necessary to combine multiple models, where each model is applicable to a different cluster of users.

2.2. Credit Card Transaction Dataset

In actual credit card transactions, the data are complex, change constantly, and continuously arrive online. More specifically, approximately one million transactions occur per day, with each transaction completing within 1 s. During peak times, a approximately 100 transactions occur per second. Thus, credit card transaction data can be referred to as a data stream. Although data mining techniques are currently used for credit card data, an operator can only monitor approximately 2,000 transactions per day. Therefore, suspicious transactions must be effectively detected by analyzing less than 0.02% of the total number of global transactions. Further, because fraudulent transactions occur at an extremely low rate, i.e., 0.02%-0.05% of all transactions, the amount of detected fraud is extremely low relative to the massive amounts of analyzed transaction data.

In a previous study [3], the transaction data using the comma-separated value (CSV) format were described as attributed in a time order. Credit card transaction data typically have 124 attributes, of which 84 are transactional in nature, including an attribute used to discriminate whether the data refer to a fraud. The other 40 attributes comprise behavioral data and therefore refer to credit card usage. The inflow file size is approximately 700 MB per month. Mining credit card transaction data streams present inherent difficulties. In particular, this process requires efficient calculations to be performed on an unlimited data stream with limited computing resources and time. Therefore, multiple data stream mining methods make use of an approximate or probabilistic solution rather than an exact solution; however, because actual unauthorized credit card use is quite rare, these imprecise solutions do not successfully detect frauds.

3. Deep Learning

Deep learning is a new approach attracting substantial attention in the field of machine learning. It significantly improves the accuracy of abstract representations by artficially reconstructing deep structures such as the neural circuitry of the human brain. Numerous deep learning algorithms were honored in various competitions, including the International Conference on Representation Learning. More specifically, deep learning is a generic term to represent multilayer neural networks that have been researched over several years [1], [2], [10]. In general, multilayer neural networks decrease overall calculation times by performing calculations within hidden layer; however, as a result, these networks can be prone to excessive overtraining because an intermediate layer is often used to approximate each layer.

Nonetheless, technological advancements have overcome such overtraining problems, while the use of GPU computing and parallel processing has increased the tractable number of hidden layers. However, more recently, to prevent further overtraining, the maxout model (described in Section 3.1.) and the dropout technique (described in Section 3.2.).

$$h_i(x) = \text{sigmoid}(x^T W_{\dots i} + b_i) \quad (1)$$

$$h_i(x) = \tanh(x^T W_{\dots i} + b_i) \quad (2)$$

3.1. Maxout

The maxout model is simply a feed-forward architecture such as a multilayer perceptron or deep convolutional neural network that uses a new type of activation function, the maxout unit [2].

In particular, given an input $x \in \mathbb{R}^d$ (x may be either v or a hidden layer's state), a maxout hidden layer implements the function

$$h_i(x) = \max_{j \in [1,k]} z_{ij} \quad (3)$$

where $z_{ij} = x^T W_{\dots ij} + b_{ij}$, $W \in \mathbb{R}^{d \times m \times k}$ and $b \in \mathbb{R}^{m \times k}$ are learned parameters. In a convolutional network, a maxout feature map can be constructed by taking the maximum across k affine feature maps (i.e., pool across channels, in addition to spatial locations). When training with the dropout, in all cases, we perform element-wise multiplication with the dropout mask immediately prior to multiplication by weights and the inputs not dropped to the max operator. A single maxout unit can be interpreted as a piecewise linear approximation of an arbitrary convex function. In addition to learning the relationship between hidden units, maxout networks also learn the activation function of each hidden unit.

The maxout approach abandons many of the mainstays of traditional activation function design. Even though the gradient is highly spare, the representation produced by maxout is not sparse at all, and the dropout will artificially sparsify the effective representation during training. Although the maxout unit may saturate on one side or another, this is a measure zero event (so it is almost never bounded from above). Because a significant proportion of the parameter space corresponds to the function delimited

from below, maxout learning is not constrained. Moreover, the maxout function is locally linear almost everywhere, in contrast to many popular activation functions that demonstrate significant curvature. Considering all these deviations from the standard practice, it may seem surprising for the maxout activation functions to work however, we find that they are very robust, easy to train with dropout, and achieve excellent performance.

$$h_i(x) = \max_{j \in [1,k]} Z_{ij} \quad (4)$$

$$z_{ij} = X^T W_{\cdots ij} + b_{ij} \quad (5)$$

3.2. Dropout

Dropout is a technique that can be applied to deterministic feedforward architectures that predict an output y given an input vector v [2].

