CREDIT CARD FRAUD DETECTION USING PARALLELIZED BAYESIAN NETWORK INFERENCING

By

Amir Ghaderi, Bsc, Ottawa University, 2011

A Major Research Paper

Presented to Ryerson University

In partial fulfillment of the requirements for the degree of

Master of Science

In the Program of

Data Science and Analytics

Toronto, Ontario, Canada, 2017

AUTOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A MAJOR RESEARCH PAPER (MRP)

I hereby declare that I am the sole author of this Major Research Paper. This is a true copy of the MRP, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this MRP to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this MRP by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my MRP may be electronically available to the public.

Amir Ghaderi

CREDIT CARD FRAUD DETECTION USING PARALLELIZED BAYESIAN NETWORK INFERENCING

Amir Ghaderi

Master of Science 2017

Data Science and Analytics

Ryerson University

ABSTRACT

The number of credit card transactions are growing, taking an ever-larger share of the worlds payment system. Improved credit card fraud detection techniques are required to maintain the viability of the worlds payment system. The aim of this Major Research project is (1) to develop a Bayesian network model that is able to predict fraudulent credit card transactions with minimal false positive predictions and (2) to parallelize the inferencing process so that the parallel version has a run time that is faster than the non-parallel version. The Bayesian network was trained on credit card transaction data obtained from European cardholders from September 2013. The results determined that Bayesian networks are able to be trained to predict fraudulent credit card transaction with zero false positive predictions. In addition, Bayesian network inferencing can be efficiently parallelized to reduce the overall run time of inferencing.

Introduction

The number of credit card transactions are growing, taking an ever-larger share of the worlds payment system. As noted by Chan et al [4], an increase in credit card transactions is leading to higher rates of stolen account numbers and subsequent losses by banks. Improved credit card fraud detection techniques are required to maintain the viability of the worlds payment system. Banks have used early fraud warning systems for many years, however these systems are not sufficient in preventing all types of fraud. Advances in data mining and machine learning have opened the door for massive large scale improvements in credit card fraud detection. In addition, computing technologies have significantly improved allowing individuals to "analyze massive amounts of transactions data" Chan et al [4].

Bayesian networks (BNs) are a type of probabilistic graphical model that represent a set of random variables and their conditional probabilities in the form of a directed acyclic graph (DAG). Bayesian networks can be used to develop an inference model that is able to predict the probability of a specific event occurring. This probability is calculated using a Monte Carlo Simulation of the inference process. A Monte Carlo Simulation refers to the process of repeating an algorithm a certain number of times in order to produce a distribution of outcomes. This distribution is then analyzed and a final probability is obtained. One of the major limitations to the Monte Carlo Simulation process is that it requires a high number of iterations to produce an accurate result.

There are two major objectives/goals for this project. The first is to develop a Bayesian network model that is able to precisely predict fraudulent credit card transactions. Ideally, this model will be able to predict fraudulent credit card transactions with minimal false positive

predictions. The second objective is to parallelize the inferencing process so that the parallel version has a run time that is faster than the non-parallel version.

Literature Review

"A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest" Heckerman [5]. When used in conjunction with statistical techniques, the graphical model has several advantages for data analysis and classification. As noted by Friedman et al [6], learning Bayesian networks is often cast as an optimization problem, where the computational task is to find a structure that maximizes a statistically motivated score. Maxwell et al [7] and Gamez et al [8], mentioned that learning the structure of a Bayesian network from large dataset is known to be an NP-Hard problem. In many different domains, the application of a heuristic search has proven to be an advantageous strategy in learning Bayesian network structures. The heuristic approach is computationally efficient and, even though it does not guarantee an optimal result, many previous studies have shown that it obtains very good solutions. For this project, I implemented a greedy search algorithm known as the hill climbing algorithm. "Hill climbing algorithms are particularly popular because of their good trade-off between computational demands and the quality of the models learned" Gamez et al [8]. My Bayesian network uses a variation of the Tree Augmented Naïve Bayes (TAN) Structure, where the root node is the predicting class. The Tree Augmented Naive Bayes (TAN) Structure "outperforms naive Bayes, yet at the same time maintains the computational simplicity (no search involved) and robustness that characterize naive Bayes" Friedman et al [9].

