Association-Based Evolution of Comprehensible Neural Logic Networks

Henry Wai-Kit Chia and Chew-Lim Tan

School of Computing,
National University of Singapore,
3 Science Drive 2, Singapore 117543
{hchia, tancl}@comp.nus.edu.sg

Abstract. Neural Logic Network (*Neulonet*) learning has been successfully used in emulating complex human reasoning processes. One recent implementation generates a single large *neulonet* via genetic programming using an accuracy-based fitness measure. However, in terms of human comprehensibility and amenability during logic inference, evolving multiple compact *neulonets* are preferred. The present work realizes this by adopting associative-classification measures of confidence and support as part of the fitness computation. The evolved *neulonets* are combined together to form an eventual *macro-classifier*. Empirical study shows that associative classification integrated with *neulonet* learning performs better than general association-based classifiers in terms of higher accuracies and smaller rule sets. This is primarily due to the richness in logic expression inherent in the *neulonet* learning paradigm.

1 Introduction

Neural Logic Network (*Neulonet*) learning is an amalgam of neural network and expert system concepts [1, 2]. Its novelty lies in its ability to emulate human decision logic via rudimentary *net rules* that represent the core human logic operations. These richer logic operations can be used to represent common human decision making processes such as a *priority* operation that depicts the notion of assigning varying degrees of bias to different decision factors, and a *majority* operation that involves some strategy of vote-counting. Within the realm of pattern classification, human logic operations supplement the standard boolean logic operations of conjunction, disjunction and negation, so as to allow for a neater and more elegant expression of complex logic in a given problem domain.

Different approaches have been devised for *neulonet* learning. These include the resolution of a set of inequalities [1], learning via neural network back-propagation training [1, 2], and more recently, *neulonet* evolution using genetic programming (GP) [3, 4]. In particular, GP *neulonet* learning facilitates the generation of a more effective pattern classifier as compared to the genetic evolution of logic based neural networks representing the standard boolean logic operations [5]. Another motivating factor for advocating the evolutionary approach as compared to back-propagation in *neulonet* learning is the fact that the weights of the network are kept unchanged throughout the genetic evolution process, thereby facilitating the straightforward extraction of logic rules.

For the genetically-programmed *neulonet* learning methodology to qualify as an ideal classification system in data mining, it needs to satisfy the data mining goals of *prediction* and *description* as prescribed in the article by Fayyad [6]. *Prediction* pertains to the ability of the system in finding patterns or rules that accurately forecast the future or unseen behaviour of related entities. On the other hand, these discovered patterns would be deemed to fulfill the *descriptive* goal if they can be represented in a human-comprehensible form to the user. In the earlier work [4], this latter goal was somewhat achieved by including parsimony as a fitness criterion which allowed for the evolution of a reasonably compact *neulonet* solution. This, in turn, corresponds to the extraction of a smaller set of human logic rules, which would ultimately render rule-inferencing more amenable towards human comprehension. However, it has to be noted that these rules can only be interpreted as an entire connecting set. Although rule interdependence promotes the capacity to infer the relationship between rules, a deeply-nested *neulonet* that corresponds to a high degree of interdependence amongst the extracted rules will inevitably hamper the layman understanding of the inherent logic in the problem domain.

The present work ensues with the premise of improving the comprehensibility in genetically-programmed *neulonet* learning by evolving multiple compact *neulonets* based on confidence and support measures adopted in association-based classification. This paper begins with an overview of neural logic networks and how *neulonet* learning is accomplished under the genetic programming paradigm. The rule extraction procedure is described with subsequent analysis of the rules to illustrate the expressiveness of human logic operations in terms of knowledge representation. The integration of GP *neulonet* learning into associative classification is explored, and empirical results presented to substantiate the advantages of including human logic operations in classification tasks.

