SOAR – Sparse Oracle-based Adaptive Rule Extraction: Knowledge extraction from large-scale datasets to detect credit card fraud

Nick F Ryman-Tubb, *Member, IEEE,* Artur d'Avila Garcez

***Abstract*—This paper presents a novel approach to knowledge extraction from large-scale datasets using a neural network when applied to the real-world problem of payment card fraud detection. Fraud is a serious and long term threat to a peaceful and democratic society. We present SOAR (Sparse Oracle-based Adaptive Rule) extraction, a practical approach to process large datasets and extract key generalizing rules that are comprehensible using a trained neural network as an oracle to locate key decision boundaries. Experimental results indicate a high level of rule comprehensibility with an acceptable level of accuracy can be achieved. The SOAR extraction outperformed the best decision tree induction method and produced over 10 times fewer rules aiding comprehensibility. Moreover, the extracted rules discovered fraud facts of key interest to industry fraud analysts.**

I. Introduction

Fraud is prevalent in many high-volume areas, such as on-line shopping, telecommunications, banking, social security claims, etc., where a manual review of all transactions is not possible and decisions must be made quickly to prevent crime. Fraud is increasing with the expansion of computing technology and globalization, with criminals devising new frauds to overcome the strategies already in place to stop them. Automating the detection of fraud, through the use of a Fraud Management System (FMS), is therefore of strategic importance. One type of fraud is payment card fraud – this is the criminal act of deception through the use of a physical plastic card or card information without the knowledge of the cardholder. When a transaction takes place, the details of that transaction are processed by the acquiring bank for authorization. It is reported that in the USA total card fraud losses cost banks and merchants $8.6 billion per year [1] and in the UK £609.9 million [2]; despite the FMS tools already in place to tackle the problem. There are three types of fraud: (1) collusion between a merchant and a cardholder using false transactions, (2) committed using the physical payment card, called Cardholder Present (CP) – such as the interception of new credit cards in the mail, stolen/lost cards or the copying of card information onto counterfeit physical cards, employee fraud at the issuing bank, etc., and (3) committed through the use of the internet or telephone, where the Cardholder is Not Present (CNP) at the point of transaction.

A. The Importance of Fraud Detection

Traditionally, public perceptions of fraud are tempered by a belief that it is a "white-collar" crime which targets the wealthy and big business and is of less personal concern, as the effects are cushioned for the victim [3]. However, mafia figures and other violent criminals are increasingly moving into fraud [4] so that payment card fraud now involves the threat of violence, including murder. In the USA, the fear of fraud now supersedes that of terrorism, computer and health viruses and personal safety [5] and in the UK the Attorney General describes fraud as, "*second only to drug trafficking in causing harm to the economy and society*." [6]. Today, the proceeds from fraud are paying for organized crime, drug smuggling and terrorism [7, 8].

Existing FMS approaches are not keeping pace [9]; with firms rating payment fraud as the most critical threat to their business; *"...as long as criminals believe they can get away with committing fraud, the problem will continue to grow to a point where it may challenge the competitiveness of the online model*". If anti-fraud technologies do not keep pace businesses lose money from: charge-backs and fines, loss of goods, loss of reputation with their payment card facilities withdrawn and in some cases business failure. To detect fraud, organizations use a range of methods, at the most basic level this is a list of internal procedures such as fixed credit limits, transaction volume limits and so on. However, only a small number now rely on manual methods alone, with the majority employing some form of automated FMS. The FMS is often a rule-based system that stores and uses knowledge in a transparent way and is easy for a fraud expert to modify and interpret. Rules provide a convenient mechanism for explaining decisions. However, the generation of comprehensible rules is an expensive and time-consuming task, requiring a high degree of skill, both in terms of the developers and the experts concerned. The performance of the FMS is dependent upon the skill of the human expert and how past data and events are interpreted. Experts are often subjective and can only deal with a limited number of transaction fields. While it was found that such systems could be easily understood and provide an initial level of success in automating fraud decision making, often their accuracy worsened over time. To try to improve the accuracy more rules are added by the experts, but the system then becomes increasingly complex, slower to process and

Manuscript received May 2 2010. This work was supported in part by Retail Decisions Europe Ltd. (http://www.redplc.com).

Nick F Ryman-Tubb is with City University London, Department of Computing, Northampton Square, London, EC1V 0HB, UK (phone: +44 (0) 20 7040 4053; e-mail: nick.ryman-tubb@soi.city.ac.uk; web: http://www.soi.city.ac.uk/neural).

Artur d'Avila Garcez is with City University London, Department of Computing, Northampton Square, London, EC1V 0HB, UK (phone: +44 (0) 20 7040 4053; e-mail: aag@soi.city.ac.uk).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

harder to maintain and understand. The approach will recognize previously known types of fraud but once it is deployed, the criminals soon devise new methods of frauds to overcome the fixed system. It therefore becomes a battle between the criminals and how rapidly the FMS can be updated and then deployed – an alternative approach is needed.

An alternative is the use of inductive learning methods. These methods do not require expert knowledge on fraud – they can learn relationships in the transaction data based purely on learning from examples. The learning system forms a model that can then be used to process new input data and produce an output decision. The ability of the model to generalize and handle noisy data is vital – it must be able to produce a reasonable decision on previously unseen transactions. For these reasons we have chosen to use a neural network approach as discussed in detail below. The SOAR extraction method is outlined in Fig. 1, experimental results in Section III show this approach outperformed the best decision tree induction method and produced over 10 times fewer rules.

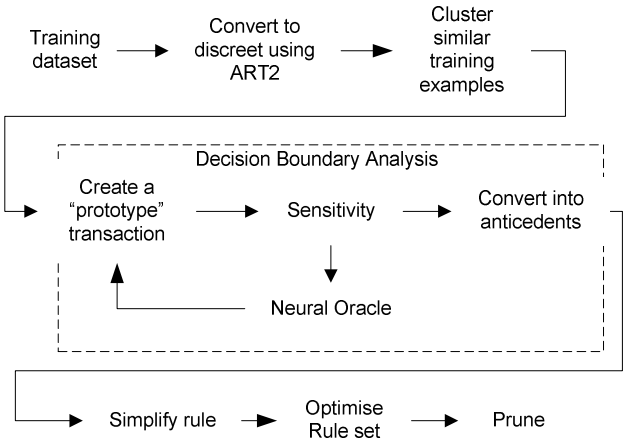

Fig. 1. Outline SOAR extraction approach.