In particular, these architectures contain a series of hidden layers $h = \{h(1), \cdots, h(L)\}$. Dropout trains an ensemble of models comprising a subset of the variables in both v and h. The same set of parameters θ are used to parameterize a family of distributions $p(y \mid v; \theta, \mu)$, where $\mu \in \mathcal{M}$ is a binary mask determining which variables to include in the model. For each example, we train a different submodel by following the gradient $\log p(y \mid v; \theta, \mu)$ for a different randomly sampled μ. For many parameterizations of p (usually for multilayer perceptrons) the instantiation of the different submodels $p(y \mid v; \theta, \mu)$ can be obtained by element-wise multiplication of v and h with the mask μ.

The functional form becomes important when the ensemble makes a prediction by averaging the submodels' predictions. Previous studies of bagging averages used the arithmetic mean. However, this is not possible with the exponentially large number of models trained by dropout. Fortunately, some models can easily yield a geometric mean. When $p(y \mid v; \theta) = \text{softmax}(v^T W + b)$, the predictive distribution defined by renormalizing the geometric mean of $p(y \mid v; \theta, \mu)$ over $\mathcal{M}$ is simply given by $\text{softmax}(v^T W/2 + b)$. In other words, the average exponential prediction for many submodels can be computed simply by running the full model with the weights divided by two. This result holds exactly in the case of a single layer softmax model. Previous work on dropout applies the same scheme to deeper architectures, such as multilayer perceptrons, where the $W/2$ method provides only an approximation of the geometric mean. While this approximation is not mathematically justified, it performed well in practice.

4. Data Analysis Platform

In this section, we describe the data-processing infrastructure that are suitable for credit card data analysis as well as the applications of deep learning to credit card data analysis.

4.1. R

R is a programming language and an environment for statistical computing and graphics [11]. Similar to the S programming language and environment developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R is part of the GNU project and is therefore open source and publicly available. While there are some important differences between R and S, R is considered a different implementation of S, with most of the code written for S running unaltered under R. As noted above, R is available for free and includes many useful libraries such as libraries that support multivariate analysis and machine learning. While R is widely used and quite suitable for data mining.

R directly performs its processing in its memory, therefore making it unsuitable for processing considerably large amounts of data.

4.2. Google BigQuery and Amazon Redshift

Google BigQuery [12] and Amazon Redshift [13] are systems designed for inquiries that use large amounts of data. These cloud-based systems can easily store large volumes of data and process such data quickly and efficiently. Therefore, these systems can be used to interactively analyze the trends in the data; however, the current data-processing techniques need to be further developed.

4.3. Apache Hadoop

Apache Hadoop is another platform that deal with large amounts of data [14]. It divides the given process into the map and reduce operations that largely operate in parallel with one another. More specifically, the map operations process the data, whereas the reduce operations summarize the accumulated results. Together, these processes realize high-speed processing of large amounts of data; however, because processing is performed in batches, the map reduce cycle can be completed before all data are stored. Moreover, it is difficult to apply separate algorithms to different batches of map reduce processing. Additionally, it is difficult to apply an algorithm repeatedly to the same data, as required in machine learning.

4.4. Apache Storm

Apache Storm is designed to process data streams [15], with data conversion operations executed for continuously flowing data. The data source is called the spout, while the component that performs the conversion process is called the blot. Apache Storm is a model that performs processing via a combination of bolts from spouts.

4.5. Apache Spark

Apache Spark is another platform for processing large amounts of data [16] that uses generalized map reduce processing. The processing is performed by caching the work memory and designed to execute efficient iterative algorithms by maintaining the shared data that are used for repeated processing in the memory. Further, the machine learning and graph algorithm libraries that help in making Apache Spark an easy build environment for data stream mining are available, e.g.: H2O is a library providing deep learning operations for Spark [17], [18].