Bayesian networks are considered to be complete models and therefore can be used to answer probabilistic queries. This process involves computing the posterior distribution of variables given evidence, and is known as probabilistic inferencing. Bayesian networks have and

are currently being used in many different domains for probabilistic inferencing. As noted by Kirkos et al [10], Bayesian networks are extremely effective models for predicting fraudulent financial statements and identifying associated factors. In terms of performance, Bayesian Belief networks out perform all other inferencing algorithms when working with financial data. In past studies, Maes et al [11], Bayesian networks have been used to predict fraudulent credit card transactions. However, as noted by Maes et al [11], the fraud detection process when using Bayesian networks is considered to be slow in comparison with other machine learning algorithms.

Researchers have been and continue to seek out incremental improvements in algorithm efficiency. As the speed at which sequential computing begins to plateau, the logical solution is to scale computing power horizontally. Hence, we see the introduction of parallel computing. "In order to solve a problem efficiently on a parallel machine, it is usually necessary to design an algorithm that specifies multiple operations on each step, i.e., a parallel algorithm" Blelloch & Maggs [12]. Bayesian network inferencing is an example of a process that has the capabilities to be split on multiple processing cores. As noted by Madsen & Jensen [13], one way of improving the efficiency of Bayesian networks is to take advantage of the inference algorithm's available parallelism. The inference algorithm relies on a Monte Carlo simulation, which can easily be parallelized, as seen in Maigne et al [14].

"Many supervised/unsupervised machine learning algorithms require a discrete feature space" Dougherty et al [3]. Bayesian networks are an example of an machine learning algorithm that requires all features to be factors, i.e. discrete variables. As noted by Dougherty et al [3], Equal Interval Width Discretization merely divides the range of observed values for a variable into k equally sized bins, where k is a user-supplied parameter. As mentioned in Kerber [15],

Equal Interval Width Discretization does not utilize instance labels in setting partition boundaries, therefore it is likely that classification information will be lost by binning values that are strongly associated with different classes into the same bin. This discretization error can be minimized by increasing the total number of bins.

Exploratory Data Analysis

The first major data exploration step that I applied to my dataset was principle component analysis (PCA). PCA is a dimensionality reduction technique that is used to identify a small number of uncorrelated variables called "principles components". The goal of PCA is to represent the maximum amount of variance in a dataset with the fewest number of principle components. As noted by Bro & Smilde [1], Principal component analysis is one of the most important and powerful method used in statistics. When working with Bayesian networks, PCA is an extremely valuable tool because it can be used to reduce the number of nodes in the directed acyclic graph. Each feature in the dataset corresponds to a node in the Bayesian network, therefore reducing the number of features significantly lowers the computational time of inferencing. Through my experimentation, I decided to limit the number of nodes in my network to 15, which comprise of the first 13 principle components, the class variable, and the amount variable.

The next major data preprocessing step that I applied to my dataset is feature normalization. Feature normalization or feature scaling is a data mining technique used to standardize the range of independent variables in the dataset. "Feature normalization is an important aspect of the classification process, since it effectively changes the characteristics of the underlying probability distributions", Davatzikos et al [2]. I normalized each of my principle components into a range of [0,1] using the following formula: $x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$

The next step in the data preprocessing phase that I applied to my dataset was variable discretization. Feature discretization is the process of converting continuous variables into discrete variables. "Many real-world classification tasks exist that involve continuous features, where such algorithms could not be applied unless the continuous features are first discretized" Dougherty et al [3]. This is exactly the case when working with Bayesian networks. Bayesian networks in R are only able to function if all of the variables in the dataset are of type "factor". In order to achieve this, I experimented with multiple different discretization methods. I finally, decided to bin the variables into 4 separate quadrants, ([0:0.25], [0.25:0.5], [0.5:0.75], [0.75:1]). It was determined that this discretization strategy proved to have the highest accuracy and precision. Ideally, I would have liked to further granularize my principle components into a larger number of bins, however a computational trade off exists. As I increase the number of bins, model accuracy and precision also increase, however this is at the cost of computational time. In a Bayesian network, each additional bin results in an extra conditional probability calculation for each record in the model. Therefore, the idea is to find the optimal number of bins, that would result in high model accuracy and precision, while minimizing the computational time. In addition to this trade off, when discretizing continuous variables there also exists a discretization error. Discretization error is the error that results from transforming a continuous variable into a discrete variable.