B. Large-Scale Datasets

Over the last two decades, businesses have been faced with growing quantities of transaction data, in terms of the number of records and fields; typically, millions of records and hundreds of fields. Their unenviable task is to extract suitable information from this raw data. As well as summaries of current or past performance, they need to study the complex inter-relationships between various factors. Given a thorough understanding, predictions can be made and decisions determined. When faced with this task, the traditional approach has been to use analytical or AI methods. These methods produce a hypothesis based on *a priori* knowledge – typically in the form of rules or a model induced from labeled examples [10]. In problems such as fraud detection, the datasets are large and therefore need to be properly sampled, as it becomes computationally impractical to use the natural population during training.

Real world transactional data is noisy, unbalanced, often incomplete at the time of transaction, computationally expensive to store and maintain and highly dimensional. Such data is skewed, has uneven distributions and contains a mixture of symbolic and continuous variables. Transactions comply with ISO 8583 [11] which allows for up to 192 "data elements", of which 34 fields are pre-defined, consisting of transaction authorizing, transaction posting, non-monetary and inquiry data. The majority of these fields are symbolic data, such as merchant code, account number, name, etc. The symbolic fields may contain just two values or 100,000's values.

There is a large quantity of transactional data (typically millions of transactions a day), which contain only a small number of frauds; for CNP this is typically 1:6,000 [12] and for CP 1:150,000 or even smaller. This paucity of fraud examples makes creating and validating a neural model a difficult task. The neural network minimizes the mean error across the entire Modeling dataset (see Table III), so that the proportion of fraud to genuine has a strong influence on performance. Most published papers on rule extraction from neural networks use small datasets. There is no reason to believe that the conclusions drawn therein will hold true when they are scaled-up to large, real-world unbalanced datasets as here.

C. Neural Networks and Fraud Management

Neural networks have been widely used to model fraud data [13-15]; a model is built and knowledge is encoded and distributed throughout the network. Supervised neural networks are essentially a large number of real-valued parameters (weights, etc.) with no obvious method to determine their meaning. The knowledge is represented by the distributed weights between the connections, the threshold values and the activation function. This is why they are often criticized as "black boxes".

There are a number of important performance metrics; the FMS tries to maximize the number of transactions correctly identified as trusted, True-Positive (TP) and the number correctly identified as suspicious, True-Negative (TN). It tries to minimize the number of transactions wrongly identified as suspicious, False-Positive (FP) and the number wrongly identified as trusted, False-Negative (FN). A confusion matrix is used to evaluate results. A system that generates too many FP's is susceptible to the "base-rate fallacy" [16] – where human reviewers have a tendency to start ignoring the information produced by the system.

D. Rule-extraction from Neural Fraud Models

The black box property of neural networks has hampered uptake in the commercial sector. It is natural when a system is used in a mission-critical part of the business for a manager to wish to understand how the automated decisions are obtained. In the context of FMS, explanation may also discover emerging patterns of fraud so that action can be taken to pre-empt further crime. One method to open the black box is to provide English-like symbolic rules derived from the neural models. Rules are in human-readable form and can therefore be easily understood. By training a neural network that automatically learn the relationships in the data

and then extracting rules – symbolic knowledge can be produced. Rules need to provide "wood-from-the-trees" information "chunks" for a human expert to directly interpret (see Comprehensibility Postulate [17]). These rules offer the promise of aiding human understanding in the patterns of fraud in the data and they may highlight previously unknown types of fraud, new and emerging fraud types and give an insight into the model.

E. Rule-Extraction Approaches

The problem of extracting rules or symbolic knowledge from a supervised neural network has attracted much interest; in part as it provides a method to combine the knowledge-intensive symbolic learning and the knowledge-free approach of statistical learning into a hybrid system. There are two key approaches to rule extraction from a neural network: (1) Decompositional, (2) Pedagogical.

Initially a decompositional approach was considered, where rules are created using heuristic searches that guide the process of rule extraction [18-20]. These methods work by decomposing the neural network architecture and therefore produce rules that represent the internal architecture. Since there is no reason for hidden neurons to represent a recognizable "concept", the extracted rules were found to be not sufficiently comprehensible.

A pedagogical approach was then considered, where a set of global rules are extracted from the neural network in terms of the relationships between only the inputs and the outputs [21, 22]. These individual rules are then combined into a set of rules that describes the neural network as a whole. The main problem with this approach is that the size of the search space is large; there are three possible conditions for a conjunctive rule. With n discrete binary features, the complexity is therefore, $o(3^n)$. A straightforward pedagogical approach is not practical in real-world problems, where even a small number of input neurons (fields) mean an unrealistic level of computing power required. A key objective was to generate as few rules as possible – as a "global" view of the fundamental fraud factors is preferred over actual accuracy.

In summary, an original approach is proposed to process large real-world datasets to extract generalizing rules that are comprehensible using a trained neural network model as an oracle to locate key decision boundaries. By reducing the dimensionality of the dataset using pre-processing and then basing the rule extraction algorithm on an oracle that uses sparse fraud transactions, a novel and practical algorithm is devised.

In section II we introduce the SOAR algorithm, experiential results along with a benchmark are then discussed in section III, followed by a comparison with previous work in section IV, our conclusions in section V and an outline for future research directions in section VI.

II. METHODOLOGY

A. Pre-Processing

Unlike most algorithms tested on only small datasets, for real-world problems, a crucial part of a solution is pre-processing the data to sample, normalize and convert it to a suitable numerical representation for the neural model. This is especially true as fraud datasets are large (see Table II) and contain a mixture of numeric (continuous) and symbolic fields that are not necessarily normal or exhibit homoscedasticity. Many real-world fields are nominal (categorical) and may be ordered or unordered, or represent a ratio. An algorithm was developed to carry out the pre-processing automatically, outlined in Table I.