SparkR is an R package that provides a light-weight R-based frontend for Apache Spark [19]. In Spark 1.5.0, SparkR provides distributed data frame implementation that supports operations, such as selection, filtering, and aggregation, that are similar to those of R data frames such as dplyr, but for much larger datasets. SparkR also supports distributed machine learning using MLlib.

Herein, we perform credit card data analysis using both R and Spark. It is possible to use an extensive R library and obtain high computational performance via parallel and distributed processing provided by Spark.

5. Quick Deep Learning Methods for Time-Series Data

The time required to perform a deep learning process is an important consideration as processing large amounts of multi-attribute data using deep learning certainly requires significant computation time. As such, it is difficult for the method to also consider the processing speed. To solve this problem, we propose the use of data stream mining without the accumulation of data. Here, it is possible to realize accelerated deep learning processing that stores data for only a short period of time using the data stream mining concept. More specifically, we propose the following three approaches that apply the concept of data stream mining to deep learning.

1. Approach 1: learning using differential data with the parameters obtained before the construction of the model
2. Approach 2: constructing a new model using differential data and the majority rule
3. Approach 3: construct a new model using differential data that failed the data judgment

To obtain the differential data, we consider the window width and propose two window width approaches: separation based on a period of time and separation based on the number of data points. Because the differential data exist only as a buffer with a short lifespan, it is possible to use the recent features for each judgment. Using approaches 2 and 3, we can construct multiple models, thereby creating the appearance of long-lifespan buffers that storage the previous models.

6. Experiments

In previous research, we used the credit approval dataset from the UCI Machine Learning Repository to evaluate our experimental results [20]. In our previous study, the proposed methods were also validated using a large transaction dataset. We constructed this dataset from the actual credit card transaction dataset that contained the same number of attributes (i.e., 125), with random values within the same range specified for each attribute [8]. Herein, we applied our proposed method to large benchmark data and real transaction dataset from a real system.

6.1. Large Benchmark Data

For comparison of the proposed method, we evaluate our proposed method by using public time-series benchmark data. We used the gas sensor dataset from the UCI Machine Learning repository [6]. We used the gas sensor array drift dataset at different concentrations data set. The data set contains 10 files which are divided depending on the periods, each file has included some gases and not included other gases. Also, the number of instances of gas is also different depending on the period. In the data set, the number of instances is 13,910 and the number of attributes is 129. Attributes include gas names, which are Ethanol, Ethylene, Ammonia, Acetaldehyde, Acetone, Toluene. Use of this data set is provided as a classification problem, regression analysis, clustering, analysis of causality.

6.2. Real Transaction Dataset from a Real System

Because the given dataset is huge, we constructed a sample dataset as follows (Table 1). Each data point corresponds to one transaction, and it has no structure. Note that there were some missing values in the data. All attributes except those labeled illegal were used.

The data were sampled from the two-day transaction data that were sorted by transaction time. These data were then divided into three equal parts, forming our learning dataset. We created a learning model from each learning dataset, and a majority decision was made via these learning models. Our evaluation data were sampled from the transaction data covering the immediately succeeding 10 days.

Table 1. Summary of the Credit Transaction Dataset

Number of Instances	120,000
Number of Attributes	125
Number of Instances for Training	20,000
Class Distribution for Training	Illegal: 382 (1.91%); legal: otherwise
Number of Instance for Test	100,000
Class Distribution for Test	Illegal: 1,148 (1.148%); legal: otherwise

For our deep learning methods, we used the Scala library of H2O [17], [18]. H2O is not only a library for both Hadoop and Spark but also an R package. Further, as noted above, in deep learning, maxout is a popular function; however, we used the rectifier and tanh functions as the activation functions. We conducted verification while changing the number of iterations for learning (i.e., 10, 20, and 1000) and the number of hidden layers and neurons [i.e., (10,10,10) and (100,100,100)]. To tune for an optimum value, we conducted the experiments several times for determining the number of learns, hidden layers, and neurons in the hidden layer. For our experiments, we used OS Linux (VM on Windows7 64bit) on a machine powered by a 3.30 GHz Intel i303229 CPU and equipped with 4GB of RAM and 500GB of disk space.

