International Journal of Computers, Systems, and Signals, 2000, vol.1, no.2, pp.154-168

A General Neural Framework for Classification Rule Mining

Zhi-Hua Zhou   Yuan Jiang   Shi-Fu Chen

National Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, P.R.China

{daniel, jy}@aiake1.nju.edu.cn   chensf@netra.nju.edu.cn

Abstract Neural network technology has already been applied in a variety of domains with remarkable success. However, it has not been well utilized in data mining and knowledge discovery. In this paper, a general neural framework named NEUCRUM is proposed for classification rule mining. This paper also presents a possible implementation of NEUCRUM whose key components are a specific neural classifier named FANNC and a novel rule extraction approach named STARE. FANNC is used to learn from pre-processed data, in which its fast learning speed and strong generalization ability are quite contributive. STARE is proposed in this paper, which is used to extract comprehensible, compact and accurate symbolic rules from trained neural networks so that the knowledge discovered is explicitly available to decision-makers. Applications show that NEUCRUM and its implementation described in this paper work well in many real domains.

Keywords Data mining, neural networks, machine learning, knowledge discovery, rule extraction, knowledge acquisition.

1. Introduction

Data Mining is a rapidly evolving research area that is at the intersection of several disciplines, including databases, machine learning, statistics, pattern recognition, artificial intelligence, optimization, visualization, and high-performance parallel computing [1]. In this area, many traditional techniques especially decision trees are very popular [2][3]. Although neural network technology has been successfully applied to a wide variety of problem domains, it has not been well exploited in data mining. The reasons can be summarized as follows. Firstly, the business databases are usually very large and updated frequently, while most neural algorithms need long training time and can not perform incremental learning. Thus trained neural networks can hardly reflect current business effectively. Secondly, most neural algorithms perform iterative learning, which puts forward the requirement of scanning databases for quite a lot of passes. Since data retrieval techniques at present are usually time-consuming, multi-pass scanning is very expensive in access cost. Thirdly, trained neural networks are blackboxes where knowledge learned is concealed in a large number of connections. Therefore human beings can hardly utilize this knowledge to aid decision-making.

However, considering the advantages of neural networks such as in some cases they can give a lower predictive error rate than traditional techniques, neural technology should play more important roles than it plays at present. Fortunately, in recent years numerous researchers have investigated this problem. Examples are as follows. Lu et al. [4] use neural networks to mine classification rules via removing redundant connections of the network and analyze the activation values of the hidden units. Craven and Shavlik [5] analyze the obstacles hindering neural data mining, and discuss two classes of possible approaches for neural data mining. Shalvi and DeClaris [6] combine unsupervised neural networks with data visualization techniques to mine medical databases where pathology data is used for identifying natural clusters of patient populations. Esp et al. [7] use unsupervised neural networks to mine gas concentration databases maintained by The National Grid Company who owns and operates the high voltage electricity transmission network for England and Wales. Brause et al. [8] design a combined probabilistic and neuro-adaptive approach to obtain a high credit card fraud coverage combined with a low false alarm rate. Shin et al. [9] propose a hybrid data mining approach of neural network and memory-based learning, where the feature weight set calculated from the trained neural network plays the core role in connecting both learning strategies. Kewley et al. [10] propose data strip mining to solve problems with many more free parameters than data points to support them, where neural network sensitivity analysis is used to determine the significance of the predictors.

In this paper, a general neural framework named NEUCRUM (NEUral Classification RUle Mining) is proposed for classification rule mining. NEUCRUM comprises six modules, i.e. data selection, data cleansing, attribute transformation, neural learning, rule extraction, and rule evaluation. A possible implementation of NEUCRUM is also presented, whose key components are a specific neural classifier named FANNC and a novel rule extraction approach named STARE. FANNC is employed to learn from pre-processed data. Due to its characteristic of one-pass incremental learning, the first two problems cumbered the application of neural technology to data mining, which is given out in the first phase of this paper, are settled. STARE is proposed in this paper, which is employed to enhance the comprehensibility of trained neural networks. Due to its ability of extracting comprehensible symbolic rules, the third problem given out in the first phase of this paper is settled. Applications show that NEUCRUM and its above implementation work well in many domains.

The rest of this paper is organized as follows. In Section 2, we propose the NEUCRUM framework. In Section 3, we present the fast neural classifier FANNC. In Section 4, we propose the rule extraction approach STARE. In Section 5, we report on two real world applications that are built on NEUCRUM. Finally in Section 6, we conclude and indicate some issues for future work.

2. NEUCRUM Framework

2.1 Summary

Classification is fundamental to research in many fields of social and natural sciences, which has been studied extensively [11]. It is an important data mining task which can be described as follows. The input data, also called the training set, consists of multiple examples each having multiple attributes. Each example is tagged with a special class label. The objective is to analyze the input data and to develop an accurate description or model for each class using the attributes presented in the data. The class descriptions are used to classify future test data for which the class labels are unknown. They can also be used to develop a better understanding of each class in the data.

NEUCRUM is a general neural network framework designed for classification rule mining. It comprises six modules, which is illustrated in Figure 1. Note that an additional input module, i.e. the database, which does not belong to NEUCRUM, is depicted in Figure 1. This is because we believe that including the data source in the figure is better for explaining the function of NEUCRUM. There is also an additional output module, i.e. the decision-maker, depicted in Figure 1. Because we want to stress that besides the requirement of correctness, novelty and potential usefulness, the result of any data mining tasks must be comprehensible to decision-makers.

