Credit Card Fraud Detection Using Bayesian and Neural Networks

Sam Maes Karl Tuyls Bram Vanschoenwinkel
Bernard Manderick
Vrije Universiteit Brussel - Department of Computer Science
Computational Modeling Lab (COMO)
Pleinlaan 2
B-1050 Brussel, Belgium
{sammaes@,ktuyls@,bvschoen@,bernard@arti.}vub.ac.be

Abstract

This paper discusses automated credit card fraud detection by means of machine learning. In an era of digitalization, credit card fraud detection is of great importance to financial institutions. We apply two machine learning techniques suited for reasoning under uncertainty: artificial neural networks and Bayesian belief networks to the problem and show their significant results on real world financial data. Finally, future directions are indicated to improve both techniques and results.

1 Introduction

A marking example of the digitalization of our society over the last years is the proliferation of credit card use. This evolution did also bring forth the problem of credit card fraud, where ever more sophisticated methods are used to steal considerable amounts of money. This paper discusses the problem of identifying or detecting fraudulent behavior in a credit card transaction system.

Reasoning under uncertainty is a key element in many artificial intelligence applictions in general and machine learning in particular. Therefore, many formalisms have been developed that support that kind of reasoning: probabilistic reasoning, fuzzy logic, etc. Here, we focus on two machine learning techniques to the credit card fraude detections problem: Artificial Neural Networks (ANNs) and Bayesian Belief Networks (BBNs). The central idea is to provide some computational learner with a set of training data consisting of some feature values (e.g. extracted from the data of a series of financial transactions) that are inherent to the system in which we want to do the fraud detection. After a process of learning, the program is supposed to be able to correctly classify a transaction it has never seen before as fraudulent or not fraudulent, given some features of that transaction.

The structure of this paper is as follows: first we introduce the reader to the domain of credit card fraud detection. In Sections 3 and 4 we briefly explain the two Machine Learning techniques used, respectively ANN and BBN. Finally, we discuss some of the experiments conducted for both techniques, followed by a conclusion and possible future directions of research.

2 Credit Card Fraud Detection

In this section we discuss credit card fraud and the specific problems that arise with it. We will focus on credit card fraud, but most properties also apply to other real world problems, such as cellular phone fraud, calling card fraud and computer network intrusion.

2.1 What is Credit Card Fraud ?

The advent of credit cards and their increasing functionality have not only given people more personal comfort, but have also attracted malicious characters interested in the handsome rewards to be earned.

Credit cards are a nice target for fraud, since in a very short time a lot of money can be earned without taking to many risks. This is because often the crime is only discovered a few weeks after date.

Some successful credit card fraud techniques are

- Copying a credit card and in some way getting hold of the secret pin-code of the user (if

needed).

- Vendors charging more money than agreed to the customer, without the latter being aware of it.

When banks lose money due to credit card fraud, card holders partially (possibly entirely) pay for the loss through higher interest rates, higher membership fees, and reduced benefits. Hence, it is both the banks' and card holders' interest to reduce illegitimate use of credit cards and that is the reason why financial institutions started to do fraud detection.

Fraud detection is, given a set of credit card transactions, the process of identifying those transactions that are fraudulent, i.e., classifying the transactions into two classes: a class of genuine and a class of fraudulent transactions.

2.2 Problems with Credit Card Fraud Detection

One of the biggest problems associated with fraud detection is the lack of both literature providing experimental results and of real world data for academic researchers to perform experiments on. This is because fraud detection is often associated with sensitive financial data that is kept confidential for reasons of customer privacy.

We now enumerate some of the properties a fraud detection system should have in order to perform good results.

- The system should be able to handle *skewed distributions*, since only a very small percentage of all credit card transactions is fraudulent. To solve this problem, often the training sets are divided into pieces where the distribution is less skewed [Chan98].

- The ability to handle *noise*. This is simply the presence of errors in the data, for instance incorrect dates. Noise in actual data limits the accuracy of generalization that can be achieved, no matter how extensive the training set is.