The next major data exploration step that I applied to my dataset is attempting to understand the correlations that exists within my dataset. Acquiring a strong understanding of the correlations that exists in any dataset is an essential component to developing an accurate and precise machine learning classification model. In order to acquire this understating, I constructed a multiple variable correlation matrix. This matrix was used to analyze the relationships between

variables and the predicting class. As you can see from figure 1, there exists correlations between a majority of the variables and the class/amount variables. There does not exist any relationships between principle components, as expected.

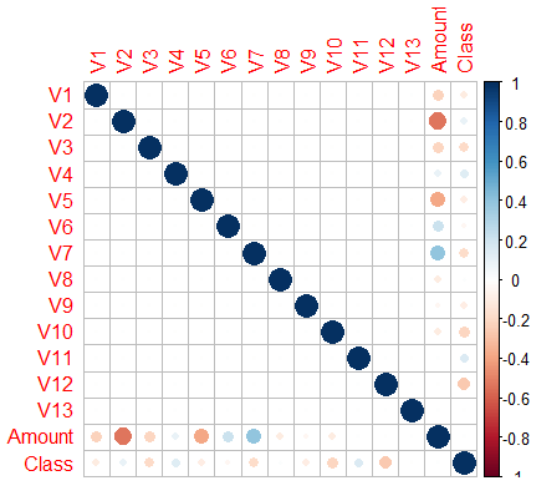

Figure 1: Multiple Variable Correlation Matrix

The last data preprocessing step that I applied was splitting my data into testing and training sets. After experimenting with various data splitting ratios, I decided to split my data ~88% training and ~12% testing. When working with Bayesian networks in R, the training set must contain all factors levels in each of the inputted variables. Therefore, I am constrained to either include a large training set or reduce the number of bins in the dataset. Reducing the number of bins would result in a loss of accuracy and precision. In addition, due to the highly-imbalanced class distribution in the dataset, it is vital that my training set contains an adequate number of positive cases. Thus, proceeding with a large training set is a more appropriate option.

Methodology

In this section I will outline the complete methodology that I followed to produce my results. The data that I have used is titled "*Anonymized credit card transactions labeled as fraudulent or genuine"* and can be found at (https://www.kaggle.com/dalpozz/creditcardfraud).

The dataset contains credit card transaction data obtained from European cardholders in September 2013. The dataset contains 284807 total transactions, of which 492 are fraud cases (0.172%), and 284315 are non-fraud cases (99.828%). As you can see the dataset is highly class imbalanced, which is typical of a credit card fraud dataset. In terms of technologies used, I used R to complete this project.

As previously mentioned, learning the structure of a Bayesian network is a NP-Hard problem. In order to learn the most optimal variation of the structure, I utilised a greedy search algorithm, known as hill climbing. The hill climbing algorithm iterates through varies Bayesian network structures, until it discovers the network with the highest statistically motivated score. After the structure of the network was determined, I manually removed edges between any of the principle components. In addition, I also added extra edges between the amount variable and the highest correlated principle components, see figure 2.

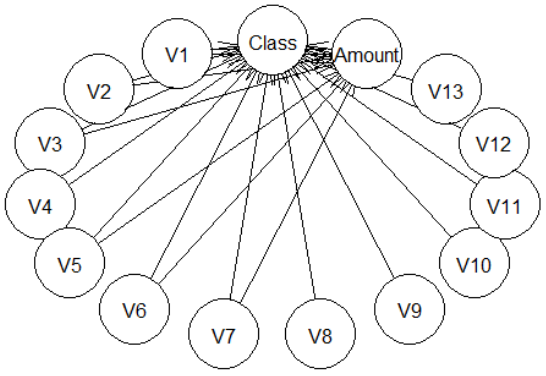

Figure 2: Bayesian Network Structure

In order to perform Bayesian network inferencing on unseen data samples, I used the R package "BNlearn" and the cpquery function. The cpquery function estimates the conditional probability of the event given the evidence. This function uses a Monte Carlo simulation to

estimate the conditional probability value. After experimenting with different iteration values, I determined that 20000 iterations provides an optimal result.

After computing the conditional probabilities of every unseen record in the testing set, the results were converted into binary predictions using a threshold value. After experimenting with different threshold values, I determined that any record with a conditional probability of greater than or equal to 0.3 would be considered as fraudulent. This threshold value was determined by selecting a value that would result in zero false positive predictions.