Table I
Outline of Pre-Processing

1)	All records checked for incomplete or missing data.
2)	Attempts are made to identify outliers in the data and to make a decision if these records should be included in the modeling process.
3)	Assessing the relevance of each field (to ensure maximum entropy) based on Shannon's Information Theory [23].
4)	Identifying redundancy in the data using linear correlation.
5)	Skew analysis of numeric data to ascertain which variables could benefit from transformation (e.g. log, square root, etc.).
6)	Frequency analysis of symbolic fields is to ascertain which low frequency fields could benefit from being grouped.
7)	Re-weighting, sampling and splitting.
8)	Continuous valued fields converted into discrete values.

To simplify the rule-extraction stage, continuous valued fields were subsequently converted into discrete values (i.e. a series of binary fields). A clustering approach was taken based on Adaptive Resonance Theory (ART2) [24]. This is an unsupervised clustering approach that develops new clusters as they are required, if sufficiently different patterns exist. ART2 calculates the Euclidean distances between the input pattern and the exemplars to determine cluster membership, except where a pattern lies outside the radii of a hypersphere set by a parameter, in which case a new cluster is formed. The algorithm was modified so that only clusters with sufficient number of members were formed. Those with few members were combined with the nearest cluster. The centre of each cluster was then used to define the start of a range, so that "bins" are formed representing a specific range.

Most humans develop habitual behaviors – this is true of their financial transactions, where reoccurring patterns of expenditure on certain goods, shops, brands, amounts can be observed over time. To try to cope with the temporal and sequential dimension of the transactions, the experimental approach herein was to create "global features" by producing derived variables that encapsulate this behavior – such as the total number of transactions on the card that day over the past five days, the average transaction amount in the same day over five days, etc. However, information is clearly lost *inter alia* by "flattening" the sequence and related temporal data.

Table II
Transaction Dataset Characteristics

Natural population of transactions	170,976,600
Number of fraud transactions in natural	1,033
Days in period	122
Approx. *a priory* fraud	1 : 165,514
Approx. transactions per day	1,400,000
Approx fraud transactions per day	9
Sample %	1%
Sampled number of transactions	1,709,796

B. Neural Network Training

A supervised neural network was chosen to be trained on the transaction examples in the Modeling dataset using a Conjugate Gradient Descent (CGD) [25] training algorithm. A three layer architecture was used based on the Multi-Layer Perceptron (MLP), this was selected as being the common choice in the previous research, noted as exhibiting good generalization ability. The MLP had 72 inputs, 10 hidden and 1 output neurons that were chosen through experimentation. All neuron values x were in the range 0 to 1 using the standard sigmoidal activation function in (1), with the slope σ set at 0.9. The MLP was trained using the sampled Modeling dataset by interleaving and repeating the fraud examples with the genuine examples.

$$f(x) = \frac{1}{1+e^{-\sigma x}} \quad (1)$$

C. Sparse Oracle-based Adaptive Rule (SOAR) extraction

The classification of a transaction by the neural model depends upon the belief that the transaction pattern is a member of the genuine or fraud class. Where examples are sparse and may exhibit class overlap, there are uncertainties and this leads to the question of how can the degree of belief be determined? The neural classifier creates a discriminant function for each point in feature space by allocating a class, so that the decision space is divided into decision regions, where each region may not be contiguous; disjoint regions may all belong to the same class and each region will have a decision surface defined by the uncertainty. Rules can be extracted by attempting to locate the class boundary. The probability of misclassification is minimized by selecting the class C_j having the smallest uncertainty, $\underline{P}(C_j)$, so that a transaction is assigned to class C_j if:

$$\underline{P}(C_j) > \underline{P}(C_k) \;\; for\ all\ k \neq j \quad (2)$$

A new approach was devised where the sparse training examples of fraud from the Modeling dataset are used as search space "seeds". The SOAR extraction algorithm designed is independent of the neural model.

Definition 1. *Training dataset*. The training dataset, **s**, is a subset of examples in the form of a vector (i ,c), where i is the input value to the neural network and c is the target class label, c={genuine, fraud}, chosen randomly from all available examples and used exclusively to train the neural network.

Definition 2. *Fraud dataset*. A list of vectors of all the fraud cases in the training dataset, $\boldsymbol{t} \in \boldsymbol{s} \cap fraud$.

Definition 3. *Literal*. A single field L from the input *i,* where $i=\{L_1, L_2, \ldots L_i\}$ each representing a concept. For continuous real world values, $L_i=[0, 1]$, and for a single symbol $L_i=\{0, 1\}$.

Definition 4. *Discrete Literal*. A continuous real world value that is quantized into a series of exclusive bins where the corresponding real value is $\{[l_1, u_1], [l_2, u_2], \ldots [l_m, u_m]\}$, where $l_i < u_i$ and $l_1 < l_2 < l_m$ and $u_1 < u_2 < u_m$ and each bin is represented by a binary digit, $L_i=\{b_1, b_2, \ldots b_m\}$. The discretised literal consists of m binary digits where each b_i is set to 1 for the corresponding range, that is, $\{1, 0, \ldots 0\}, \{0, 1, \ldots 0\}$ until $\{0, 0, \ldots 1\}$, shown as $(\{b_1, b_2, \ldots b_m\}, \oplus)$. Here, $\{b_1, b_2, \ldots b_m\} \equiv \{[l_1, u_1], [l_2, u_2], \ldots [l_m, u_m]\}$.

Definition 5. *Rule*. A rule, r_n consists of a list of discretised literals; $r=\{L_1, L_2, \ldots L_n\} \rightarrow$ Fraud. L_n denotes a conjunction of literals, i.e. $r=L_1 \wedge L_2 \wedge L_n \rightarrow$ Fraud.

Definition 6. *Rule Set*. A rule set R is made up of a list of rules; $R=\{r_1, r_2, \ldots r_n\}$. r_n denotes a disjunction of rules.