7. Experimental Results

7.1. Large Benchmark Data

To verify our approach 3 of "Majority Rule Approach", we apply it to time series data set. The table 2 shows gas estimation result. According to the procedure of the proposed method, after creating the model using deep learning, we obtained 9 models. The incorrect rate of training data by each model is less than 50%, but incorrect rate of test data by each model is less than 60%. When majority voting of the proposed method with these 9 models, the incorrect rate was 95.96% and the model could not estimate gases correctly. The reason why the gases could not be identified is that there were files with no periodicity and gases not included in the file.

7.2. Real Transaction Dataset from a Real System

To evaluate our results, we used the following metrics: (1) the correct answer rate, i.e., the ratio of transactions that were illegal actually illegal among the transactions that were deemed illegal and (2) the detection rate, i.e., the ratio of transactions judged to be invalid among all illegal transactions.

In Table 3, we summarize the experimental results. Here, the number of illegal judgments represents the number of transactions determined to be illegal based on our deep learning method, whereas the number of true illegal in judgements represents number of transactions determined to be illegal based on our deep learning method and its decision is true. Results obtained from the actual system using the logistic regression model contained 85 true illegal occurrences from 1,000 illegal judgments, with a correct answer rate of 8.50% and a detection rate of 7.40%.

8. Considerations

8.1. Large Benchmark Data

In the model that makes a majority decision of the proposed method, the accuracy of each model may not be so high, but at least 50% is necessary. If the accuracy of the model is less than 50%, correctness of judgment by majority vote cannot be guaranteed.

When dealing with time series data, it is not suitable for the proposed method because it is difficult for the periodic feature to appear in the learning model or the prediction result unless it is a continuous period. There is a possibility that the method proposed in this paper is effective when creating a learning model with data that summarizes a certain period of continuous time. Consecutive periods need to match the characteristics of the data. In the case of seasonal data, if the data is created for each season and the learning model is created, the proposed method may be effective. Considering these points, it is necessary to properly separate the time series data with the proposed method is considered to be possible. Handling of classes not included in the data set needs to be considered appropriately. For example, it is conceivable to devise measures such as changing weights when majority votes are made according to classes included. Also, when an unclassified class appears in new data, it may be considered to assign it as an unknown class as an exception class by combining analysis as an outlier. By these ideas, it can be considered that it is possible to create a model with certain accuracy even for classes not included in a certain period in the proposed method.

Table 2. Gas Estimation Result by Proposed Method

	Acetaldehyde	Acetone	Ammonia	Ethanol	Ethylene	Toluene
Acetaldehyde	2	212	349	16	1	20
Acetone	188	70	0	341	0	1
Ammonia	50	28	7	489	0	26
Ethanol	55	287	13	67	174	4
Ethylene	73	83	0	424	0	20
Toluene	89	135	265	11	100	0

Table 3. Result of Credit Transaction Dataset

	Simple Deep Learning		Majority Rule Deep Learning	
	10 times	20 times	10 times	20 times
Number of Illegal Judgment	2,986	1,862	2,420	2,462
Number of True Illegal Judgements	567	574	257	340
Correct Answer Rate	18.88 %	30.82 %	11.54 %	13.81 %
Detection Rate	49.39 %	50.00 %	24.13 %	29.62 %

8.2. Real Transaction Dataset from a Real System

As presented in our results, the proposed method exhibited better performance than the actual system but was inferior to simple deep learning, because there were variations in the fraudulent data within the dataset and the models were overfitted to these data. Although satisfactory accuracy levels were not obtained as a result of our proposed work, there are areas that clearly have room for improvement, including parameter tuning and the selection method for learning data. In the future, we believe that our approach will be worthy of consideration as a candidate for high speed processing.

9. Conclusions

Herein, we considered the application of deep learning to credit card data analysis. We introduced two major applications and proposed methods for applying deep learning techniques to these applications. To verify our proposed methods, we used benchmark experiments with other machine learning approaches. Through our experiments, wes confirmed that using deep learning, we achieved the same level of accuracy as with the Gaussian kernel SVM method. Our proposed methods were also validated using a large-scale transaction dataset.