2 Neural Logic Network - *Neulonet*

A *neulonet* has an ordered pair of numbers associated with each node and connection as shown in figure 1. Let Q be the output node and P_1, P_2, $\ldots P_N$ be input nodes. Let values associated with the node P_i be denoted by $(a_i,\, b_i)$, and the weight for the connection from P_i to Q be $(\alpha_i,\, \beta_i)$. The ordered pair for each node takes one of three values, namely, (1,0) for "true", (0,1) for "false" and (0,0) for "don't know". (1,1) is undefined. Equation (1) defines the activation function at the output Q with λ as the threshold, usually set to 1.

$$Act(Q) = \begin{cases} (1,0) \text{ if } \sum\limits_{i=1}^{N} (a_i\alpha_i - b_i\beta_i) \geq \lambda \\ (0,1) \text{ if } \sum\limits_{i=1}^{N} (a_i\alpha_i - b_i\beta_i) \leq -\lambda \\ (0,0) \text{ otherwise.} \end{cases} \tag{1}$$

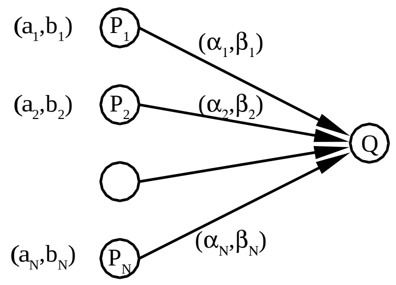

Fig. 1. Schema of a Neural Logic Network - *Neulonet.*

A wide variety of human decision logic, which are often too complex to be expressed neatly using standard boolean logic, can be represented using rudimentary *neulonets* with different sets of connecting weights. These *net rules* can be generally divided into five broad categories as shown in figure 2.

3 Neural Logic Network Composition using Genetic Programming

Net rules can be combined to form composite *neulonets* to realize more complicated decisions as shown in figure 3. This can be achieved using genetic programming [7], which is an extension of the conventional genetic algorithm where instead of subjecting bit patterns to genetic evolution, the individuals in the gene population are computer programs. In the context of *neulonet* evolution, these computer programs are represented in the form of *neulonets.* A brief description of the genetic *neulonet* construction process (detailed in [3]) is presented below.

3.1 Neulonet Structure Undergoing Adaptation

Adaptation is performed on the population of *neulonets*, each being a recursive composition of *net rules* and input terminals in a tree-like structure. The input terminals are the attributes of the data set, as well as bias decision nodes depicting default true, false and unknown decisions. The initial population of *neulonets* is generated in a similar fashion as Koza's "ramped-half-and-half" generative method [7] that accounts for both depth and fan-out of the *neulonet.*

3.2 Genetic Operations

We describe three fitness-proportionate genetic operations for modifying the *neulonet* structure undergoing evolution. The reproduction operation first selects a single *neulonet* from the population. The selected individual is then copied to the new population. The crossover operation involves swapping the chosen parts of two *neulonets* with constraints imposed on the swapping process to preserve