  One way to deal with this problem is by cleaning the data [Faw97].

- *Overlapping data* is another problem in this field. Many transactions may resemble fraudulent transactions, when actually they are legitimate. The opposite also happens, when a fraudulent transaction appears to be normal.

- The systems should be able to *adapt* themselves to new kinds of fraud. Since after a while successful fraud techniques decrease in efficiency, due to the fact that they become well known. Then a "good" fraud tries to find new and inventive ways of doing his job.

- There is a need for *good metrics* to evaluate the classifier system. As an example, the overall accuracy is not suited for evaluation on a skewed distribution, since even with a very high accuracy, almost all fraudulent transactions can be misclassified.

- The systems should take into account the *cost* of the fraudulent behavior detected and the cost associated with stopping it. For example, no profit is made by stopping a fraudulent transaction of only a few Euros.

  This means that there should be a *decision layer* on top of the fraud detection system. The decision layer decides what action to take when fraudulent behavior is detected via the fraud detection system, taking into account factors like the amount of the transaction and the quality of the customer doing the transaction.

3 Artificial Neural Networks

3.1 Definition

Neural Networks exist in many ways and different forms. The type of neural network we will discuss in this paper is the Feed Forward Multi-layer Perceptron. A feed forward multi-layer perceptron consists of different layers of perceptrons that are interconnected by a set of weighted connections. We can distinguish three types of layers:

- Input layer: Receives input from an input stream, which can be a database or some device or something else.

- Hidden layer: Is hidden from the outside world and receives input only from the input layer or another hidden layer.

- Output layer: Connects the network to the outside world again and provides the final output of the network.

A feed forward multi-layer perceptron has no cycles and there is full connectivity between the perceptrons of two consecutive layers. Signals can be propagated in two directions: function signals are propagated forwards, i.e. from input layer through the

hidden layer(s) to the output layer and error signals are propagated backwards, i.e. from output layer through the hidden layer(s) to the input layer.

3.2 Learning

The type of learning we will discuss is commonly called supervised learning or error correction learning. The algorithm that we have used to do this is called **Backpropagation of Error Signals** or in short Backprop. Every iteration of the algorithm consists of two passes [1]:

1. **Forward pass**: every perceptron calculates the weighted linear combination of all its inputs and applies to the result of this summing junction an activation function. The result of the activation function provides the perceptron of its output value.

2. **Backward pass**: At the output layer of the network we calculate the error with respect to the desired output value for a certain pattern. This error is propagated backwards through the network enforcing a correction on the weights of all connections in the network. This technique is based on the observation that all perceptrons in the network have a shared responsibility for the error that has been calculated at the output layer.

For more details concerning Backprop and other issues concerning learning in a feed forward multi-layer perceptron we refer to [Mae00].

4 Bayesian Networks

4.1 Definition

A Bayesian network is a directed acyclic graph that consists of a set of random variables. Each variable has a finite set of mutually exclusive states. A set of directed links or arrows connects pairs of nodes. The intuitive meaning of an arrow from node X to node Y is that X has a direct **influence** on Y.

A Bayesian network represents the dependence between the variables and gives a compact specification of the joint probability distribution. In fact a BBN is a factorization of the joint probability. Each node of the network has a conditional probability table (CPT) that quantifies the effect of the parent nodes.

[1] At every iteration of the algorithm we present the network with a certain pattern taken from a training set. The features of the pattern are presented to the different perceptrons at the input layer of the network.

The parents of a node are all those nodes that have arrows pointing to it. For orphan nodes this reduces to prior probabilities.

4.2 Learning

This section deals with the problem of constructing a network automatically from direct empirical observations.

Taking Bayesian belief networks as the basic scheme of knowledge representation, the learning task separates into two additional subtasks:

1. Identifying the topology of the network, specifically, the missing links and the directionality of the arrows.

2. Learning the numerical parameters (the prior and conditional probabilities) for a given network topology.

Since the second task is trivial given enough data, we will only concentrate ourselves on learning the topology of the network. There are several approaches to do this, such as dependency analysis [Chen97] and global optimization. We opted for STAGE [Boy98] which is an instance of the latter.