Definition 7. *Prototype*. A prototype, P_n, is in the same format as input I, grouped by similarity and then calculating the mean value of all corresponding literals so that a new centroid is produced with the average lower and upper bounds for all exemplars consisting a single cluster, $P_i=\{[A_{1l}, A_{1u}], [A_{2l}, A_{2u}], \ldots [A_{ml}, A_{mu}]\}$.

Definition 8. *Fraud Threshold*. The fraud threshold is a user defined value which determines the class based on the output, o, from the neural oracle $[0, 1]$. $o \geq$ *Fraud Threshold* $\rightarrow$ Fraud, $o <$ *Fraud Threshold* $\rightarrow$ Genuine.

Definition 9. *Outlier*. An outlier is a Discretised Literal where the range is outside the discretised range for that literal, i.e. $l_1 < L > u_m$. This case is encoded as all zeros, $L=\{b_1, \ldots, b_m\}=0$.

Definition 10. *Antecedent*. A single literal L (concept) usually forming a list of conjunctive literals.

The extraction process below takes as its input the fraud dataset, t. The process first constructs a set of prototypes $P_i \rightarrow$ *Fraud* by clustering $\boldsymbol{t_i}$ into similar groups. Each prototype P_i is then processed using sensitivity, where points that lie on the neural decision boundary are located creating a single rule r_n. A Rule Set is returned in R:

SOAR-ExtractRules
Input: *Fraud dataset* $\mathbf{t_i}$ *(i>0)*, Fraud Threshold
Output: *Rule Set* $R=\{r_0, r_1 \ldots r_n\}$

1. *Prototype P*:=ConstructPrototypes($\mathbf{t_i}$)
1. **Initialise** *Rule Set R*=[ ]
2. **for each** *Prototype* P_i
3. Rule r= Sensitivity(P_i, *Fraud Threshold*)
4. ***let*** $R := R \cup r$
5. **return**: *Rule Set R*

Clustering is used to reduce the number of rules produced to create a rule set that is more comprehensible and generalizing. The algorithm creates a single prototype that can be considered as the closest to the centre of the cluster. Finally, a list of prototypes is returned, each of which are used to subsequently generate a single rule:

```
ConstructPrototypes
Input: Fraud dataset t_i  ( i>0)
Output: list of prototypes
    1.  Number_clusters , Clusters := Cluster(t_i)
    2.  for each Number_clusters n
    3.      Prototype : = CalculateCentroid(Clusters_n)
    4.  return: list of prototypes
```

A well known algorithm ART2 was chosen [24] and is applied to t_i to group together similar fraud examples. A step-by-step description of the ART2 algorithm used here is given in [26]. The ART2 algorithm creates a new cluster when the input t_i does not belong to the cluster that was determined as most probable, based on a user defined parameter.

```
Cluster
Input: Fraud dataset t_i  (i>0)
Output: Clusters, weights
    1.  let weighs=t_i
    2.  let Cluster=1
    3.  for each t_i
    4.      calculate Euclidean_distance(t_i, weights)
    5.      let closest=0
    6.      for each cluster
    7.          if distance_cluster <=threshold then closest=cluster
    8.      if closest > 0 then update_weights(weights,closest)
    9.      else  form_new_cluster(weights, t_i,closest)
    10. return: Cluster, weights
Form_new_cluster
Input: Fraud cases (weights, t_i, closest)
    1.  let Cluster = Cluster + 1
    2.  weights_cluster,closest= t_i
Update_weights
Input: Fraud cases (weights, t_i, closest)
    1.  weights_cluster,closest= t_i
```

The mean value of the literal L_k is calculated for each exemplar in the cluster so that a new centroid is produced with the average lower and upper bounds for all exemplars consisting of a single cluster:

```
CalculateCentroid
Input: Cluster C_i
Output: centroid
    1.  for each Literal L_k in cluster C_i
    2.      if L_k is a Discreet Literal then
    3.          let m: = number binary digits that comprise L_k
    4.          let centroid (L_k) := (∑_m discrete values) / m
    5.      else centroid (L_k) := L_k
    6.  return: centroid
```

The core algorithm for SOAR is now given. This algorithm efficiently searches for points on the decision boundary that represent fraud. The prototype generated in the previous steps is "expanded", to cover the largest area on the decision boundary that continues to represent fraud. This is accomplished using the neural network as an oracle, where the binary digits in each literal in the prototype are sequentially activated, e.g., {1,0,…0}, {0,1,…0} until {0,0,…1}, and the class membership determined by the oracle, e.g. *{genuine, fraud}*. In the case of a discretised numeric type, each b_i represents a contiguous real range and so the search is simplified by only having to activate each in turn, $(\{b_1,b_2,b_m\}, \oplus)=1$.

Step 3 ensures that only fields that represent continuous values are expanded, since it would make no sense to expand an unordered nominal field.

```
Sensitivity
    Input: Prototype P_i,  Fraud Threshold
    Output: Rule(s) r
    1.  initialise Rule r=[ ], Antecedent a=[ ] , upper bound u=0, lower
        bound l=max-value, Number rules c=0
    2.  for each Literal L_k in P_i
    3.    if (L_k is a Discrete Literal) then
    4.      Rule r'=outlier(Prototype P_i,  literal L_k , Fraud Threshold)
    5.      for each m (m>0)        \\ m is number of binary digits in L_k
    6.          let New Centroid : = P_i with Discrete Literal
                                          L= ({b_1,…,b_m},⊕)=1
    7.          score: = QueryNeuralOracle (New Centroid)
    8.          if (score >= Fraud Threshold) then
    9.              let lower bound l: =min(l, L_m)
    10.             let upper bound u:= max(u, L_m)
    11.             let fired : = true
    12.         else if (fired :=true)
    13.             a=Generate-antecedent(L,  Antecedent a, l, u, r', c )
    14.             let fired : = false
    15.             let u=0, l=max-value
    16.         \\ end for each binary digit m
    17.       if (fired :=true) Generate-antecedent(L,  a, l, u, r' ,c)
          \\generate final part of the antecedent
    18.   else  \\ type is discreet symbolic
    19.     if (L_m: = true) then a=L ∧ → Fraud
    20.   return: r
Generate-antecedent
    Input: Discrete Literal L,  Antecedent a, lower bound l, upper bound
    u, Rule r',  Number rules c
        1.  if r'≠[] then r_c=a ∧  r'→ Fraud, c=c+1
        2.  else r_c=a ∧  (l≤L≤u) ) → Fraud, , c=c+1
```