Note that we do not strictly distinguish data mining and knowledge discovery in this paper. However, in rigorous definition, knowledge discovery refers to the overall process of discovering useful knowledge from data while data mining refers to a particular step in this process, which is the application of specific algorithms for extracting patterns from data [12]. From this view, NEUCRUM should be called as a knowledge discovery framework instead of a data mining framework. And the six modules of NEUCRUM can be categorized into three big blocks as depicted in Figure 1. The first block

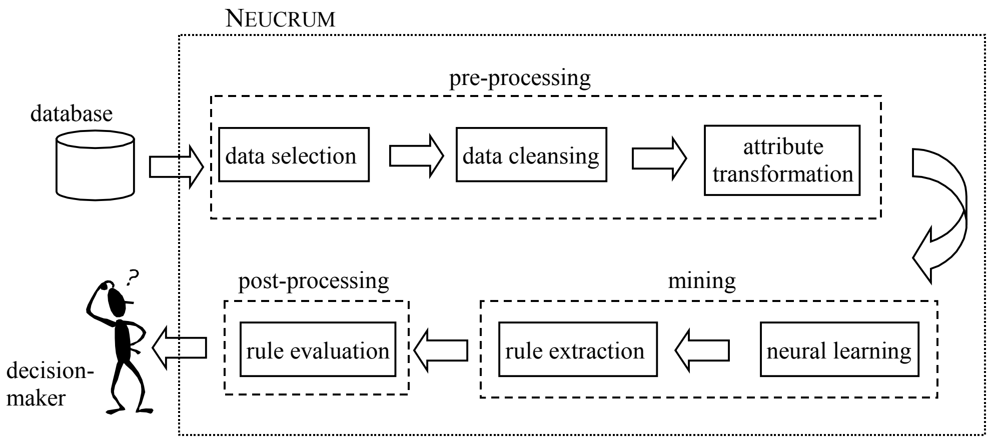


Figure 1: NEUCRUM: neural framework for classification rule mining

is a pre-processing block that comprises the data selection, data cleansing and attribute transformation modules. The second block is a mining block that comprises the neural learning and rule extraction modules. And the third block is a post-processing block that comprises only the rule evaluation module.

2.2 Data Selection

The function of the data selection module is to select appropriate data from the database according to the mining objective. Databases are usually organized in a transaction-oriented style, which is not fit for the mining of decision support knowledge. The data selection module must identify the mining objective, i.e. label the records, and adjust or even re-organize some part of the database, e.g. join several small database tables to a large one, so that it is convenient for the successive process. More important, the data selection module must remove from the databases those attributes that are irrelevant to the mining objective. This process is complicated by the fact that some “obviously” irrelevant attributes may contain surprising information. On the other hand, leaving irrelevant attributes in the data set may lead to aberrant results. So the tradeoff may be removing only those attributes that are seen as clearly irrelevant. Overall, data selection is a time-consuming process that has a substantial influence on the final mining results. In this module, expertise or a priori domain knowledge plays important roles.

2.3 Data Cleansing

Data cleansing is a very important stage in data mining. In recent years, some researchers begin to focus on this topic. Examples are Hernandez and Stolfo’s work [13]. In NEUCRUM, the function of the data cleansing module is to clean the selected data so that they can be well used in successive processes. The necessity of this module mainly exists in two facts. Firstly, after the processing of data selection module, e.g. joining of several database tables, the data integrity may not be guaranteed as before. For example, some joined records may have missing values at some attributes although all the individual tables are intact because some items do not appear in some tables. Secondly, it is often possible to benefit the successive mining process by adding some new attributes that are derived from existing attributes. This is because some relation or knowledge becomes explicitly provided so that the mining system does not need to implicitly discover them by itself.

2.4 Attribute Transformation

Although the data that come from databases have been processed by data selection and data cleansing modules, they are not ready for neural learning yet. This is because the value range of some attributes may be too large. In order to facilitate neural learning, the value ranges should be normalized to a relatively small range, e.g. [0, 1]. As to a category attribute that has finite number of possible values, it is easy to transform those values to a small value range by mapping each value to a small interval, e.g. $value_1$ maps to [0, 0.2), $value_2$ maps to [0.2, 0.4), and so on. As to a continuous attribute that has infinite number of possible values, the normalization process is complicated. The prevailing method is to use a linear mapping, e.g. dividing the difference between the value to be mapped and the *minimum value* by the difference between the *maximum value* and the *minimum value*. Although this simple normalization scheme is helpful in many cases, there exist some hidden troubles. Firstly, since the relationships among attributes and their values are very intricate, there is no guarantee that no useful information is lost during the "squashing" of the linear mapping. Secondly, since many key parameters, e.g. the *maximum value*, of the mapping is directly derived from the observation of a restricted number of processed database records, there is no guarantee that the mapping is valid for any data that may be encountered in the future, e.g. a future value may be even bigger than the *maximum value*. In short, attribute transformation involves some complex problems that should be deeply investigated. We believe that if a priori knowledge on the value distribution of the attributes is available, a knowledge-based normalization scheme may be more helpful than the popular linear mapping.

2.5 Neural Learning

The function of the neural learning module is to use neural algorithms to learn the hidden knowledge from the data that flowed from the attribute transformation module. Since there are abundant technical reports, papers, and books focusing on neural learning, a general introduction is not provided here. However, we want to stress that although there are great number of neural algorithms, only a very small number of them can be used in NEUCRUM. Firstly, since NEUCRUM is designed to perform classification rule mining, the appropriate algorithm must be a classifier. Secondly, since the data to be learned may be in great volume and may be updated frequently, the appropriate algorithm must be very fast in learning so that the access cost of scanning the databases can be relaxed and current business can be effectively reflected. Moreover, the ideal algorithm should have incremental learning ability that is very useful in reducing the computational cost spent in multi-pass learning and the storage cost spent in storing previously learned data. The specific neural algorithm used in our implementation of NEUCRUM is presented in Section 3.

2.6 Rule Extraction

Considering that neural networks are blackbox models, the knowledge learned by trained neural networks must be re-represented in a comprehensive style so that it is available to decision-makers. This work is performed by the rule extraction module. The earliest research related to rule extraction from trained neural networks can be traced back to Gallant's work [14]. After that, researchers swarm into this area so that in recent years rule extraction has become a hot topic in both the neural network and the machine learning communities. As the result, many approaches are developed [15][16], which can be divided into two groups in general. The first one is architecture-analysis-based approaches that regard rule extraction as a search process that maps the architecture of the trained neural network to a set of rules. Examples are Fu [17], Towell and Shavlik [18], Sestito and Dillon [19], Setiono [20], Krishnan et al. [21], and Tsukimoto [22] 's works. The second one is function-analysis-based approaches that do not disassemble the architecture of the trained neural networks. Instead, those approaches regard the network as an entity and try to extract rules that can explain its function. Examples are Saito and Nakano [23], Craven and Shavlik [24], Thrun [25], Benitez et al. [26], Duch et al. [27], and Taha and Ghosh [28] 's works. In NEUCRUM, the rule extraction module is implemented by a novel function-analysis-based approach, i.e. STARE, which is proposed in Section 4.