Global optimization is the problem of finding the best possible configuration from a large space of possible configurations.

Formally, an instance of such a global optimization consists of a *state space* X and an *objective function* Obj: $X \rightarrow \Re$. This objective function evaluates the quality of the state as a final solution, by transforming the state into a real number. The goal of global optimization is to find a state $x^* \in X$, which minimizes Obj, that is, $\text{Obj}(x^*) \leq \text{Obj}(x)\ \forall x \in X$. If the space X is so small that every state can be evaluated, obtaining the exact solution x^* is trivial, otherwise, special knowledge of the problem structure must be exploited.

The STAGE algorithm aims to exploit the following observation: the performance of a local search algorithm depends on the state from which the search starts. We express this as follows:

The value function $V^\pi(x) \equiv$ expected best Obj value seen on a trajectory that starts from state x and follows local search method π. Intuitively $V^\pi(x)$ evaluates x's *promise* as a starting state for π.

We seek to approximate V^π using a function approximation model such as linear regression or multilayer perceptrons, where states x are encoded as real-valued feature vectors. These input features may encode any relevant properties of the state, including the original objective function $\text{Obj}(x)$ itself.

We denote the mapping from states to features by $F : X \rightarrow \Re^D$, and our approximation of $V^\pi(x)$ by $\widetilde{V}^\pi(F(x))$.

Training data for supervised learning of $\widetilde{V}^\pi$ may be readily obtained by running π from different starting points. Moreover, if the algorithm π behaves as a Markov chain, intermediate states of each simulated trajectory may also be considered alternate "starting points" for that search, and thus used as training data for $\widetilde{V}^\pi$ as well. This insight enables us to get not one but perhaps hundreds of pieces of training data from each trajectory sampled.

The learned evaluation function $\widetilde{V}^\pi(F(x))$ evaluates how promising x is as a starting point for local search algorithm π. To find the best starting point, we must optimize $\widetilde{V}^\pi$ over X. We do this by simply applying stochastic hill-climbing with $\widetilde{V}^\pi$ instead of Obj as the evaluation function.

To summarize, STAGE repeatedly alternates between two different stages of local search: running the original local search method π on Obj, and running hill-climbing on $\widetilde{V}^\pi$ to find a promising new starting state for π. Thus, STAGE can be viewed as a *smart multi-restart* approach to local search. The operation of STAGE is schematically depicted in Figure 1.

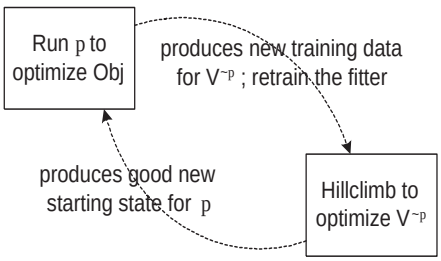

Figure 1: STAGE alternates between optimizing Obj with π and hill-climbing to optimize $\widetilde{V}^\pi$.

4.3 Stage applied to Bayesian networks

In this part we will apply the STAGE approach from the previous subsection to the problem of Bayesian network structure learning.

First we need an objective function Obj which must return a value that quantifies the quality of a Bayesian network.

We will use the Minimum Description Length (MDL) metric [Lam94], which trades off between maximizing fit accuracy and minimizing model complexity. Consider a network x with A nodes. The objective function decomposes into a sum over the nodes of the network x:

$$Obj(x) = \sum_{j=1}^{A} (-Fitness(x_j) + K \cdot Complexity(x_j)) \tag{1}$$

The *Fitness* term computes a mutual information score at each node x_j by summing over all possible joint assignments to variable j and its parents:

$$Fitness(x_j) = \sum_{v_j} \sum_{V_{Par_j}} N(v_j \wedge V_{Par_j}) \log \frac{N(v_j \wedge V_{Par_j})}{N(V_{Par_j})} \tag{2}$$

Here, $N(\cdot)$ refers to the number of records in the database that match the specified variable assignment.

