

# Fraud Detection

## Technical comparisons of methods

The following is a brief review of some of the methods used by Card Fraud detection systems. The final section describes how Detect has addressed the problems highlighted in the main text. The following is a high-level review but some technical aspects are addressed in the appendices.

All fraud detection systems are classifiers. They attempt to classify each transaction as being legal or fraud. In general they do this by generating a score and comparing it with a threshold. The difference between the systems lies in how the score is generated.

### The card fraud problem

The problem of Card Fraud detection has several key characteristics:

Card fraud is rare. Typically less than 0.1% of the transactions are fraudulent. This means that there are very few examples of fraud in the data. The data is said to be *highly skewed* and this is the root cause of many of the problems associated with this class of problem sometime known as Rare Event detection.

Reliability of the data. The marking (aka tagging) of transactions as being fraudulent is usually a manual task and is often subject to various sources of error. These errors effectively introduce a level of 'noise' into the data. This can be a major problem.

Measuring success. The overall effectiveness of a rare event detection system means very little. In the case of fraud detection, even if it missed all the fraud it could still be claimed to be 99.9% accurate. The application of good metrics is essential when evaluating these systems. The same is true when measuring the error-rate of the system.

### The techniques used to detect fraud

#### Filtering

This where a system is checking transactions against blacklists and/or whitelists. This is an important feature of any detection system but is rarely used by itself.

#### Rule-based

Many systems are rule based. Rules can be very simple and just check for excessive spending on an account or they can be highly complex and detect changes in behaviour relative to a peer-group or another reference. More complex rules can be adaptive and can essentially be regarded as classifiers in themselves.

#### Combining evidence

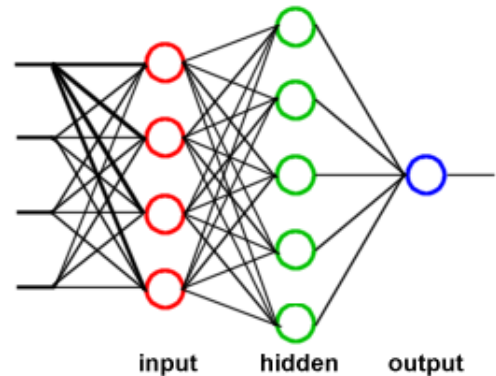
Where you have many different rules or classifiers each providing evidence for the possibility of fraud it is then necessary to combine this evidence to produce an overall probability of fraud. This is not an easy task as in general rules are rarely completely independent. The problem of cross-correlation between rules is a serious issue for any system that combines evidence but is especially a problem for simple rule-based systems. This problem is exacerbated by the inevitable fact that some rules will be better than others.

In robotics the same issue arises in the form of combining information (evidence) from sensors to determine something like 'can I reach that cup'. The topic of Sensor Fusion is very relevant to the problem of combining evidence in fraud detection systems.

### Artificial Neural Networks (ANN)

There are many types of ANN but the most common type is the feed-forward multilayer perceptron illustrated here. This network consists of three layers of nodes; the input layer, the hidden layer and the output layer. Data is passed forward through the network as illustrated by the black connecting lines. A transaction presented to the input will result in a score at the output.

The training of an ANN consists of adjusting the connection weights associated with each of the connection lines so as to minimise the errors in the output from the network when presented with inputs from the training data. Training a neural network is a form of optimisation (see appendix A). and can be a very slow process on complex highly non-linear data.



Apart from the difficulty in training neural networks they have several other draw backs.

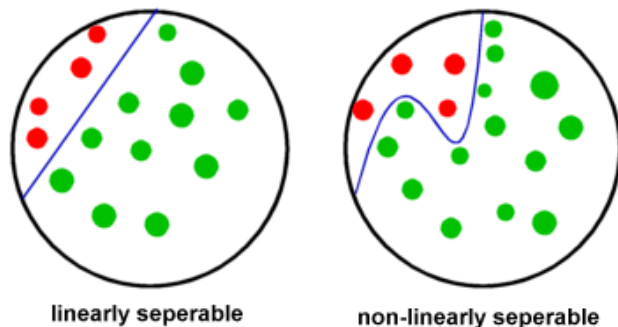
The nature of the training process does not allow incremental learning so whenever the network needs to be re-trained it is necessary to pass the full training set through the network.

When a neural network produces a score it is very difficult to understand why it produced the result it did. This is in contrast to most other methods.

On the other hand Neural networks can be very good at dealing with highly skewed data like card fraud data and once trained are very fast.