2.7 Rule Evaluation

Since a vast number of rules may be extracted from a trained neural network, a rule evaluation module is needed to assess those rules so that only those with correctness, novelty, potential usefulness and comprehensibility are presented to the decision-maker. In this module, the extracted rules must be translated into a language that can be understood by domain experts. Then the rules are analyzed by those experts so that erroneous and trivial ones are filtered out. This is a very important but very difficult stage of data mining because criteria of evaluation are different in different tasks, which highly relies on the application domain and strongly requires the intervention of human beings. Although some works [29][30] have already been done on devising different evaluation criteria, it seems that it is still far away from automated rule evaluation. Therefore in most cases the extracted rules are manually evaluated by knowledgeable human experts.

3. Fast Neural Classifier FANNC

FANNC [31] is a fast adaptive neural classifier that exploits the advantages of both Adaptive Resonance Theory [32] and Field Theory [33]. It performs one-pass learning and achieves not only strong generalization ability but also fast learning speed. Besides, FANNC has incremental learning ability. When new training examples are fed, it does not retrain the entire training set. Instead, it can learn the

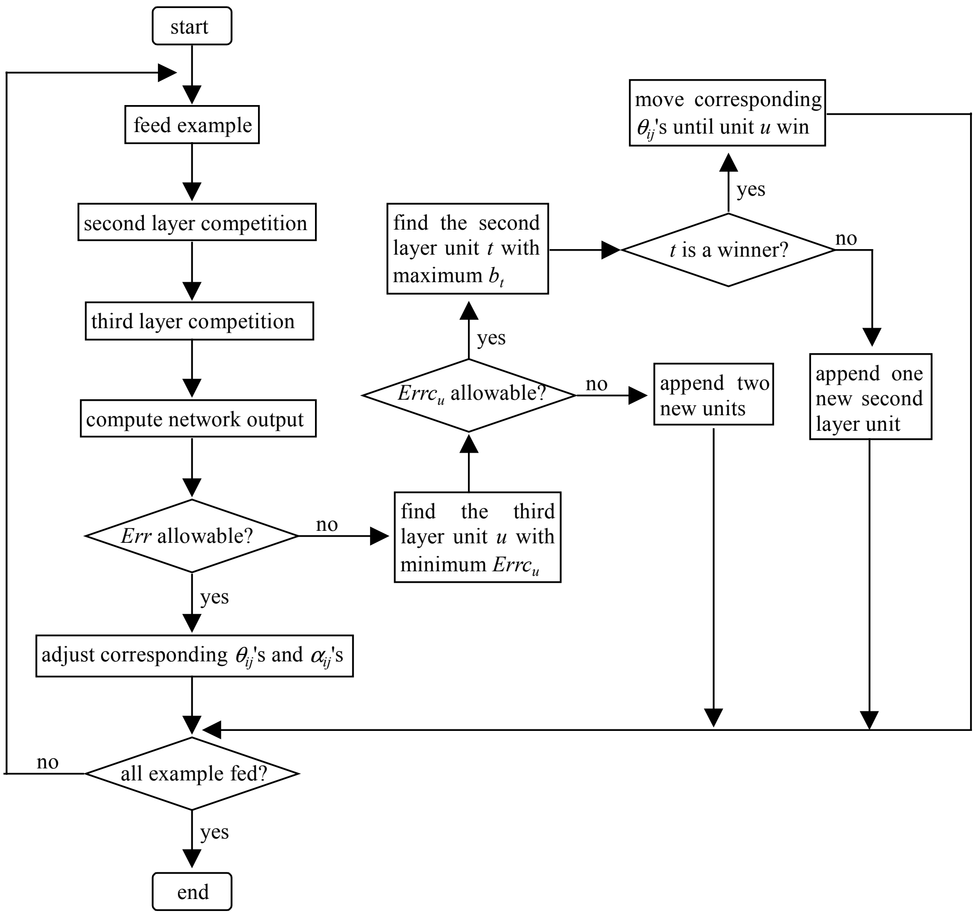

start
feed example
second layer competition
third layer competition
compute network output
Err allowable?
no
yes
adjust corresponding θ_{ij}'s and α_{ij}'s
all example fed?
no
yes
end
find the third layer unit u with minimum $Errc_u$
$Errc_u$ allowable?
yes
no
find the second layer unit t with maximum b_t
append two new units
t is a winner?
yes
no
move corresponding θ_{ij}'s until unit u win
append one new second layer unit

Figure 2: Flowchart of the learning course of FANNC

knowledge encoded in those new examples through modifying some parameters associated with existing hidden units, or slightly adjusting the network topology by adaptively appending one or two hidden units and relevant connections to existing network. Since the network architecture is adaptively set up, the disadvantage of manually configuring the number of hidden units of most feedforward neural networks is overcome. Benchmark tests show that FANNC works well even when only a few training examples are provided. Therefore FANNC is good enough to cope with the first and second obstacles that cumber neural data mining, which is stated in the beginning of this paper. Figure 2 depicts the learning course of FANNC. For the meaning of the symbols in the figure and the details of FANNC, please refer [31].