The *Complexity* term simply counts the number of parameters required to store the conditional probability table at node j:

$$Complexity(x_j) = (Arity(j) - 1) \prod_{i \in Par_j} Arity(i), \tag{3}$$

where the *Arity* of a node in a Bayesian network is defined as the number of states the variable associated with the node can adopt.

The constant K in 1 is set to $log(R)/2$, where R is the number of records in the database.

Including the *Complexity* term from 3 into the objective function from 1 introduces Occam's razor in a natural way: since both K and the *Complexity* term are always positive values and since we want to minimize the objective function, there will be a tendency towards simple structures.

The local search method π that we will use is stochastic hill-climbing. Because directed cyclic graphs are not allowed in Bayesian networks we have to ensure that the graphs visited in π are acyclic. This is done by maintaining a permutation $x_{i_1}, x_{i_2}, \ldots, x_{i_A}$ on the A nodes, and all links in the graph are directed from nodes of lower index to nodes of higher index. Local search begins from a linkless graph on the identity permutation after which the following move operators are used: [2]

- With probability 0.7, choose two random nodes of the network and add a link between them (if

[2] Again, Occam's razor applies, since the simplest structures are evaluated first, followed by more complex structures.

that link isn't already there) or delete the link between them (otherwise).

- With probability 0.3, swap the permutation ordering of two random nodes of the network. This may cause multiple graph edges to be reversed, namely those that don't point from nodes with a lower index to those with a higher index after the permutation.

For learning, STAGE was given the following seven extra features:

- Features 1-2: mean and standard deviation of *Fitness* over all the nodes
- Features 3-4: mean and standard deviation of *Complexity* over all the nodes
- Features 5-6: mean and standard deviation of the number of parents of each node
- Feature 7: the number of "orphan" nodes, i.e., nodes that don't have parents

The fitter used to train $\widetilde{V}^{\pi}$ can be any method to do function approximation, like linear or quadratic regression. We used feedforward multi-layer perceptron in our implementation.

5 Experiments

5.1 Methodology

In this subsection we present a methodology to describe an experiment and its results.

The data we use is real world data that has been provided to us by Serge Waterschoot at Europay International (EPI). This data consists of a set of features that contain useful information about a transaction, we will label these features by F_i without specifying them. Unfortunately, we cannot specify them, because the agreement with EPI forbids us to do so.

We introduce a measure of performance that is independent of the learning problem at stake and gives us a clear idea about the quality of a result. For this purpose we introduce the Receiver Operating Curve (ROC) (see fig 1).

After the training of a network (ANN or BBN) it will be applied to a set of features it has never seen before. Of course, we can say something about the classification of the transactions in this set, for example, how many of these transactions have been correctly classified as genuine. Or even better, how much percent of the total of these transactions have been classified correctly as genuine.

We are especially interested in knowing how many fraudulent transactions are classified correctly as fraudulent and at the same time how many genuine transactions are classified incorrectly as fraudulent. The first is called the true positive rate and the latter the false positive rate. The ROC will combine this information in one graph. We will plot the false positives (x-axis) against the true positives (y-axis). By doing this, we get a concave shaped graph that contains for each data point on this graph the following information:

- on the x-axis you can read the percentage of transactions that have been classified incorrectly as fraudulent
- on the y-axis you can read the percentage of transactions that have been classified correctly as fraudulent

The steeper the increase with respect to the y-axis, while the values on the x-axis remain small, the better there has been learned, the better the predictions. The line, dividing the first quadrant, called the bisection, corresponds to a model that has done no learning.

5.2 Fraud Detection with Artificial Neural Networks

It is often wrongly assumed that neural networks are a fast, easy and reliable technique to obtain good results in different areas. In practice it is found that the great difficulty in applying neural networks resides in the choice of a good set of pre-processing operations and a good trade-off between the different parameters that have to be chosen.