In order to compute the 20000 thousand iterations in parallel and thus reducing overall computational time, I used the R packages "parallel", "doParallel", and "foreach". I executed the 20000 iterations of the cpquery function across three processing cores (~6666 iterations per core). I decided to parallelize across 3 processing cores, because the maximum number of cores that the R program can operate with on one machine is (total number of available cores -1).

Results

In order to validate the performance of the Bayesian network inferencing, I used a combination of a confusion matrix, Receiver Operating Curve (ROC), and the Area under the curve (AUC). Since the data set is highly class imbalanced, (i.e. the majority of the classes in the dataset are non-fraudulent), the ROC curve is an appropriate validation metric. A ROC graph provides information regarding the number of fraudulent transactions correctly classified (true positive rate) and the number of genuine transactions that were incorrectly classified (false positive rate). As you can see in Table 1, the Bayesian network was able to correctly classify fraudulent transactions, while producing zero false positive predictions. As you can see from Figure 3, the model has a AUC value of 0.5968, which is greater than the baseline value of 0.50.

These performance measures indicate that Bayesian networks can be used as efficient models to detect fraudulent credit card transactions.

Table 1: Confusion Matrix, Actual Classes Vs Predicted Classes

Actual \ Predicted	Non-Fraudulent	Fraudulent
Non-Fraudulent	34745	0
Fraudulent	50	12

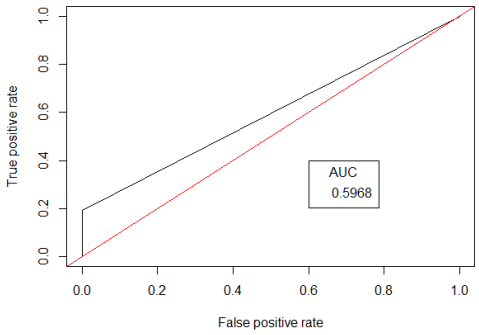

Figure 3: ROC Curve and AUC Value

In order to compare the performance of the parallelized inferencing model to the non parallelized version, I computed the Karp-Flatt metric for various iteration values. The Karp-Flatt metric is a measure to evaluate the efficiency of the algorithm parallelization. The closer the value to zero the more efficient the algorithm parallelization is. Below is the formula to calculate the Karp-Flatt Metric. Where n represents the number of machines and Ψ represents the actual speed up percentage.

$$e = \frac{\frac{1}{\Psi} - \frac{1}{n}}{1 - \frac{1}{n}}$$

As you can see from Table 2, as the number of iterations increases the Karp-Flatt metric decreases. Indicating that the algorithm parallelization is more efficient as the number of iterations increase. At the low iteration values the non-parallel version has a run time that is faster than the parallel version. This is the case because in the parallel version of the algorithm there is a cost associated to communication between computing nodes. At the low iteration

values this communication cost is larger than the benefit of parallelizing the algorithm. However, as you can see from Figure 4, the benefit of parallelization surpasses the cost of communication when the iterations values become greater than or equal to approximately 12500.

Table 2: Karp-Flatt Metric for Various Iteration Values

Iterations	Non-Parallel Time (sec)	Parallel Time (sec)	Speed Up (Ψ)	Karp-Flatt Metric
300	374	472	0.79	1.39
1000	447	481	0.93	1.11
10000	612	667	0.92	1.13
15000	842	788	1.07	0.90
21000	1027	983	1.04	0.94
30000	1252	1113	1.12	0.83
45000	1768	1248	1.42	0.56
60000	2268	1572	1.44	0.54
100000	3690	2135	1.73	0.37

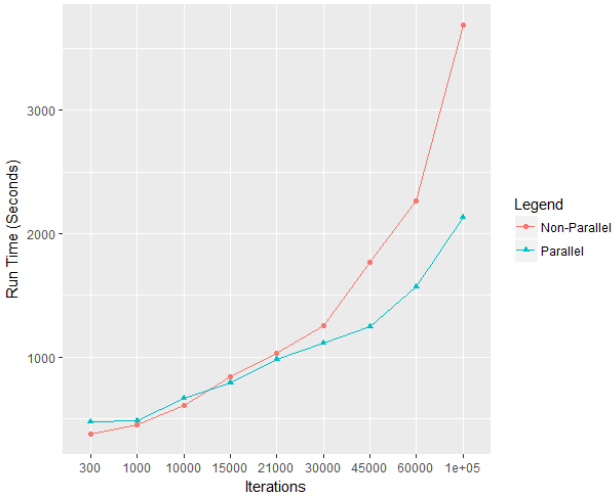

Figure 4: Time Comparison Between The Parallel Inference Model And The Non-Parallel Inference Model