4. Rule Extraction Approach STARE

4.1 Data Generation

STARE is a function-analysis-based approach. Suppose that we have a trained neural network $\mathcal{N}$. If we feed an input pattern $A_k = (a_1, a_2, \ldots, a_n)$ to $\mathcal{N}$, we can get its corresponding output pattern $C_k = (c_1, c_2, \ldots, c_q)$ from the output units. Through combining A_k and C_k, we get an example (A_k, C_k) that is generated by $\mathcal{N}$ and can represent the function of $\mathcal{N}$ on the specific point A_k in the instance space. Considering we have a large example set $\mathcal{S}$ that comprises a lot of examples generated by $\mathcal{N}$, and the examples uniformly distribute across the whole instance space, we believe that the function of $\mathcal{N}$ is encoded in $\mathcal{S}$. Therefore if we can extract a comprehensible rule set $\mathcal{R}$ from $\mathcal{S}$, the function of $\mathcal{R}$ will approach that of $\mathcal{N}$ while the size of $\mathcal{S}$ approaching infinity, that is,

$$\lim_{|\mathcal{S}|\to\infty} func(\mathcal{R}) = func(\mathcal{N})$$

The example set $\mathcal{S}$ can be generated through slowly sliding A_k in the instance space, that is, varying each a_i ($i = 1, 2, \ldots, n$) across its value range so that diversified input patterns can appear as more as possible. For a category attribute, it is easy to be done through making all the possible values appear in turn. For a continuous attribute, it is necessary to uniformly sample across its value range with a high frequency. For example, we may get 100 patterns for an attribute whose value range is [0, 1].

4.2 Continuous Attribute Processing

After the creation of example set $\mathcal{S}$, the rule set $\mathcal{R}$ can be extracted via STARE. Note that STARE always extracts rules from category attributes. Only when new rules can not be extracted out, STARE resorts to the continuous attribute that has the best clustering effect. In other words, the number of discretized clusters is the least one that is enough for extracting valid rules. The discretized continuous attribute is regarded as a new category attribute, and used together with existing category attributes for rule extraction.

Here the discretization is done by a modified version of ChiMerge [34]. The examples are sorted according to the value of the attribute to be discretized. Initial clusters are formed through regarding each unique value as a cluster. The χ^2 value of adjacent clusters is computed and the pairs of adjacent clusters with the lowest χ^2 value are merged. In Kerber's original algorithm [34], merging continues until all pairs of clusters that have χ^2 values exceeding a user defined parameter χ^2-threshold. Instead of setting a fixed threshold value as a stopping condition for merging, we continue the merging as long as there are no examples that belong to different classes are assigned identical discretized values.

Comparing with the continuous attribute processing used by most rule extraction approaches, which discretizes all the continuous attributes at the beginning, our processing has some advantages. Firstly, in most cases, especially when there are lots of attributes, some continuous attributes do not appear in the extracted rules. The cost spent in their discretization is wasteful. In STARE, since the extracted rule set $\mathcal{R}$ may fully cover $\mathcal{S}$ before all continuous attributes being discretized, unnecessary discretization is avoided. Secondly, different attributes may distribute differently in their value range. If they are discretized to equal number of clusters, some helpful information may be obliterated. In STARE, since each time only one continuous attribute is processed, different attributes can be discretized to different number of clusters. Thirdly, since the continuous attributes are discretized one by one, the computational complexity for discretization gradually descends due to our unique rule creation process, which is elaborated later.

On the other side, our continuous attribute processing also has some disadvantages. Since continuous attributes are processed one by one, the relationship among continuous attributes is ignored, which makes STARE work quite well on problems where there is weak interaction among attributes. When facing problems where there is strong interaction among attributes, the performance of STARE degrades. However, we believe that it is still competitive to that of other rule extraction approaches because such kind of problems is very difficult in nature.

4.3 Rule Creation

The rule creation process of STARE is quite similar to what has been used by Fu [17], that is, a subset $\mathcal{H}$ of input attributes is found out and a rule is created based on it. At first, $\mathcal{H}$ is only composed of one attribute a_i that is randomly picked out from current category attributes. Consider the possible values of a_i, if there is a value u that all the examples possessing such value in $\mathcal{S}$ fall into a certain class $\mathcal{C}$, a rule r is created via regarding $a_i = u$ as the rule antecedent and regarding $\mathcal{C}$ as the rule consequent. If there is no such u, a_i is replaced by another category attribute a_j, and a similar process searching for a_j 's fitful value v occurs.

If no rule has been created after examining all the single attributes, another category attribute is appended to $\mathcal{H}$. In other words, $\mathcal{H}$ is composed of two attributes now. Suppose they are a_i and a_j, the resulting rule has two conjunctive antecedents, i.e. $a_i = u$ AND $a_j = v$. Thus, the number of rule antecedents increases while the number of attributes in $\mathcal{H}$ increasing. Since we believe that rules with more than 3 antecedents are nearly incomprehensible to human beings, we limit the maximum number of attributes in $\mathcal{H}$ to be 3. It is obvious that the limitation results in the increasing of the number of rules, but we believe that it is valuable because it also increases the comprehensibility of the extracted rules. However, the relationship between the number of rule antecedents and the comprehensibility of rules should be deeply investigated.

If no rule has been created after examining all the possible subsets of category attributes, a continuous attribute is discretized and regarded as a new category attributes as described above. If no rules can be created after all continuous attributes being discretized, or $\mathcal{S}$ has been fully covered by $\mathcal{R}$, the rule creation process terminates.

Note that it is not very expensive to get an adequate subset $\mathcal{H}$, because "experience" of previous search can help to alleviate the computation of following search. For example, after discretizing a continuous attribute to a new category attribute a_k, we need only search for subsets containing a_k because other possible subsets have already been examined. On the other side, following observation is noteworthy that when the continuous attributes convey the most information while the category attributes convey little information, the rules extracted by STARE may be inaccurate. However, we believe that STARE is still competitive to other rule extraction approaches under such circumstances, because such kind of problem is difficult for rule extraction in nature.

4.4 Priority Formation

In STARE, if a new rule r is extracted, the examples covered by it are deleted from the example set $\mathcal{S}$. In other words, examples being covered by $\mathcal{R}$ is not used in further rule extraction. We name the part of instance space that has been covered by $\mathcal{R}$ as "known space", and name the rest part of instance space as "unknown space". It is obvious that the known space is always growing and the unknown space is always shrinking while the rule set $\mathcal{R}$ maturing. This results that the computational complexities of both continuous attribute discretization and attribute subset search are gradually lowering. This also requires that our rules be expressed in a priority format. The earlier a rule being extracted, the higher its priority is. The reason is that the rules can be viewed as being extracted from a series of unknown spaces where the later ones are subsets of the earlier ones. Moreover, the priority rule form is also the requirement of our continuous attribute processing. The reason is that since the continuous attributes are discretized one by one and the unknown space is always shrinking, the working areas of the rules involving different continuous attributes are definitely different.