The discretisation process can produce a literal where the range $[l_1,u_m]$ does not cover the entire real world range – in the case of outliers, for example. This case is encoded as all zeros, $\{b_1,\ldots,b_m\}=0$. If $\{b_1,\ldots,b_m\}=0 \rightarrow$ *Fraud,* then by definition all values outside $[l_1,u_m] \rightarrow$ *Fraud:*

```
Outlier
    Input: Prototype P_i, Discrete Literal L where L={b_1,b_2,…b_m}, Fraud
    Threshold
    Output: Rule r'
    1.  initialise Rule r'=[ ]
    2.  let New Centroid : = Prototype P_i with Discrete Literal L
        {b_1,b_2…b_m}=0
    3.  score: = QueryNeuralOracle(New Centroid)
    4.  if score >= Fraud Threshold then
    5.      generate Rule r'=¬ (l_0≤L≤u_k)  ∧ → Fraud
    6.  return: r'
```

The main difference between SOAR and existing approaches is outlined in section IV; (a) each variable is first discretized using ART2, so that the subsequent step of Decision Boundary Analysis (DBA) no longer needs a binary search to find a hypercube and (b) it creates an n-polytope by expanding each binary-discretised field, in a method similar to sensitivity analysis. In this case, the fraud examples only are first clustered and then used as the basis for DBA. This is computationally efficient and scales well. For each "expansion" in this search, the oracle is used to produce a classification, so that the upper and lower bounds on each field that make up the rule can be efficiently located. SOAR is a general approach that assumes the fraud class is reasonably sparse – which in most real-world classification problems is often the case.

III. Experimental Results

We performed experiments using a 1% random sample (a sampling rate of S_g=100), taken from the natural population, containing 100% of the known fraud transactions (see Table II). The sample was then split into two independent datasets used to train and then validate the neural model (see Table III); here, the segmentation ratio (R) is just 5%.

Table III
Dataset

	TOTAL	*Modeling*		*Validation*	
Total Transaction Records	1,709,796	80,724	5%	1,629,072	95%
Fraud Records	1,033	724	70%	309	30%

The *a priory* incidence of fraud was found to be extremely low in the natural population, with just one fraud per 165,517 transactions making this a difficult unbalanced modeling task. Performance metrics were produced using a confusion matrix on the Validation dataset with accuracy given in (3) and precision (4).

$$accuracy = \frac{TP+TN}{TP+FN+FP+TN} \quad (3)$$

$$precision_{genuine} = \frac{TP}{TP+FP} \quad (4)$$

A. . Neural Model

From Table IV, the neural network achieved 45% accuracy, 45% precision for genuine transactions, 82% precision for fraud transactions and a False Positive Ratio (FPR) of 1:2,184. The neural network is conservative as it misclassifies a large number of genuine transactions due to the sparse examples.

Table IV
Neural Model Results

		PREDICTED BY NEURAL	
		Genuine	Fraud
ACTUAL VALUE	Genuine	742,456	925,930
	Fraud	91	424

The transactions were then grouped by their payment card ID to form an individual account. Should any transaction be predicted to be fraud by the neural network – then the account is alerted, see Fig. 2 and Fig. 3.

Fig. 2. The output of the neural model is scaled to [0,1,000] and plotted against the number of payment card accounts alerted as fraud.

Metrics used to determine performance are projected back to the natural population to provide realistic results, as in (5).

$$Account\ Alerts/day = \frac{(\#genuine \times S_g \times \mathrm{R}) + (\#\mathrm{fraud} \times \mathrm{R})}{days} \quad (5)$$

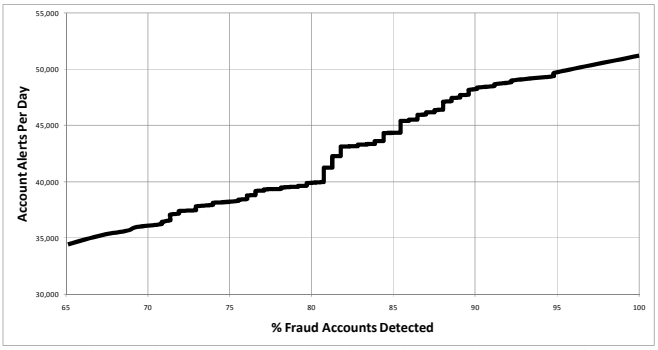

Fig. 3. The number of accounts alerted per day as a function of the % of fraudulent accounts detected by the neural network. The graph can be used to select an appropriate number of alerts for the specified fraud account detection.

Fig. 4 shows the performance of the neural model, here the more the chart bends to the top left, the better.

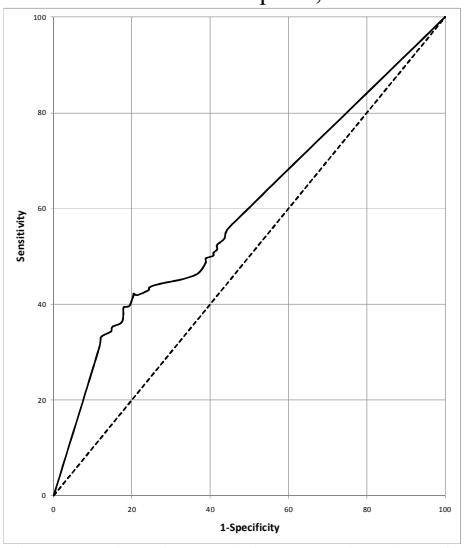

Fig. 4. Receiver Operating Characteristic (ROC) chart shows how well the neural model is able to be specific (catch only "frauds") and sensitive (catch all "frauds") simultaneously. A completely random guess would give a point along the diagonal line plotted (the line of no-discrimination).