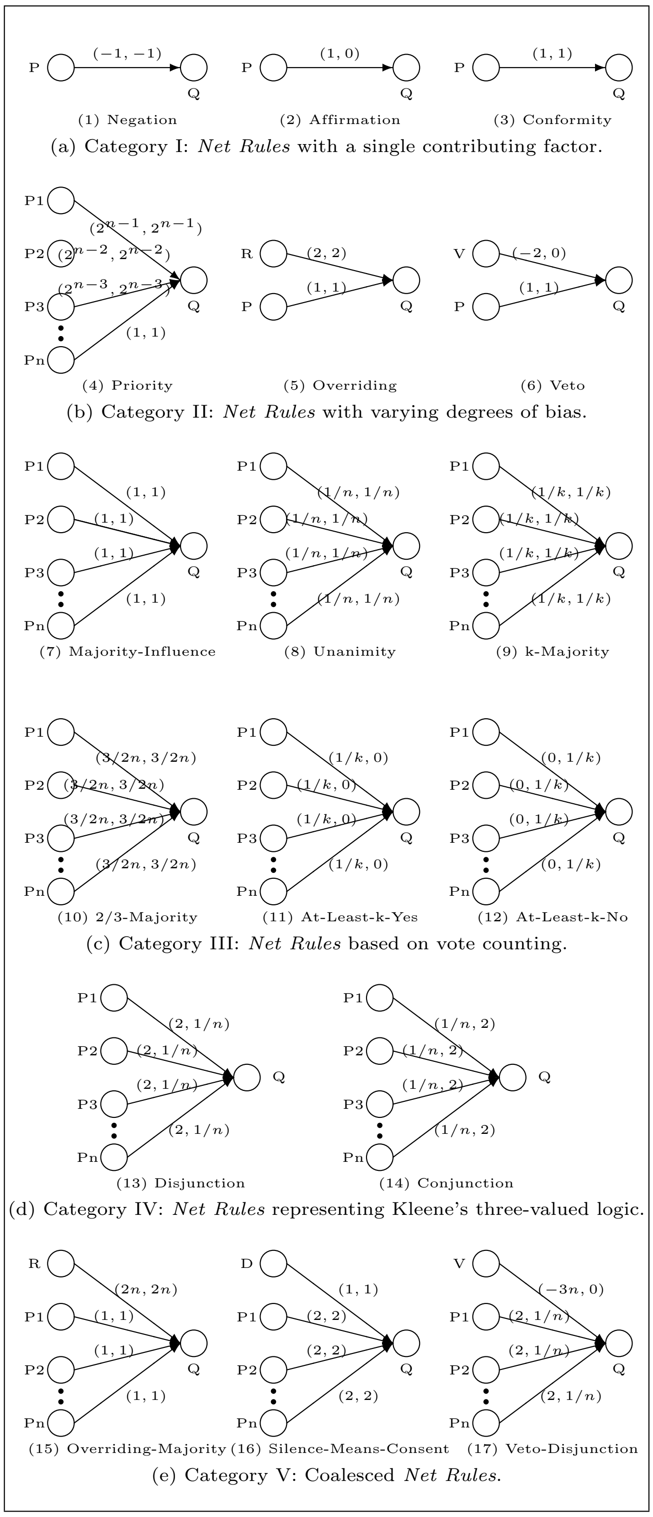

Fig. 2. Library of *net rules* divided into five broad categories

the syntactic integrity. For instance, swapping should not leave any dangling intermediate output nodes as such nodes will not be able to receive input values during firing. The mutation operation involves changes to a chosen *neulonet*. A random mutation point is picked such that the *neulonet* whose root is the mutation point is replaced by another randomly generated *neulonet*.

3.3 Fitness Measure and Termination Criterion

Each *neulonet* in the population is assigned a normalized fitness measure $f(\epsilon, \sigma; \kappa)$ based on the errors produced by the *neulonet*, ϵ, as well as the size of the *neulonet*, σ as defined in equation (2). The fitter the *neulonet*, the larger will be its fitness measure. The factor, $\kappa \in [0, 1]$, is used to weigh the effects of accuracy over size in the fitness measure. A higher value for κ places more emphasis on finding an accurate solution at the expense of the size of the *neulonet*. Moreover, the empirical fine-tuning of an appropriate κ value allows the extraction of a simpler set of *net rules* that avoids unnecessary complications, so as to provide a satisfactory overall generalization capability. Note that σ_{min} in equation (2), denotes the smallest possible size of a *neulonet*.

$$f[\epsilon, \sigma; \kappa] = \frac{1}{1 + \kappa\epsilon + (1 - \kappa)(\sigma - \sigma_{min})} \tag{2}$$

Termination of evolution is controlled using a parameter that specifies a period in which the termination criterion is examined upon its elapse. The test for termination involves recording the current generation's fittest *neulonet*, and comparing it against the preceding record. Termination transpires when there is no improvement in the classification accuracy and size of the current recorded *neulonet*, in which case, it is designated as the solution to the problem domain.