Such kind of rule format has some advantages due to that the rules are matched by priority when they are in use. Firstly, the rules can have concise appearance because the rules with higher priority implicitly work as rule antecedents for the ones with lower priority. Secondly, some rules can not be extracted without the help of earlier extracted ones because the number of explicit rule antecedents is limited to 3 in our rule creation process as mentioned above. Thirdly, when common rules are in use, a conflict check must be executed to prevent specific rules from being replaced by general ones. Since priority rules has already contained order information, the time-consuming conflict check process can be omitted.

4.5 Fidelity Evaluation

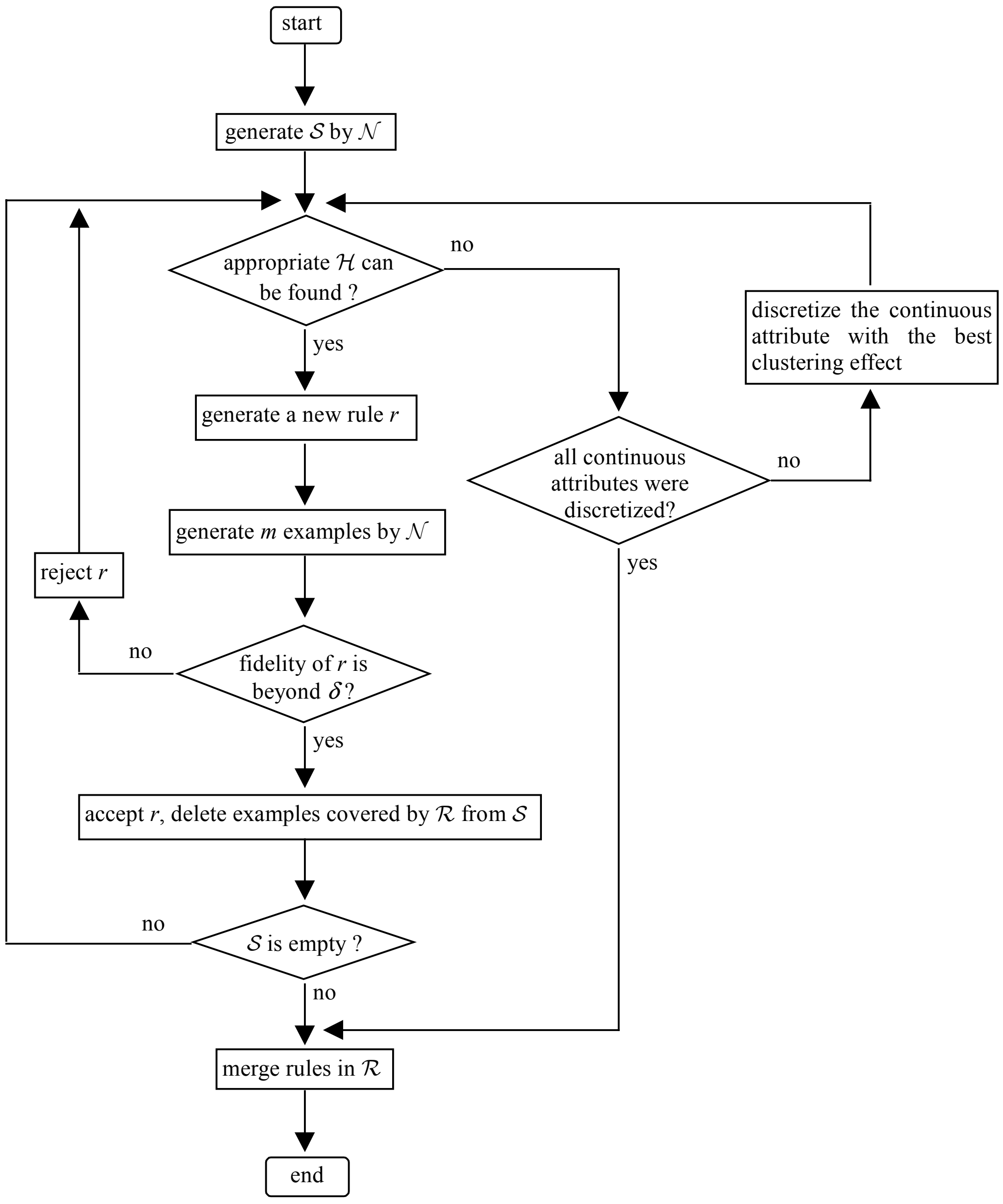
Since STARE only creates rules from category attributes, the dimension of the unknown space can be measured as the number of category attributes. Thus, discretizing continuous attributes can be viewed as increasing the dimension of unknown space when the rule extraction task is too difficult to be accomplished. Such dimension-increasing techniques have been proved to be valid in solving many complicated problems [35]. However, it also introduces a new problem. Since the unknown space with lower dimension is transformed to a space with higher dimension, the distribution of examples in $\mathcal{S}$ may be "distorted". In other words, the examples may not uniformly distribute across the new unknown space. Therefore the extracted rules may only be valid for a part of the unknown space. In order to solve this problem, statistics is introduced.

The most recently created rule r is evaluated before coming into $\mathcal{R}$ as a member. This is used to examine the fidelity of r, that is, the percentage of examples for which rule r and neural network $\mathcal{N}$ make the same classification. At first, m examples are generated by $\mathcal{N}$ in the same way as that used in generating $\mathcal{S}$. Among those examples, the ones being covered by $\mathcal{R}$ is repeatedly replaced by newly generated ones until none of the m examples is covered by $\mathcal{R}$. Then, rule r is used to classify the m examples. If the accuracy of r is beyond a pre-set lower bound δ, r is accepted to be a new member of $\mathcal{R}$. Else r is rejected and another rule is to be created. Considering the m examples are generated by $\mathcal{N}$ and they can represent the function of $\mathcal{N}$ in present unknown space, r 's accuracy is actually its fidelity. Experiments show that when m equals the size of $\mathcal{S}$ of that time, very accurate rules can be extracted. Note that the size of the extracted rule set can be determined by the user through setting different values to δ. Since big δ results in high frequency of rule rejection, it is obvious that in general the smaller δ is, the bigger the extracted rule set is.