### Support Vector Machines (SVM)

SVMs are a form of Artificial Neural Network that has gained a good deal of popularity in the last few years. They employ an approach introduced by Vapnik and Cortes whereby the data is mapped to a higher dimension where it can then be divided linearly. The diagram below illustrates the concept of linear seperability. By mapping the data into a higher dimension it is often possible to find a line (hyperplane) that divides or classifies the data.



This technique has been shown to be very effective and to generalise well.

While the theoretical concept behind SVMs is very different to the standard ANN described above there is a direct equivalence.

The method of determining the mapping to the higher-dimension is effectively another example of optimisation (see later section) and subject to the same problems.

## Bayesian

The objective of any fraud detection system is to produce a score that reflects the probability that a particular transaction is fraudulent given some set of evidence. In other words we want to find *the probability of fraud given the evidence*. However, by profiling the historic (training) data this will only tell us *the probability of the evidence given it is fraud*. Bayes tells us how to use these so called a priori probabilities to compute the desired posterior probability. The simple form of Bayes is just an expression of conditional probability

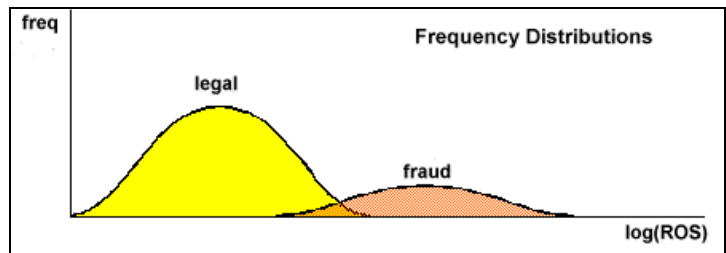
$$P(F | evidence) = \frac{P(evidence | F)P(F)}{P(evidence)}$$

### Profiling

Profiling variables is a standard statistical approach. If we profiled rate-of-spend for instance in both the legal (untagged) data and the fraud (tagged) data we would get frequency distributions that look like those illustrated:

The fraud distribution is greatly exaggerated just to illustrate.

This is an example of a very good differentiator of fraud as the two distributions are well separated. In general, this is not the case.



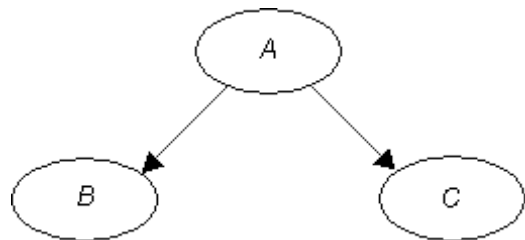
From these two distributions one can compute the probability of fraud (see appendix B)

### Bayesian Belief Networks

A Bayesian Belief Network (BBN) models the causal relationships of a dataset and provides a graphical representation of this causal structure through the use of so called Directed Acyclic Graphs (DAGs). This DAG representation then provides a framework for inference and prediction.

An example might be:

If A,B and C are parameters of a transaction then the diagram represents a causal relationship between these variables whereby there is a relationship  $A \rightarrow B$  and  $A \rightarrow C$  but there is no direct relationship between B and C.



The existence of the causal relationship  $A \rightarrow B$  is represented by the arc/edge between the nodes A and B while the strength of the relationship is represented by the conditional probability  $P(B | A)$ .

This formulism is very powerful and captures the essential relationships in the data. There are two steps to training a BBN. The first is to discover the causal relationships and the second is to compute the conditional probabilities. The latter step is trivial. There are many approaches to discovering the topology of the network; all are very slow.



BBNs have been shown to be capable of producing better performance than standard neural networks.

## Detect

Detect has combined a variety of techniques to address the problems inherent in many of the approaches we have discussed above.

There are an infinite number of variables one can derive from even simple transaction data. If we consider just rate-of-spend again. This can be computed as an average over 1 hour, 6 hours, 12 hours, 24 hours etc. Which is the best predictor of fraud?

A good predictor is one that has sufficient supporting evidence in the data as well as being a good differentiator according to the training data.

During training Detect uses a sophisticated algorithm to explore the space of possible variables and combinations of variables and chooses the best fraud differentiators. It is a form of rule-discovery.

At run time, when presented with a transaction, Detect will determine which variables to use and will combine the evidence using methods rooted in Sensor Fusion. The final step is to compute the probability of fraud and this is essentially a Bayesian calculation as described above.

The training has been designed to be incremental so that it is fast and efficient.