4 Rule Extraction

The subsequent extraction of logic rules is straightforward as each constituent *net rule* of an evolved solution *neulonet* fully expresses the human logic in use. These human logic *net rules* can be easily identified from any given *neulonet*. We use the US Congressional Voting Records Database [8] to illustrate rule-based learning using *net rules*. This data set consists of 435 instances and 16 boolean attributes with unknown or missing values. Each instance is classified as either *republican* or *democrat*. GP *neulonet* learning generated a solution with a manageable number of rules while being sufficiently accurate. A desirable result was achieved by setting κ to 0.94. The evolved *neulonet* solution is shown in figure 3 with a classification accuracy of 97.7%. The set of logic rules extracted for this *neulonet* is given in table 2. Note that a "true" outcome denotes the *republican* class.

Observe that the bias-based "Veto" *net rule* **Q4** has rule **Q2** as the veto factor, which implies that **Q2** plays a more important role than **Q3**. This veto power

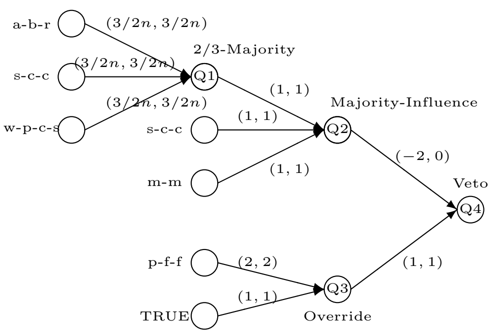

Fig. 3. Evolved *Neulonet* solution for voting records.

Q1 ⇐ 2-out-of-3-Majority[a-b-r=y, s-c-c=y, w-p-c-s=y]
Q2 ⇐ MajorityInfluence[**Q1**, s-c-c=y, m-m=y]
Q3 ⇐ Overriding[p-f-f=y, TRUE]
Q4 ⇐ Veto[**Q2**,**Q3**]

Table 2. Extracted *net rules* from the voting records data.

will come into effect only when **Q2** returns a positive decision. Application of backward induction indicates that such a situation occurs only if the majority of the contributing factors of rule **Q2** are true. One instance occurs when *synfuels-corporation-cutback(s-c-c)* and *mx-missile(m-m)* are true. The contributing factors comprising rules **Q1** and **Q2** can be viewed as belonging to two different classes of decision makers. The fact that rule **Q1** forms part of rule **Q2** leads us to deduce the relative importance between the classes of contributing factors in both the rules. Moreover, the presence of the attribute *s-c-c* in both rules signifies its higher level of participation in the overall decision. *Net rules* **Q1** and **Q2** provide a concise way of expressing the appropriate rule activations, rather than enumerating every possible combination of decisions factors for each rule activation using separate rules as in the case of general production rules. Finally, if the outcome of **Q2** is false or unknown, then the higher priority *physician-fee-freeze(p-f-f)* attribute in rule **Q3** will influence the final outcome, provided that this attribute value is deterministic, i.e. “true” or “false”. Otherwise, rule **Q4** will assume a true outcome. GP *neulonet* learning could be used to exploit the “richer” thought processes that are extensively used for human reasoning, particularly in real-world problem domains.

An intrinsic feature of *neulonet* rule extraction is the high degree of interdependence among the rules. The set of rules in table 2 has to be interpreted in its entirety as a connecting set in order to discover the underlying logic of the problem domain. This interdependence among the set of logic rules captures the logic relationship between rules. However, empirical observation shows that this same property becomes more of a liability when the *net rules* are nested at more than three levels deep. An ideal classification system should, in essence, be able to exploit this interdependence property albeit to a lesser degree.