The dollar savings that could be made by stopping an account when it is first alerted by the neural model is set out in Fig. 5.

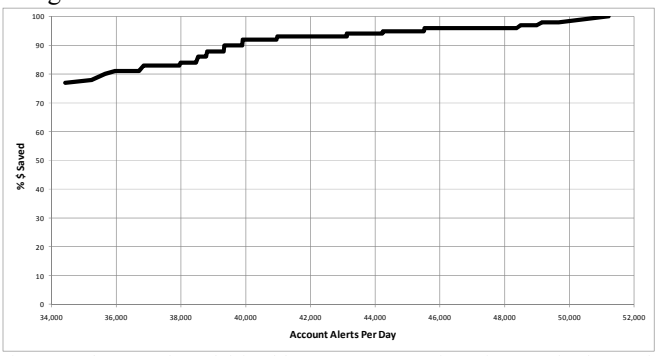

Fig. 5. The neural model is able to generate savings from early detected fraud on an account – despite the poor classification results.

B. SOAR Extraction

A rule set of 44 rules was extracted from the trained neural model using SOAR extraction algorithm – achieving 62% accuracy, 62% precision for genuine transactions, 61% precision for fraud transactions and a FPR 1:3,245.

The rule-set performs differently from the neural network (i.e. the fidelity of extraction). The quality of the rules will depend on the performance of the neural network. Counterintuitively, SOAR extraction is able to outperform the neural model as the boundaries located form crisp rules so that the (inaccurate) generalization of the neural model is not fully captured in the extraction. Further, rules that generate a large false-positive when tested with the Modeling data are removed.

Table V
SOAR Extraction Results

		PREDICTED BY RULES	
ACTUAL VALUE		Genuine	Fraud
	Genuine	1,012,223	616,540
	Fraud	119	190

Despite the poor accuracy of the neural model, the extracted rules have a reasonable real-world performance and importantly have discovered interesting characteristics of fraudsters that had not been previously recognized by the payment card fraud domain experts.

C. Decision Tree Benchmark

To provide a benchmark, a Decision Tree (DT) was induced using the same dataset. This decision tree was then tested using the Validation dataset.

Table VI
Decision Tree Results

		PREDICTED BY DT	
ACTUAL VALUE		Genuine	Fraud
	Genuine	602,212	1,026,551
	Fraud	59	250

The results were compared to the SOAR extraction results in Table VII.

Table VII
Benchmark

	SOAR	DT
#Rules (Depth)	44	492
Antecedents	12	21
Accuracy	62%	35%
Precision fraud	62%	75%
Precision genuine	61%	35%
False-Positive ratio	1:3,245	1:13,150

It can be seen that the decision tree rule-induction method has more than ten times the number of rules to achieve just over half the fraud detection accuracy, with almost double the number of average antecedents. The monothetic algorithms used make comparisons at each node based upon constant thresholds, which when the data contains noise, generates a large number of rules and contributes to the lack of accuracy especially when generalizing. Previous work has compared DT to a connectionist approach [27] and concluded that in almost all cases, neural networks outperform in terms of accuracy with generalization.

D. Analysis

The top five SOAR extracted rules are presented in the Appendix and capture 15% of the fraud. From these top rules, it was quickly noted that "Installments=(IM2) AND Mode=(1)" appears in all five rules, indicating that fraud is most likely when the payment card has had no payments or has a outstanding credit balance for less than 6 months (i.e. cards with a longer balance appear to be genuine) and that it is a domestic transaction. Specific affiliated cards appear to be higher risk (e.g. an airline-affiliated credit card). "CardType" indicates that a gold or blue card has a higher risk of fraud (rather than the premium brand platinum card). There is no reference to the merchant ID, indicating that fraud is not specific to a particular group of merchants or location. The SOAR extraction algorithm used does not guarantee that the rule set is sound – since it uses a sampling technique to locate decision boundaries. Such a technique does not cover the entire search space and therefore some relationships may not be captured – nevertheless some important fraud relationships have been discovered.

To provide a measure of rule comprehensibility the following two measures were used:

1) Total number of rules to classify unseen transactions.
2) Average number of antecedents per rule in the rule set.

The number of antecedents in each rule is a good measure of human comprehensibility (see Table VII); shorter and simpler rules are more straightforward to understand than complex rules.

Rules are processed one at a time, if a rule "fires" then fraud is indicated and no further rules are processed for that transaction. Therefore, the rule-depth has a direct influence on the performance of the rule classifier. Fig. 6 details the depth of SOAR rules in the rule set and calculates the precision for fraud detection using (2) – selecting 30 rules, detects 155 frauds, a 18% reduction in precision, using 75% of rules.

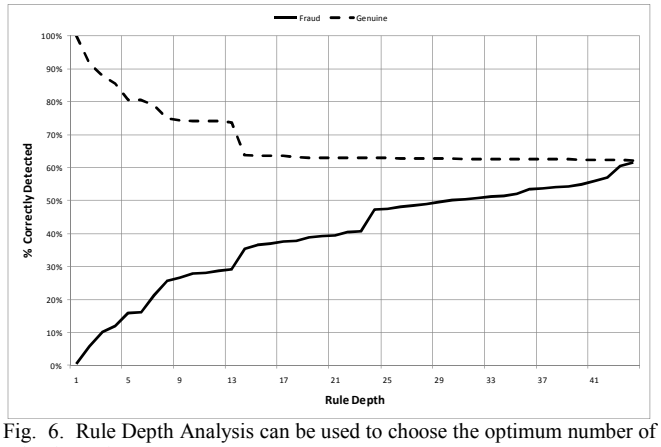

Fig. 6. Rule Depth Analysis can be used to choose the optimum number of rules for a given accuracy.