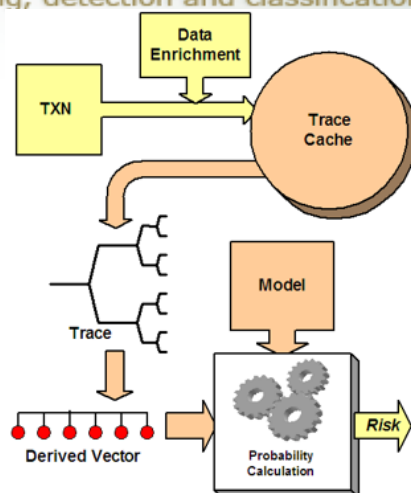
The Bayesian nature of the final calculation means that it is simple to understand why it has made a particular decision.

In order to make a probabilistic risk assessment, what we want to do (ideally) is compare the current transaction, together with its associated history, and count all the times previous similar situations ended in fraud versus all the similar situations in which no fraud occurred. The probability of fraud given the current situation can then be computed using Bayesian inference.

Reality is somewhat more complex, and some of the major problems of designing a risk engines are deciding how to compute 'similar' and making the whole process fast enough and small enough to be practical. **Detect** uses a combination of *soft statistics* and *machine learning* techniques especially designed for solving classification problems with highly skewed data sets. Performance optimisation is achieved using a form of ROC analysis for feedback.

### Data Flow

At each transaction comes in, the basic information is enriched using supporting data such as geo-location, time zone and clustering information. This is then combined with a rolling profile for each channel (e.g. card account) called a *trace* (see figure 1).



Schematic showing data flow through the risk engine.

The first step is to retrieve the existing trace (if any exists) for the appropriate channel. Specialised caching techniques are used to ensure that this happens as fast as possible. The trace includes various running estimates such as transaction rate, spend rate, average spend per transaction etc. Each running estimate is broken down into many combinations so rates, for example, are measured for different countries, different categories of merchant, different times of day and different transaction type codes etc. Because there are so many different possible combinations, and because rates need to be estimated over both long and short time-scales, there are many thousands of parameters being tracked at any moment.

Updating the trace with the enriched transaction data results in a filtered vector of derived variables. In order to make a risk assessment, the variables vector needs to be compared with a *model* of the historical variable space. This model is generated by feeding historical transaction data through the same process as described above and then 'boiling down' the resulting variables into a mathematical model of the variables which are best at discriminating fraud from non-fraud.

This 'boiling down' process is called *generalisation* and the trick is not to overdo it. Generalisation makes fast probability calculations possible but inevitably results in loss of historical detail. *Detect* preserves as much of the training data as possible using a technique called *selective abstraction*.

In order to make a risk calculation, the risk engine searches the similarity space between the trace variables and the stored model. The search is *pessimistic* in that it looks for the highest risk assessments it can find given minimum criteria for supporting evidence. This evidence is then integrated to give a final risk assessment for the transaction trace.

### **Risk Output Types**

The raw output from the risk engine is a value between 0 and 1 giving the *worst supportable case* probability of fraud for the given transaction-channel combination.

A more useful value is the **accumulated risk** measure which is the raw risk processed with a form of exponential smoothing for the channel. This also gives a measure between 0 and 1, but one which can be 'pumped up' by a succession of relatively low risk transactions occurring in close succession. The value decays exponentially over time if no further apparently risky transactions occur.

The **financial risk** metric is an accumulated measure of risk multiplied by transaction value. This also uses exponential smoothing to compute a running estimate of *expected loss* for the channel. This gives the *worst supportable case* financial exposure that the channel presents to the bank.

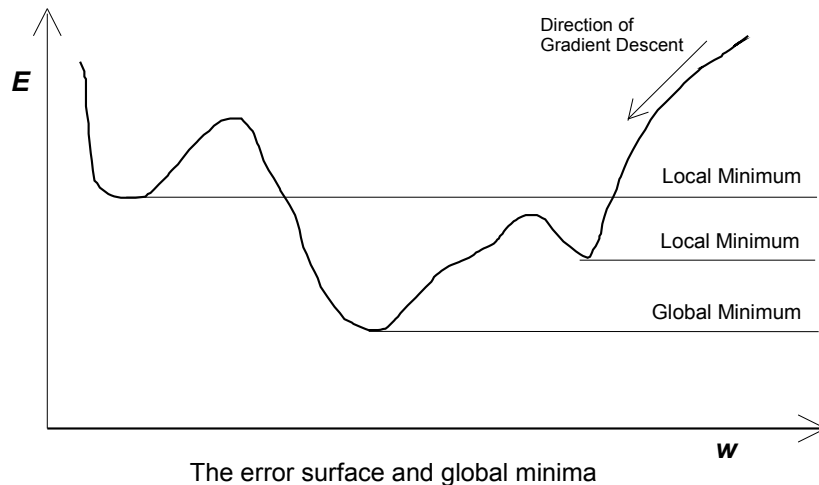


### **Predictive Tagging**

All supervised learning systems depend on marked (aka tagged) data. Detect includes a mechanism called Predictive Tagging that uses correlations in fraud transaction data to predict account delinquency. This information can then be used to mark the data prior to training and/or to validate manually tagged data. Ultimately the quality of the training data is a limiting factor on the performance of any system.