5 Incorporating *Neulonet* Learning into Associative Classification

Associative classification is the technique of employing association rule mining concepts [9] for classification, and represents an alternative to purely accuracy-based classifier systems. Among the many associative classifiers in the literature, the CBA system [10] is of particular interest. CBA comprises two distinct stages: a rule-generation stage followed by a classifier-building stage. In the rule-generation stage, all independent *class association rules* (CARs) are generated. The classifier-building stage, on the other hand, provides a methodology to group these CARs together to form an eventual classifier. Clearly, this two stage process is easily adaptable for the proposed *neulonet* associative classification technique. Rather than generating the CARs that only involve an implicit conjunction operator, the novelty lies in evolving compact *neulonet association rules* (NARs) in their place. These NARs would then be utilized for classifier building without any major modifications to the existing classifier-building algorithm in CBA. On the microscopic level, an NAR would be able to exploit the interdependence property in *neulonet* rule-inference, while on the macroscopic level, the relevant NARs could collectively form an accurate classifier.

5.1 Extending Notions of Confidence and Support to *Neulonet* Learning

Association rule mining is a common technique used for market basket analysis [9]. Assigned to every assocation rule is the two important notions of *confidence* and *support*. Basically, the *confidence* of a rule provides a measure on the accuracy of the rule. On the other hand, the *support* of a rule is an indication of the amount of data that is consistent with the rule. In order to incorporate genetically-programmed *neulonet* learning into the associative classification domain, these two notions have to be accounted for in the fitness measure during *neulonet* evolution.

In CBA, a class association rule is expressed in the form

$$a_1, a_2, \ldots, a_k \rightarrow y$$

where each of the terms a_i denotes a boolean attibute, and y denotes the class label. For ease of explanation, denote the LHS of the above expression as the *condset* (the set of conditional attributes). For each CAR, the condition support count, $condsupCount$, is the number of cases in the data set $\mathbf{D}$ that satisfy the *condset*. On the other hand, the rule support count, $rulesupCount$, is the number of cases in $\mathbf{D}$ that satisfy the *condset*, with the additional requirement that each of these cases should be labeled with class y. The support, S, and confidence, C, of a CAR is defined as follows:

$$S = \frac{rulesupCount}{|\mathbf{D}|} \tag{3}$$

$$C = \frac{rulesupCount}{condsupCount} \tag{4}$$

Implicit in the *condset* of the CAR is the conjunction operator that operates on the conditional attributes. As such, the CAR can be re-expressed as

$$a_1 \wedge a_2 \wedge \ldots \wedge a_k \rightarrow y$$

In terms of *net rule* syntax, the following equivalent expression is obtained.

$$y : \mathbf{Q1} \leftarrow \texttt{Conjunction}(a_1, a_2, \ldots, a_k)$$

The label y denotes that the above *net rule* is associated with the class y, so that an outcome of "true" in **Q1** will signify the class y. In the present context, *condsupCount* and *rulesupCount* need to be redefined accordingly. For a *neulonet* generated to learn a particular class label y, *condsupCount* is defined as the number of cases in the data set **D** for which the *neulonet* solution returns a "true" outcome. Similarly, *rulesupCount* is defined as the number of cases in **D** for which the *neulonet* solution returns a "true" outcome, with the additional requirement that each of these cases must be labeled with the class y.

The usual fitness measure for *neulonet* evolution given in equation (2) is based upon the error rate ϵ and size σ of the evolved *neulonet*. In the context of associative classification, ϵ needs to be replaced by a penalty function $\Psi(S, C)$, that accounts for the support S and confidence C levels.

$$f[S, C, \sigma; \kappa] = \frac{1}{1 + \kappa\Psi(S, C) + (1 - \kappa)(\sigma - \sigma_{min})} \tag{5}$$