The first experiment [3] shows the importance of pre-processing. In figure 2 (a) you can see two ROC curves, the best result in this graph has been obtained by performing a correlation analysis on the 10 features of the original data set. This resulted in the observation that one feature was strongly correlated with many of the other features. Removal of this feature clearly improves the results. For clarity we will point out some of the results on the ROC curves:

1. Dark ROC, is the best result, pre-processing: normalization, desired values are offset ϵ away

[3] For all experiments we used: a training set to train the network, a test set to calculate the average mean square error over the perceptrons at the output layer and a validation set to produce the ROC curves.

from the real desired values and correlation analysis (resulting in the removal of one feature): for 70 percent true positives we have only 15 percent false positives.

2. Light ROC, pre-processing: normalization, desired values are offset ϵ away from the real desired values: for 60 percent true positives we already have 15 percent false positives.

The second experiment shows the influence of parameter tuning on the process of learning. By decreasing the learning rate at certain intervals of the learning process we can improve the speed and efficiency of this process. This is illustrated in figure 2(b). Figure 2 (c) shows the corresponding ROC curve of this experiment. The learning rate has been dropped at epochs 5, 10, 30, 57.

5.3 Fraud Detection with Bayesian Belief Networks

We conducted one experiment on a dataset where each transaction is described by 4 features and a fraud label. A structure that received a high score from the STAGE algorithm can be seen in Figure 2 (d). As you can see two features influence fraud and fraud influences two features. Figure 2 (e) depicts the ROC associated with this structure and we can see that it performs well. For example, when 68% of the fraudulent transactions are correctly recognized, then only 10% of the genuine transactions are falsely classified as fraudulent.

Another experiment was conducted on a dataset where each transaction is described by 10 features and a fraud label. One structure returned by the learning algorithm can be seen in Figure 2 (f), and its matching ROC in Figure 2 (g). On the ROC we see that when we allow 15% of the fraudulent transactions to be incorrectly classified, 73% of the fraudulent transactions.

5.4 Comparison

- The results of BBN and ANN are compared in table 1. We can see that BBN performs better than ANN applied to fraud detection. For instance comparing ANN-(a) to BBN-(e) in the table, you can see differences of approximately 8%. This means that in some cases, BBN detects 8% more of the fraudulent transactions.

- Learning times can go to several hours for ANN and take up to 20 minutes for BBN.

- The evaluation of new examples is typically much faster for ANN than for BBN.

experiment	±10% false pos	±15% false pos
ANN-fig 2(a)	60% true pos	70% true pos
ANN-fig 2(a)	47% true pos	58% true pos
ANN-fig 2(c)	60% true pos	70% true pos
BBN-fig 2(e)	68% true pos	74% true pos
BBN-fig 2(g)	68% true pos	74% true pos

Table 1: This table compares the results achieved with ANN and BBN, for a false positive rate of respectively 10% and 15%.

6 Conclusion

In this paper we showed that good results can be achieved by applying ANN and BBN to fraud detection. As the comparison shows Bayesian Networks yield better results concerning fraud detection and their training period is shorter but the fraud detection process is considerably faster with ANNs.

Finally we point out what interesting ongoing work we are still doing and what extensions could be added to our implementations of both ANN and BBN.

Future work ANN

- Pruning algorithms exist to cut away connections and perceptrons that are practically not used during training, this can significantly improve the performance of backprop.

- The use of variations, like radial basis networks, support vector machines (SVMs) and backprop performing weight updates with respect to some other error function.

Future work BBN

- Extend the implementation so that continuous variables are allowed in the networks.

- Implement a structure learning method that is based on dependency analysis (as opposed to the search & scoring method, as in STAGE) and compare the results with those of the STAGE algorithm.