In summary, the flowchart of the STARE approach is sketched in Figure 3, where “merge rules in $\mathcal{R}$” means the merger of the rules that have the same rule consequent and successive priorities in $\mathcal{R}$.

4.6 Experimental Testing

We have performed two series of experiments on STARE. Firstly, we regard STARE as a neural network



start
generate $\mathcal{S}$ by $\mathcal{N}$
appropriate $\mathcal{H}$ can be found ?
no
yes
discretize the continuous attribute with the best clustering effect
generate a new rule r
all continuous attributes were discretized?
no
generate m examples by $\mathcal{N}$
reject r
yes
no
fidelity of r is beyond δ?
yes
accept r, delete examples covered by $\mathcal{R}$ from $\mathcal{S}$
no
$\mathcal{S}$ is empty ?
no
merge rules in $\mathcal{R}$
end

Figure 3: Flowchart of the STARE approach

rule extraction approach and compare it with Craven and Shavlik's TREPAN [24]. Secondly, we regard STARE as a rule learning algorithm and compare it with C4.5 rules [36]. The software version of C4.5 is *See5.0 for Windows* (see http://www.rulequest.com). There are 6 data sets, among which 5 come from UCI machine learning repository [37], including *1985 Auto Imports*, *Credit Screening*, *Hepatitis*, *Iris Plant*, and *Lung Cancer*, and the rest one (*IS Fault Diagnosis*) comes from an informational software fault diagnosis project [38]. Since we do not want to compare the ability of those algorithms in dealing with incomplete information, the examples having missing values are neglected. In both series of experiments, We always regard the original data sets as test sets, and introduce a 5% random noise to the original examples to construct corresponding training sets. 5% random noise means regarding the data set as a table and randomly selecting 5% fields to set to random attribute values.

In the first series of experiments, since both STARE and TREPAN put no special claim for neural network architecture or training algorithm, we use a prevailing BP algorithm [39] to train a single-hidden-layer network for each problem, then use both approaches to extract rules from the BP networks. The test set fidelity of the STARE rules and TREPAN rules are compared because we believe that fidelity is the most important issue for neural network rule extraction approaches. The experimental results are tabulated in Table 1. Although the number of STARE rules is slightly more than that of TREPAN rules, pairwise one-tailed *t*-tests indicate that the fidelity of STARE rules is significantly better than that of TREPAN rules at the 95% confidence level. In fact, the rule number is not comparable because the rules are expressed in different formats, that is, STARE extracts priority rules that have at most 3 antecedents while TREPAN extracts M-OF-N rules. And as we have mentioned before, STARE extracts more rules partially due to the increasing of comprehensibility.

Table 1: Comparison of STARE and TREPAN

Data set	*Rule number*		*Test set fidelity*	
	TREPAN	*STARE*	*TREPAN*	*STARE*
1985 Auto Imports	27	36	93.1%	100%
Credit Screening	32	38	96.6%	98.3%
Hepatitis	11	12	92.5%	100%
Iris Plant	10	13	92.0%	97.3%
Lung Cancer	7	8	92.6%	100%
IS Fault Diagnosis	33	39	91.2%	100%

In the second series of experiments, we mainly compare the test set accuracy of STARE rules and C4.5 rules because we believe that predictive accuracy is the most important factor for rule learning algorithms. We do not compare the time cost because if considering the time used for training the BPs, the time used for generating STARE rules is definitely more than that used for generating C4.5 rules. However, since STARE is not bound with BP, its time cost is variable when adopting different neural classifiers. The experimental results are shown in Table 2. Pairwise one-tailed *t*-tests indicate that in most cases the predictive accuracy of STARE rules is significantly better than that of C4.5 rules. We believe that it is because STARE rules have profited from the generalization ability of the trained neural networks. As a complementarity, Table 2 also includes the predictive accuracy of the BPs.

Table 2: Comparison of STARE rules and C4.5 rules

Data set	*Rule number*		*Test set accuracy*		
	C4.5 rule	*STARE rule*	*C4.5 rule*	*STARE rule*	*BP*
1985 Auto Imports	20	36	95.6%	98.7%	98.7%
Credit Screening	13	38	90.5%	96.8%	97.9%
Hepatitis	6	12	95.0%	96.3%	96.3%
Iris Plant	4	13	97.3%	96.7%	98.7%
Lung Cancer	5	8	88.9%	85.2%	85.2%
IS Fault Diagnosis	21	39	78.7%	86.9%	86.9%

5. Applications

In Section 4, experimental results were presented on the STARE rule extraction approach. In this section, two applications built on NEUCRUM are briefly presented, which show that NEUCRUM and its implementation described in this paper work well in many domains.

The first application is in the area of weather forecasting. The objective is to mine rules that can predict typhoon type from future nephogram data. The database we used is provided by Jiangsu Observatory, P. R. China. After manual data selection and data cleansing, there are 5,327 records each is associated with 13 attributes, among which are 5 category attributes and 8 continuous attributes. The attributes and their value range are tabulated in Table 3 where the category attribute F_0 is the class label of each record.

Table 3: Attributes and value ranges of typhoon prediction task

Field	*Meaning*	*Value Range*
F_0	type	high pressure inshore, subtropical pressure, west wind chamfer, no chamfer
F_1	south pressure greater than north	no, yes
F_2	588 line west ridge longitude	west of 116E, 116E~120E, 120E~127E, east of 127E
F_3	west wind chamfer beyond 35N	None, 104E~120E, out of 104E~120E
F_4	central value of the nearest H-P ring	> 5920 gpm, ≤ 5920 gpm
F_5	latitude span from center to eastern 588 line	0~13
F_6	central latitude change in 24 hours	0~40
F_7	latitude span from center to subtropical ridge	0~15
F_8	subtropical pressure center latitude	0~50
F_9	subtropical pressure center longitude	90~179
F_{10}	longitude begin	113.5~123
F_{11}	latitude begin	21.5~33
F_{12}	north bound of subtropical pressure 588 line	22~34.1

All the 5,327 records are transformed to a representation that is fit for neural learning. Then a FANNC network is trained and STARE is employed to extract symbolic rules. 214 rules are extracted in all. After translating those rules into a style that is very close to natural language, we invite an experienced senior nephogram analyst to help us evaluate those rules. Erroneous and trival rules are discarded. The final rules are re-organized and shown in Table 4. Since the experienced senior nephogram analyst thinks those rules are correct, novel, potential useful and comprehensible, we believe that this mining task built on NEUCRUM framework is successfully accomplished.