We found that deep learning requires a significant amount of time to process such large amounts of data. Therefore, we proposed three different quick approaches to deep learning for time-series data. Although our proposed method performed better than the actual system, it was inferior to simple deep learning. As a result, we did not attain a satisfactory level of accuracy. As such, there are areas that clearly show room for improvement, including parameter tuning and the selection method for learning data. In the future, we believe that our approach will be worthy of consideration as a candidate for high-speed processing.

Given these results, our future works focuses on parameter tuning and the selection method for learning data. We also plan to verify the performance for an entire dataset that has considerably low illegal use rates.

10. References

[1] Y. Bengio, “Learning deep architectures for AI,” Found. Trends Mach. Learn., vol. 2, no. 1, pp. 1–127, Jan. 2009, (Access Date: 15-Nov-2017). [Online]. Available: http://dx.doi.org/10.1561/2200000006.

[2] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, " Maxout Networks," ArXiv e-prints, Feb. 2013.

[3] T. Minegishi and A. Niimi, "Detection of fraud use of credit card by extended VFDT," in World Congress on Internet Security (WorldCIS-2011), London, UK, Feb. 2011, pp. 166–173.

[4] B. Wallace, K. Small, C. Brodley, and T. Trikalinos, "Class imbalance, redux," in 2011 IEEE 11th International Conference on Data Mining, Vancouver, BC, Canada, Dec. 2011, pp. 754–763.

[5] H. He and E. A. Garcia, "Learning from imbalanced data," IEEE Trans. on Knowl. Data Eng., vol. 21, no. 9, pp. 1263–1284, Sept 2009.

[6] M. Lichman, "UCI machine learning repository," 2013, (Access Date: 15-Nov-2017). [Online]. Available: http://archive.ics.uci.edu/ml

[7] T. J. OZAKI. Data scientist in ginza, tokyo. (Access Date: 15-Nov-2017). [Online]. Available: http://tjo-en.hatenablog.com/

[8] A. Niimi and M. Ikeda, "Three different approaches for fast implementation of deep learning for time series data," in World Congress on Internet Security (WorldCIS-2016), London, UK, Nov. 2016, p. 5pages.

[9] A. Niimi, "Deep learning for credit card data analysis," in World Congress on Internet Security (WorldCIS-2015), Dublin, Ireland, Oct. 2015, pp. 73–77.

[10] Q. Le, "Building high-level features using large scale unsupervised learning," in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, May 2013, pp. 8595–8598.

[11] R: The R project for statistical computing. (Access Date: 15-Nov-2017). [Online]. Available: https://www.r-project.org/

[12] Google cloud platform, what is BigQuery? - Google BigQuery. (Access Date: 15-Nov-2017). [Online]. Available: https://cloud.google.com/big-query/what-is-bigquery.

[13] AWS Amazon Redshift, cloud data warehouse solutions. (Access Date: 15-Nov-2017). [Online]. Available: https://aws.amazon.com/redshift/

[14] Apache Hadoop, welcome to Apache Hadoop! (Access Date: 15-Nov-2017). [Online]. Available: https://hadoop.apache.org/

[15] Apache Storm, Storm, distributed and fault-tolerant realtime computation. (Access Date: 15-Nov-2017). [Online]. Available: https://storm.apache.org/

[16] Apache Spark, lightning-fast cluster computing. (Access Date: 15-Nov-2017). [Online]. Available: https://spark.apache.org/

[17] 0xdata - H2O.ai - fast scalable machine learning. (Access Date:15-Nov-2017). [Online]. Available: http://h2o.ai/

[18] A. Candel and V. Parmar, Deep Learning with H2O. H2O, 2015, (Access Date: 15-Nov-2017). [Online]. Available: http://learnpub.com/deeplearn-ing

[19] SparkR (R on Spark) - Spark 1.5.0 documentation. (Access Date: 15-Nov-2017). [Online]. Available: https://spark.apache.org/docs/ latest/sparkr.html

[20] A. Niimi, "Deep learning with large scale dataset for credit card data analysis," in Fuzzy Systems and Data Mining II, Proceedings of FSDM2016, Macau, Dec. 2016, pp. 149–158.

11. Acknowledgements

The authors thank Intelligent Wave Inc. for their comments on credit card transaction datasets. Furthermore, this research was supported by JSPS KAKENHI Grant Number JP17K00310.