## Appendix A - Optimisation of a system

When training, many of the techniques are ultimately trying to reduce a measure of the system's error by adjusting the system's parameters. The process of finding the system's parameters whereby the error is at its lowest belongs to a generic class of problem known as optimisation. Optimisation, can be a very difficult task for complex problems like fraud detection. The problem is illustrated below for the simple case where we are trying to optimise a single parameter ( $w$ ):



The error is at its lowest at the global minimum. However, if we just travel down hill from either side of the graph we will get stuck in the local minima and never find the global minima. *This is the heart of the problem.* If we consider this in the context of a system with hundreds of parameters then we have a convoluted  $n$ -dimensional error-surface. Finding the global minimum in these circumstances is very difficult. There have been many techniques devised to solve this problem. Many are heuristic and unreliable. Those techniques that have a sound mathematical basis and can guarantee to find the solution can take a very long time and consume a great deal of computing resources. One such method is Simulated Annealing. Another extremely clever method is that introduced by Chao (see reference 1) but is not yet widely used.

### **Over-Optimisation**

Some times it is argued that we do not want to find the global minimum as the system has then over-learnt on the training data and will not generalise to other data. This can be true where the error function is only computed over the training data. Entrapment in local minima is still a big problem either way and can cause the system to under perform. However, in general it is better to compute an error that includes unseen data in which case the global minima becomes the desired goal. The reason is that otherwise one is not including the ability to generalise in the training which is obviously a key measure to optimise.

## Appendix B - The Probability Calculation

Condition probability is defined by the relationship  $P(V | S) = \frac{P(V \cap S)}{P(S)}$

Hence  $P(V | S)P(S) = P(V \cap S)$

and since  $P(V \cap S) = P(S \cap V)$  we have  $P(V | S)P(S) = P(S | V)P(V)$

giving the simple Bayes  $P(S | V) = \frac{P(V | S)P(S)}{P(V)}$

Now let  $V$  be the set of values for a derived variables and  $v \subset V$

Also let  $S$  be the set of states for the system  $s \subset S = (fraud, legal)$

The system can only be in one of the states  $s$ . In this particular case there are only two possible states and we have  $P(fraud) + P(legal) = 1$

In general we seek  $P(s = fraud | v)$  and from Bayes we have

$$P(fraud | v) = \frac{P(v | fraud)P(fraud)}{P(v)} \tag{1}$$

This then leads to the following:	We know that $P(fraud) + P(legal) = 1$ and as either <i>legal</i> or <i>fraud</i> must occur we can therefore also write $P(fraud   v) + P(legal   v) = 1$
-----------------------------------	--

As  $P(fraud \cap v) = P(fraud | v)P(v)$  and  $P(legal \cap v) = P(legal | v)P(v)$

we can therefore write

$$P(fraud \cap v) + P(legal \cap v) = [P(fraud | v) + P(legal | v)]P(v) = P(v)$$

and then using  $P(fraud \cap v) = P(fraud | v)P(v)$ , etc we have

$$P(v | fraud)P(fraud) + P(v | legal)P(legal) = P(v)$$

So we can write (1) as:

$$P(fraud | v) = \frac{P(v | fraud)P(fraud)}{P(v | fraud)P(fraud) + P(v | legal)P(legal)} = \frac{1}{1 + \frac{P(v | legal)P(legal)}{P(v | fraud)P(fraud)}}$$

If we consider this formula in the context of the frequency distributions calculated for fraud  $F(v)$  and legal  $L(v)$  then we have:

$$P(fraud | v) = \frac{(F(v) / N_F)(N_F / N)}{(F(v) / N_F)(N_F / N) + (L(v) / N_L)(N_L / N)} = \frac{F(v)}{F(v) + L(v)}$$

This is one of the calculations performed by the **Detect Risk Engine**.

## References

1. Chao J. How to Find Global Minima in Finite Times of Search - IJCNN 1991 2:1079-1084



2. Hertz J. Introduction to the Theory of Neural Computation  
Addison-Wesley 1991 pg 109
3. Haykin s. Neural Networks  
Macmillan 1994 217-219
4. McClelland and Rumelhart. Parallel Distributed Processing MIT Press 1986