Conclusions and Future Work

There were two major objectives/goals for this major research project. The first was to develop a Bayesian network model that is able to predict fraudulent credit card transactions with minimal false positive predictions. The second objective was to parallelize the Bayesian network inferencing process, so that the parallel version has a run time that is faster than the non-parallel

version. In conclusion, both of these objectives were successfully met, as demonstrated in the results section. Future work should mainly focus on two specific areas, (1) minimizing the discretization error experienced when discretizing the continuous variables and (2) increasing the number of positive predictions that the model captures.

References

[1] R. Bro and A. Smilde, "Principal component analysis", Analytical Methods, vol. 6, no. 9, p. 2812, 2014.

[2] C. Davatzikos, K. Ruparel, Y. Fan, D. Shen, M. Acharyya, J. Loughead, R. Gur and D. Langleben, "Classifying spatial patterns of brain activity with machine learning methods: Application to lie detection", NeuroImage, vol. 28, no. 3, pp. 663-668, 2005.

[3] J. Dougherty, R. Kohavi and M. Sahami, MACHINE LEARNING: PROCEEDINGS OF THE TWELFTH INTERNATIONAL CONFERENCE: Supervised and Unsupervised Discretization of Continuous Features, 1st ed. Palo Alto: Morgan Kaufmann, 1995, pp. 194-202.

[4] P. Chan, W. Fan and A. Prodromidis, "Distributed data mining in credit card fraud detection", IEEE Intelligent Systems, vol. 14, no. 6, pp. 67-74, 1999.

[5] D. Heckerman, Learning in Graphical Models: A Tutorial on Learning with Bayesian Networks, 1st ed. Cambridge: Springer Netherlands, 1998, pp. 301-354.

[6] N. Friedman, I. Nachman and D. Peér, "Learning bayesian network structure from massive datasets", artificial intelligence, pp. 206-215, 1999.

[7] D. Maxwell, D. Heckerman and C. Meek, "Large-Sample Learning of Bayesian Networks is NP-Hard", Machine Learning Researc, vol. 17, pp. 1287-1330, 2004.

[8] J. Gámez, J. Mateo and J. Puerta, "Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood", Data Mining and Knowledge Discovery, vol. 22, no. 1-2, pp. 106-148, 2010.

[9] N. Friedman, D. Geiger and M. Goldszmidt, "Bayesian Network Classifiers", Machine Learning, vol. 29, no. 23, pp. 131-163, 1997.

[10] E. KIRKOS, C. SPATHIS and Y. MANOLOPOULOS, "Data Mining techniques for the detection of fraudulent financial statements", Expert Systems with Applications, vol. 32, no. 4, pp. 995-1003, 2007.

[11] S. Maes, K. Tuyls, B. Vanschoenwinkel and B. Manderick, "Credit Card Fraud Detection Using Bayesian and Neural Networks", American Association Neurological Surgeons, pp. 261-270, 1993.

[12] G. Blelloch and B. Maggs, Algorithms and theory of computation handbook, 2nd ed. Chapman & Hall/CRC, 2010, pp. 25-25.

[13] A. Madsen, F. Jensen, A. Salmerón, H. Langseth and T. Nielsen, "A parallel algorithm for Bayesian network structure learning from large data sets", Knowledge-Based Systems, vol. 117, pp. 46–55, 2017.

[14] L. MAIGNE, D. HILL, P. CALVAT, V. BRETON, R. REUILLON, D. LAZARO, Y. LEGRE and D. DONNARIEIX, "PARALLELIZATION OF MONTE CARLO SIMULATIONS

AND SUBMISSION TO A GRID ENVIRONMENT", Parallel Processing Letters, vol. 14, no. 02, pp. 177-196, 2004.

[15] R. Kerber, "discretization of numeric attributes", Artificial intelligence, pp. 123-128, 1992.