To ensure that the effect of the weighting factor κ is consistent across both fitness measures, the penalty function should be defined such that the range of values of $\Psi(\cdot)$ in equation (5) is consistent with that of $\epsilon \in [0, |\mathbf{D}|]$ in equation (2). Furthermore, in accordance with CBA's precedence ordering of CARs, the fitness measure should be able to express the ordering of NARs where precedence is based on the confidence level first, followed by the support level. Notice that the range of integer values for *condSupCount* and *ruleSupCount* lie in $\{0, 1, 2, \ldots, |\mathbf{D}|\}$ and $\{0, 1, 2, \ldots, condSupCount\}$ respectively. For a *condSupCount* of $|\mathbf{D}|$, the confidence level C occurs as discrete values in the range

$$C \in \left\{0, \frac{1}{|\mathbf{D}|}, \frac{2}{|\mathbf{D}|}, \ldots, \frac{|\mathbf{D}| - 1}{|\mathbf{D}|}, 1\right\}$$

Clearly, for any other value of *condSupCount* ($< |\mathbf{D}|$), the interval between two successive confidence levels will be larger than $1/|\mathbf{D}|$. The idea is therefore to shrink the range of support levels from [0,1] to $[0, 1/|\mathbf{D}|)$ instead, so that mere addition of confidence and support levels will result in a value that correlates with the precedence ordering. Define the precedence value p as

$$p(C, S) = C + S\left(\frac{1}{|\mathbf{D}| + 1}\right) \tag{6}$$

where p lies in the range $[0, (|D|+2)/(|D|+1)]$ with a higher value of p denoting a higher precedence. Since the formulation of the penalty function requires a higher precendence to correspond to a lower penalty value, define the raw penalty function[1] as

$$\overline{\Psi}(C, S) = \frac{|\mathbf{D}| + 2}{|\mathbf{D}| + 1} - p(S, C) \tag{7}$$

Finally, define the penalty function $\Psi(\cdot)$ as

$$\begin{aligned} \Psi(C, S) &= \overline{\Psi}(C, S)|\mathbf{D}| \\ &= \left[\frac{|\mathbf{D}| + 2}{|\mathbf{D}| + 1} - \left(C + \frac{S}{|\mathbf{D}| + 1}\right)\right]|\mathbf{D}| \end{aligned} \tag{8}$$

and substitute this into the modified fitness measure of equation (5).

Additionally, CBA ensures that CARs generated have support and confidence levels that are above the specified minimum support and confidence levels respectively. This same strategy is adopted during *neulonet* evolution.

5.2 Constructing the *Neulonet* Associative Classifier

As in CBA, *neulonet* associative classification involves two equivalent phases. The motivation for the first stage of *neulonet* association rule (NAR) generation (table 3) is simply to find a minimal ordered sequence of NARs that would cover the required hypothesis space. Note that for an n-class classification problem, n separate sequences of NARs are generated for resolving each class value i [11]. During the second stage of classifier building (table 4), the best NARs from all sequences will then compete to be included into the eventual classifier.

We use the Voting Records Database to analyse the NARs generated. To maintain the comprehensibility of each *neulonet association rule*, the evolved NARs are kept small at a depth of at most two, with each *net rule* having a fan-out not exceeding three. The minimum support and confidence levels were set at 0.25 and 0.5 respectively. *Neulonet* associative classification produced a classifier with 18 NARs as compared to CBA's classifier of 29 CARs. Table 5 shows an extract of the first three NARs which would have achieved an accuracy of 94.9%. In fact, the first rule **Q1** by itself would already have achieved an accuracy of 94.0% with *republican* as the default class; the layman interpretation being: if the *physician-fee-freeze (p-f-f)* attribute is true, then the decision

[1] Ideally, a *normalized* penalty function with values in the range [0,1] should be defined. However, the *raw* penalty function with the range $[0, 1 + (1/(|\mathbf{D}| + 1))]$ is an adequate approximation, especially for large data sets.