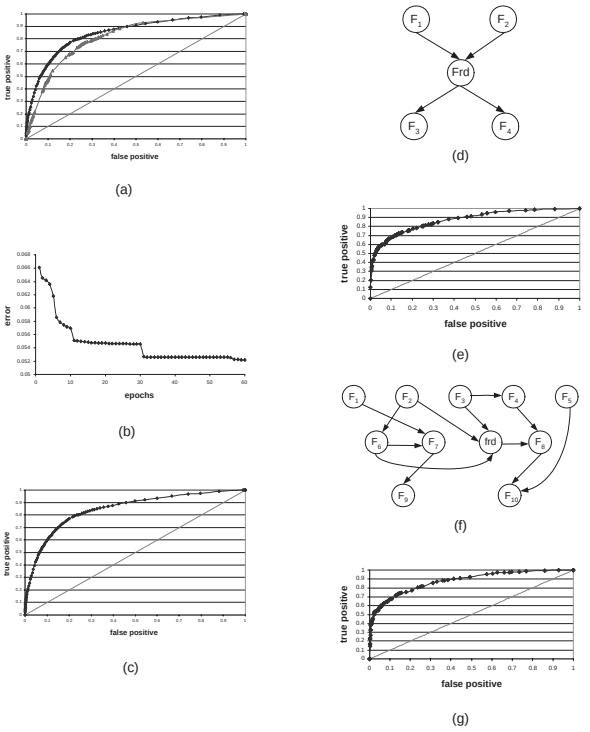

Figure 2: (a) Two ROC curves: The best result is obtained by performing a correlation analysis. (b)Average mean square error: every epoch is a complete pass of the training set through the network. (c) ROC curve corresponding to the error curve in (b). (d) A structure of five features, which received a high score from the STAGE algorithm. (e) The ROC associated with (d). (f) A structure of ten features, which received a high score from the STAGE algorithm. (g) The ROC associated with (f).

References

[Bis96] Bishop, C.M., Neural Networks for pattern recognition. Oxford University Press, 1996.

[Boy98] Boyan, J.A., Learning evaluation functions for global optimization, 1998.

[Chan97] Chan, P.K., Stolfo, S.J., Fan, D.W., Lee, W., Prodromidis, A.L., Credit card fraud detection using meta learning: Issues and initial results. Working notes of AAAI Workshop on AI Approaches to Fraud Detection and Risk Management, 1997.

[Chan98] Chan, P.K., Stolfo, S.J., Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, pp. 164–168, 1998.

[Chen97] Cheng, J., Bell, D.A. and Liu, W., Learning Bayesian networks from data: An efficient approach based on information theory. In Proceedings of ACM CIKM'97.

[Dou95] Dougherty, J., Kohavi, R., Sahami, M., Supervised and unsupervised discretization of continuous features. Proceedings of the Twelfth International Conference on Machine Learning (pp. 194–202). Tahoe City, CA: Morgan Kaufmann, 1995.

[Faw97] Fawcett, T., Provost, F., Adaptive Fraud Detection. Data Mining and Knowledge Discovery, 1(3), 1997.

[Fay97] Fayyad, U.M., Mannila, H., Piatetsky-Shapiro, G., Data mining and knowledge discovery. Kluwer Academic Publishers, 1997.

[Hay99] Haykin, S., Neural Networks: A comprehensive foundation. Second Edition, Prentice Hall, 1999.

[Hec96] Heckerman, D., A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Redmond, Washington, 1995.

[Jen98] Jensen, F.V., An introduction to Bayesian networks. London, England: UCL Press, 1998.

[Jor99] Jordan, M.I., Learning in graphical models. MIT Press, Cambridge, 1999.

[Lam94] Lam, W., and Bacchus, F. Learning Bayesian belief networks. An approach based on the MDL principle. Computational Intelligence, 10, 269–293.

[Mae00] Maes, S., Tuyls, K., and Vanschoenwinkel, B., Machine Learning Techniques for Fraud Detection. Master thesis, VUB, 2000.

[Pea88] Pearl, J., Probabilistic reasoning in intelligent systems: Networks of plausible inference. Morgan Kaufmann, San Mateo, CA, 1988.

[Ree99] Reed, R.D. and Marks, R.J., Neural Smithing: supervised learning in feed-forward artificial neural networks. MIT Press, 1999.

[Rus95] Russell, S., Norvig, P., Artificial intelligence: A modern approach. Prentice Hall Series in Artificial Intelligence. Englewood Cliffs, New Jersey, 1995.