Table 4: Rules mined in typhoon prediction task

Index	*Rule*
1	(588 line west ridge longitude = 116E~120E) AND (west wind chamfer beyond 35N = 104E~120E) → subtropical pressure
2	(588 line west ridge longitude = west of 116E) AND (central latitude change in 24 hours < 4.455) → no chamfer
3	(588 line west ridge longitude = 116E~120E) → west wind chamfer
4	(central value of the nearest H-P ring > 5920 gpm) →high pressure inshore

The second application is in the area of criminal categorization. The objective is to mine rules that can reveal the characteristic of economic criminals, which is helpful for future criminal identification. The database we used is provided by Jiangsu Superior Court, P. R. China. After manual data selection and data cleansing, there are 1,669 records each is associated with 6 attributes, among which are 5 category attributes and 1 continuous attribute. The attributes and their value range are tabulated in Table 5 where the category attribute F_0 is the class label of each record.

Table 5: Attributes and value ranges of criminal identification task

Field	*Meaning*	*Value Range*
F_0	type	corruption, defalcation, bribery
F_1	sex	male, female
F_2	educational background	beyond Master, Bachelor, junior college grad, senior high school grad, below junior high school grad
F_3	occupation	master of enterprise, master of government office, master of mass organization, business agent, commercial missionary, labourer, other
F_4	position	director general, division chief, section chief, clerk
F_5	age	18~80

All the 1,669 records are transformed to a representation that is fit for neural learning. Then a FANNC network is trained and STARE is employed to extract symbolic rules. 62 rules are extracted in all. After translating those rules into a style that is very close to natural language, we invite an experienced law officer to help us evaluate those rules. The final rules are re-organized and shown in Table 6, where <*dangerous in*> is an operator designed to combine the rule antecedents and the rule subsequent in this task. Since the law officer thinks those rules are irradiative for future economic criminal identification, we believe this mining task built on NEUCRUM is successfully accomplished.

Table 6: Rules mined in criminal identification task

Index	*Rule*
1	(master of enterprise) AND (division chief) <*dangerous in*> corruption
2	(commercial missionary) AND (division chief) AND (senior high school grad) AND (27 < age < 35) <*dangerous in*> defalcation
3	(master of government office) AND (male) AND (48 < age < 63) <*dangerous in*> bribery

6. Conclusions

Since most neural algorithms perform tardily iterative learning and the knowledge learned by trained neural networks are concealed in a large number of connections, neural technology is not popular in data mining area. In this paper, a general neural framework named NEUCRUM designed for classification rule mining is proposed. One of its possible implementation is also presented, which has two key components. The first key component is a fast neural classifier named FANNC that performs one-pass incremental learning with strong generalization ability. The second key component is a novel rule extraction approach named STARE that is proposed in this paper. Experiments show that STARE is a good rule extraction approach as well as a good rule learning algorithm. Moreover, this paper describes two real world classification rule mining applications built upon NEUCRUM framework, which demonstrates that NEUCRUM and its implementation presented in this paper work quite well in many domains.

Note that NEUCRUM is a general framework instead of a specific method. This means that its implementation presented in this paper is not the only solution. In fact, it is very easy to implement NEUCRUM with other neural classifiers and rule extraction approaches, where the neural classifier must be fast enough and the rule extraction approach must have the ability of extracting accurate and comprehensible rules.

Moreover, it is also possible to regard NEUCRUM as a general neural framework of data mining, provided that fast neural algorithms and rule extraction approaches that are eligible for the mining objective are available. For example, if we have a fast neural regression estimator and an approach that can extract comprehensible rules from neural regression estimators, we can build time-series mining systems based upon NEUCRUM. This is a work that we want to do in the near future. We also wish to develop techniques to measure the strength of interaction among different continuous attributes so that several continuous attributes that have strong interaction can be simultaneously discretized, which enables the exploitation of useful interaction while inhibiting the impact of useless interaction.

Acknowledgements

The authors are indebted to their colleagues at MLNN Group, AI Lab, Nanjing Univ., for their suggestive discussion and zealous help. The authors are also grateful to Jiangsu Observatory and Jiangsu Superior Court for their kind assistance. Earnest thanks are due to Dr. Andries P. Engelbrecht and the anonymous reviewers for their helpful comments and suggestions. The National Natural Science Foundation of P. R. China and the Natural Science Foundation of Jiangsu Province, P. R. China, supported this research.

References

[1] R. Agrawal, T. Imielinski, A. Swami, Database Mining: A Performance Perspective. *IEEE Transactions on Knowledge and Data Engineering*, **5**(6): 914-925, 1993.

[2] J. Han, *Data Mining Techniques*. Tutorial at the 1996 ACM-SIGMOD International Conference on Management of Data, Montreal, Canada, 1996.

[3] B. Kero, L. Russell, S. Tsur, W.-M. Shen, An Overview of Database Mining Technologies. *Proc. of the KDD Workshop in the 4th International Conference on Deductive and Object-Oriented Databases*, Singapore, 1-8, 1995.

[4] H. Lu, R. Setiono, H. Liu, Effective Data Mining Using Neural Networks. *IEEE Transactions on Knowledge and Data Engineering*, **8**(6): 957-961, 1996.

[5] M. W. Craven, J. W. Shavlik, Using Neural Networks for Data Mining. *Future Generation Computer Systems*, **13**(2-3): 211-229, 1997.

[6] D. Shalvi, N. DeClaris, An Unsupervised Neural Network Approach to Medical Data Mining Techniques. *Proc. of the 1998 IEEE International Joint Conference on Neural Networks*, Anchorage, AL, Vol.1, 171-176, 1998.