IV. Comparison with Related Work

A similar approach to SOAR extraction was from a Support Vector Machine (SVM) classifier [28]. Synthetic examples were derived from the support vectors using the target class generated by the trained SVM working as an oracle. The extraction step used K-means clustering to reduce the number of "synthetic" examples and rules were then derived. However, the approach was found not to scale well and the clustering step was replaced [29] with a decision tree [30]. In large scale problems, decision trees generate a substantial number of rules and are affected by "noise". This approach fails to create the comprehensible rule sets required in the fraud application that adequately capture the generalization of the classifier. The extraction step above was also replaced in [31] by a method to directly locate points on the SVM classification boundary using a random data generator to produce large number of examples based on kernel density estimates. Here the DBA approach was to construct hypercubes, although other methods are available [32-35].

V. Conclusion

The experiments show that symbolic rules can be automatically extracted from a trained neural network using SOAR extraction algorithm that are comprehensible and can be used within a fraud environment. The SOAR extraction algorithm outperformed the best decision tree method and produced over 10 times fewer rules aiding comprehensibility despite the sparse dataset. The SOAR rules discovered that domestic transactions, with accounts of a short outstanding balance, where they are specifically affiliated to a brand and are not a platinum card and are independent of merchant or location, have the highest risk of fraud.

All previous research was tested on small sample datasets – that do not necessarily reflect the difficulties of real world data. The unbalanced, sparse, dataset was difficult to model. These experiments have shown that the communal random patterns in the sparse subclass of the fraud class cause serious problems with convergence in the MLP neural network. This is particularly the case near decision boundaries, where insufficient examples of the fraud class will cause the neural network to learn from the noise that could be common among a large fraction of that class. Further, if the characteristics of a subclass of fraud place it in a position in data space that is isolated from the other subclasses, then many training examples from that subclass are required to avoid over fitting the random patterns common to members of that subclass. Given that only a few fraud examples are available in the training dataset, a specific subclass of fraud is likely to have only a few examples and to be contaminated with noise – and therefore may not be properly learned. However, despite these drawbacks, the extracted rules have discovered fraud characteristics not previously known and so the implementation of such an approach in the real-world is a practical proposition [36].

VI. Future Work

This work has created a firm framework for further research. The performance of the MLP classifier needs to be improved**.** Comparison against an SVM classifier would provide an interesting further benchmark. Semi-supervised neural networks appear to offer a potential solution to the classifier problem. Where the dataset has a minority class and the classification decision is needed in favor of this class, such an approach has been found to be robust [37]. The Generative Topographic Map (GTM) [38] is a promising alternative clustering method. GTM is a probabilistic clustering technique that uses a two-dimensional map that defines a probability density. A hybrid approach is hoped to significantly improve classification accuracy on sparse data. It offers the ability to base SOAR extraction within a probabilistic framework such that rules extracted can use a measure of uncertainty (2). In general, most misclassification errors occur in those regions where the uncertainty is largest, so SOAR can be extended to take this into account.

In general, shallow supervised classifiers require considerable computation power and are specific to the task; needing to be precisely created for the specific characteristics of the dataset. A key intuition from the results is that there are a range of inter-relationships between the transaction fields, some weak, which define different fraud behaviors and that these behaviors may be segmentable at different levels of abstraction. A deeper architecture capable of capturing these relationships would seem to offer a promising research approach. Such deep structures have been shown in literature to act as excellent classifiers on sparse datasets [39-42]. It is expected that the higher levels in the deep architecture represent "precepts". However, the top-level neurons do not necessarily map to obvious features (such as "high spender" or "weekend shopper") unless they are designed to do so, which then becomes domain specific. The SOAR extraction algorithm could be extended to extract rules from such a deep structure.

Acknowledgment

In memory of Sushmito Ghosh who published the foremost research in the application of neural networks in payment card fraud detection [13] and initiated this work at Retail Decisions USA. Sushmito passed away in March 2009.

Appendix

The top five SOAR extracted rules generated (out of the 44) sorted by accuracy of fraud predication are shown Table VIII and directly relate the input fields to the prediction of fraud and so are easily understood.

Table VIII
Top Five SOAR Extracted Rules

1	Instalments=(IM2) AND Mode=(1) AND Affiliate=(ACC2) AND MCC=(MCC2) AND MerchantMCCName=(1) AND Online=(1) AND CardType=(1) AND LifetimeHighCash=(LHCA5) AND NOT Amount=[$6-$2000] OR Amount=[$6-$550]
2	Instalments=(IM2) AND Mode=(1) AND Affiliate=(ACC2) AND Online=(1) AND CardType=(2)+(9) AND LifetimeHighCash=(LHCA4)
3	Instalments=(IM2) AND Mode=(1) AND Affiliate=(ACC2) AND Online=(0) AND SMS=(0) AND CardType=(2)+(9) AND LifetimeHighCash=(LHCA4)
4	Instalments=(IM2) AND Mode=(1) AND Affiliate=(ACC1) AND Online=(1) AND CardType=(2)+(9) AND LifetimeHighCash=(LHCA4)
5	Instalments=(IM2) AND Mode=(1) AND Affiliate=(ACC2) AND CardType=(2)+(9) AND LifetimeHighCash=(LHCA4)

REFERENCES

[1] P. Crosman, "Card fraud costs U.S. payment providers $8.6 billion per year," Bank Systems & Technology, New York, 13 January, 2010.

[2] "Card fraud facts and figures," UK Payments Administration, 2009.

[3] A. Castle, "Drawing conclusions about financial fraud: crime, development, and international co-operative strategies in China and the West," Transnational Financial Crime Program, The International Centre for Criminal Law Reform & Criminal Justice Policy, Vancouver, Canada, 2008.

[4] "Identity theft is perfect source of money laundering for terrorists," November, 2009. Available: http://www.identitytheft.com/index.php/article/stolen_credit_terroist_attacks.

[5] Unisys-Corporation, "Research shows economic crisis increases Americans' fears about fraud and ID theft," Unisys Pasadena, USA, April, 2009.

[6] M. Button, L. Johnston, and K. Frimpong, "The fraud review and the policing of fraud: Laying the foundations for a centralized fraud police or counter fraud executive?," Policing, pp. 241-250, 2008.