[7] D. G. Esp, M. Carrillo, A. J. McGrail, Data Mining Applied to Transformer Oil Analysis Data. *Proc. of the 1998 IEEE International Symposium on Electrical Insulation*, Vol.1, 12-15, 1998.

[8] R. Brause, T. Langsdorf, M. Hepp, Neural Data Mining for Credit Card Fraud Detection. *Proc. of the 11th IEEE International Conference on Tools with Artificial Intelligence*, Evanston, IL, 103-106, 1999.

[9] C.-K. Shin, U. T. Yun, H. K. Kim, S. C. Park, A Hybrid Approach of Neural Network and Memory-Based Learning to Data Mining. *IEEE Transactions on Neural Networks,* **11**(3): 637-646, 2000.

[10] R. H. Kewley, M. J. Embrechts, C. Breneman, Data Strip Mining for the Virtual Design of Pharmaceuticals with Neural Networks. *IEEE Transactions on Neural Networks*, **11**(3): 668-679, 2000.

[11] S. M. Weiss, C. A. Kulikowski, *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*, San Mateo, CA: Morgan Kaufmann, 1991.

[12] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, Knowledge Discovery and Data mining: Towards A Unifying Framework. *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, OR, 82-88, 1996.

[13] M.-A. Hernandez, S. J. Stolfo, Real-World Data Is Dirty: Data Cleansing and the Merge/Purge Problem. *Data Mining and Knowledge Discovery*, **2**(1): 9-37, 1998.

[14] S. I. Gallant, Connectionist Expert Systems. *Communications of the ACM*, **31**(1): 152-169, 1988.

[15] R. Andrews, J. Diederich, A. B. Tickle, Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge-Based Systems*, **8**(6): 373-389, 1995.

[16] A. B. Tickle, R. Andrews, M. Golea, J. Diederich, The Truth Will Come to Light: Directions and Challenges in Extracting the Knowledge Embedded within Trained Artificial Neural Networks. *IEEE Transactions on Neural Networks*, **9**(6): 1057-1067, 1998.

[17] L. Fu, Rule Learning by Searching on Adapted Nets. *Proc. of the 9th National Conference on*

Artificial Intelligence, Anaheim, CA, 590-595, 1991.

[18]G. G. Towell, J. W. Shavlik, The Extraction of Refined Rules from Knowledge-based Neural Networks. *Machine Learning*, **13**(1): 71-101, 1993.

[19]S. Sestito, T. Dillon, Knowledge Acquisition of Conjunctive Rules Using Multilayered Neural Networks. *International Journal of Intelligent Systems*, **8**(7): 779-805, 1993.

[20]R. Setiono, Extracting Rules from Neural Networks by Pruning and Hidden-Unit Splitting. *Neural Computation*, **9**(1): 205-225, 1997.

[21]R. Krishnan, G. Sivakumar, P. Bhattacharya, A Search Technique for Rule Extraction from Trained Neural Networks. *Pattern Recognition Letters*, **20**(3): 273-280, 1999.

[22]H. Tsukimoto, Extracting Rules from Trained Neural Networks. *IEEE Transactions on Neural Networks*, **11**(2): 377-389, 2000.

[23]K. Saito, R. Nakano, Rule Extraction from Facts and Neural Networks. *Proc. of the International Neural Network Conference*, Paris, France, 379-382, 1990.

[24]M. W. Craven, J. W. Shavlik, Using Sampling and Queries to Extract Rules from Trained Neural Networks. *Proc. of the 11th International Conference on Machine Learning*, New Brunswick, NJ, 37-45, 1994.

[25]S. Thrun, Extracting Rules from Artificial Neural Networks with Distributed Representations. *Advances in Neural Information Processing Systems*, Vol.7, 505-512, 1995.

[26]J. M. Benitez, J. L. Castro, I. Requena, Are Artificial Neural Networks Black Boxes? *IEEE Transactions on Neural Networks*, **8**(5): 1156-1164, 1997.

[27]W. Duch, R. Adamczak, K. Grabczewski, Extraction of Logical Rules from Neural Networks. *Neural Processing Letters*, **7**(3): 211-219, 1998.

[28]I. A. Taha, J. Ghosh, Symbolic Interpretation of Artificial Neural Networks. *IEEE Transactions on Knowledge and Data Engineering*, **11**(3): 448-463, 1999.

[29]T. Arciszewski, Engineering Semantic Evaluation of Decision Rules. *Journal of Intelligent and Fuzzy Systems*, **5**(3): 285-295, 1997.

[30]N. Lavrac, Selected Techniques for Data Mining in Medicine. *Artificial Intelligence in Medicine*, **16**(1): 3-23, 1999.

[31]Z. Zhou, S. Chen, C. Chen, FANNC: A Fast Adaptive Neural Network Classifier. *Knowledge and Information Systems*, **2**(1): 115-129, 2000.

[32]S. Grossberg, Adaptive Pattern Classification and Universal Recoding, I: Parallel Development and Coding of Neural Feature Detectors. *Biological Cybernetics*, **23**(2): 121-134, 1976.

[33]D. Wasserman, *Advanced Methods in Neural Computing*, NY: Van Nostrand Reinhold, 1993.

[34]R. Kerber, Chi-Merge: Discretization of Numeric Attributes. *Proc. of the 9th National Conference on Artificial Intelligence*, San Jose, CA, 123-128, 1992.

[35]V. Vapnik, *The Nature of Statistical Learning Theory*, NY: Springer-Verlag, 1995.

[36]J. R. Quinlan, *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1993.

[37]C. Blake, E. Keogh, C. J. Merz, *UCI Repository of Machine Learning Databases* [http://www.ics.uci.edu/~mlearn/MLRepository.htm]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[38]J. He, Z. Zhou, Z. Chen, An Adaptive Neural Network Approach to Fault Diagnosis. *Proc. of the 5th International Conference for Young Computer Scientists*, Nanjing, China, 151-152, 1999.

[39]D. Rumelhart, G. Hinton, R. Williams, Learning Representation by Backpropagating Errors. *Nature*, **323**(9): 318-362, 1986.