[7] BBC. "Counting the cost of UK fraud," 4th April, 2008. Available: http://news.bbc.co.uk/1/hi/business/4463132.stm.

[8] C. Everett, "Credit card fraud funds terrorism," Computer Fraud & Security, vol. 2003, no. 5, 2003.

[9] "Fourth annual UK online fraud report: online payment fraud trends, merchant and consumer response," Cybersource, Reading, UK, 2008.

[10] J. R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1, pp. 81-106, 1986.

[11] ISO. "Financial transaction card originated messages -- Interchange message specifications," May, 2008. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=31628.

[12] VISA, "Card fraud in Australia among lowest in the world," Visa International, November, 2006.

[13] S. Ghosh, and D. L. Reilly, "Credit card fraud detection with a neural network," in International Conference on System Sciences, Hawaii 1994, pp. 621-630.

[14] N. Ryman-Tubb, "Brain power – fraud prevention," Mobile Asia Pacific, pp. 20-22, 1998.

[15] A. Shen, R. Tong, and Y. Deng, "Application of classification models on credit card fraud detection," International Conference on Service Systems and Service Management, 2007, pp. 1-4, 2007.

[16] M. Bar-Hillel, "The base-rate fallacy in probability judgments," Decisions and Designs, 1977.

[17] R. S. Michalski, Machine learning: An artificial intelligence approach, p.^pp. 83-134: Morgan Kaufmann, 1983.

[18] G. G. Towell, and J. W. Shavlik, "Extracting refined rules from knowledge-based neural networks," Machine Learning, vol. 13, no. 1, pp. 71-101, 1993.

[19] R. Setiono, "Extracting rules from neural networks by pruning and hidden-unit splitting," Neural Computation, vol. 9, no. 1, pp. 205-225, 1997.

[20] A. S. d'Avila Garcez, K. Broda, and D. M. Gabbay, "Symbolic knowledge extraction from trained neural networks: A sound approach," Artificial Intelligence, no. 125, pp. 155-207, 2001.

[21] M. Craven, and J. W. Shavlik, "Using sampling and queries to extract rules from trained neural networks," International Conference on Machine Learning, pp. 37-45, 1994.

[22] S. Thrun, Advances in Neural Information Processing Systems, Extracting rules from artificial neural networks with distributed representations, Cambridge, MA, USA: MIT Press, 1995.

[23] C. E. Shannon, "A Mathematical Theory of Communication," Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, 1948.

[24] G. A. Carpenter, and S. Grossberg, "ART2 : Self-organization of stable catagory recognition codes for analog input patterns," Applied Optics, vol. 26, pp. 4919-4930, 1987.

[25] R. Fletcher, and M. J. D. Powell, "A rapidly convergent descent method for minimization," Computing Journal, vol. 6, pp. 163-168, 1963.

[26] Y.-H. Pao, "Nets for discovering cluster structure," Adaptive pattern recognition and neural networks, pp. 178-181: Addison-Wesley, 1989.

[27] G. G. Towell, J. W. Shavlik, and M. O. Noordewier, "Refinement of Approximate Domain Theories by Knowledge-Based Neural Networks," presented at the 1990 The Eighth National Conference on Artificial Intelligence (AAAI-90), Boston, Massachusetts, July/August, pp. 861-866.

[28] N. Barakat, and J. Diederich, "Learning-based rule-extraction from support vector machines," in 12th International Conference on Computer Theory and Applications, 2004.

[29] N. Barakat, and J. Diederich, "Eclectic rule-extraction from support vector machines," International Journal Computational Intelligence, vol. 2, no. 1, pp. 59-62, 2005.

[30] J. R. Quinlan, C4.5: Programs for machine learning, p.^pp. 302: Morgan Kaufmann, 1993.

[31] L. Ren, and A. S. d'Avila Garcez, "Symbolic knowledge extraction from support vector machines: A geometric approach " Lecture Notes in Computer Science, vol. 5507, pp. 335-343, 2009.

[32] A. P. Engelbrecht, and I. Cloete, "Feature extraction from feedforward neural networks using sensitivity analysis," presented at the 1998 International Conference on Systems, Signals, Control, Computer, Durban, South Africa, 22 - 24 September, 1998, pp. 221-225.

[33] A. P. Engelbrecht, and H. L. Viktor, Engineering applications of bio-inspired artificial neural networks, Rule improvement through decision boundary detection using sensitivity analysis: Springer Berlin / Heidelberg, 1999.

[34] T. H. Goh, "Semantic extraction using neural network modelling and sensitivity analysis," IJCNN '93-Nagoya. Proceedings of 1993 International Joint Conference on Neural Networks, vol. 1, pp. 1031-1034, 1993.

[35] H. L. Viktor, A. P. Engelbrecht, and I. Cloete, "Reduction of symbolic rules from artificial neural networks using sensitivity analysis," IEEE International Conference on Neural Networks, vol. 4, pp. 1788-1794, 1995.

[36] N. Ryman-Tubb, "Implementation - the only sensible route to wealth creating success: a range of applications," presented at the 1994 EPSRC : Information Technology Awareness in Engineering, London.

[37] L. Yi, G. Hong, and L. Feldkamp, "Robust neural learning from unbalanced data samples," presented at the 1998 IEEE International Joint Conference on Neural Networks, pp. 1816-1821.

[38] C. Bishop, M. Svensén, and C. Williams, "GTM: A principled alternative to the self-organizing map," Advances in neural information processing systems, MIT Press, vol. 9, pp. 354-360, 1997.

[39] R. R. Salakhutdinov, and G. E. Hinton, "Deep Boltzmann machines," in International Conference on Artificial Intelligence and Statistics (AISTATS), Florida, USA, 2009.

[40] G. E. Hinton, "Learning multiple layers of representation," Trends in Cognitive Sciences, vol. 11, no. 10, pp. 428-434, 2007.

[41] G. E. Hinton, and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks " Science, vol. 313, no. 5786, pp. 504-507, 2006.

[42] Y. Bengio, "Learning deep architectures for AI," Université de Montréal, 2007.