would be *republican*, since **Q1** is not satisfied. Otherwise, the decision is *democrat* when either *absorption-of-budget-resolution (a-b-r)* or *synfuels-corporation-cutback (s-c-c)* are true). In contrast, the first five CARs generated in CBA achieved a relatively lower accuracy of 93.8%. Rule-for-rule comparisons show that *neulonet* associative classification does indeed perform better than CBA as expected. This is again attributed to the more expressive human logic *net rules* used in classification. Furthermore, as the interdependence property is kept to a minimum, each NAR remains easily comprehensible in layman terms.

```
Algorithm: GenerateNARs(i)
1. Initialize queue $\mathbf{C}_i$ as empty.
2. Iteratively perform the following until data set $\mathbf{D}$ no longer contains
   any instances with class value $i$, or no other NAR can be evolved.
   2.1. Evolve a NAR that resolves class $i$.
   2.2. For each instance $\mathbf{d}$ in $\mathbf{D}$, mark $\mathbf{d}$ if
        (a) Firing NAR with $\mathbf{d}$ returns "true"; and
        (b) The class label of $\mathbf{d}$ is $i$.
   2.3. Insert NAR at end of queue $\mathbf{C}_i$.
   2.4. Remove all marked instances in $\mathbf{D}$.
3. Return queue $\mathbf{C}_i$.
```

Table 3. NAR generation for class *i*.

```
Algorithm: BuildClassifier
1. Initialize classifier $\mathbf{C}$ as empty.
2. Iteratively perform the following until all $\mathbf{C}_i$ queues are empty
   2.1. Compare all NARs at head of each $\mathbf{C}_i$ and remove NAR with highest fitness.
   2.2. For each instance $\mathbf{d}$ in data set $\mathbf{D}$, mark $\mathbf{d}$ if
        (a)  Firing NAR with $\mathbf{d}$ returns "true"; and
        (b)  The class label of $\mathbf{d}$ is the same as the class resolved by the chosen NAR.
   2.3. Perform the following if some $\mathbf{d}$ is marked.
        (i)   Insert NAR at end of $\mathbf{C}$.
        (ii)  Remove all marked instances in $\mathbf{D}$.
        (iii) Compute default class of $\mathbf{D}$ (majority class of remaining instances).
        (iv)  Compute number of errors of $\mathbf{C}$.
3. Find the first NAR in $\mathbf{C}$ with the lowest error and drop all NARs after that.
4. Add default class of last NAR to the end of $\mathbf{C}$.
5. Return classifier $\mathbf{C}$.
```

Table 4. Classifier building.

6 Empirical Study and Discussion

Empirical study is undertaken to compare the classifiers constructed from CBA with that of *Neulonet* Associative Classification using the entire library of *net rules* (NACfull) in figure 2. Furthermore, classifiers constructed by NAC using a restricted set of *net rules* (1), (13) and (14) that simulates the use of boolean logic

Resolving democrat :
Q1 ⇐ Veto-Disjunction[p-f-f=y, a-b-r=y, s-c-c=y]
Resolving republican :
Q3 ⇐ 2-out-of-3-Majority[**Q2**, p-f-f=y, s-c-c=n]
Q2 ⇐ Silence-Means-Consent[i=y, a-b-r=y, d-f-e=n]
Resolving democrat :
Q4 ⇐ Veto[a-s-t-b=n, c=n]
default class : republican

Table 5. Three NARs generated for the Voting Records data.

operations in associative classification (NACstd) will also be tested. The details of experimentation closely resemble those in [10]. Experiment results based on ten-fold cross validation on data sets from [12] are presented in table 6. Results for CBA are reproduced from [10].

From the tabulated results, it is clearly evident that *neulonet* associative classification using human logic *net rules* is generally the better choice. In fact, both NAC classifiers outperform CBA. Generally, NACs generate a smaller number of rules than CBA; there are almost twice as many CARs compared with NARs generated in the NAC classifiers. The enhanced logic expression using interdependent human logic rules is the primary contributing factor towards the construction of better classifiers. Although using a single implicit conjunction operator in general association-based classifiers makes the association rules (or CARs in the case of CBA) easy to understand, it deters the construction of accurate classifiers with minimal number of rules. On the other hand, the large variety of *net rules* and their corresponding semantics might seem complex at first, but nonetheless, these operators represent common human decision processes, and should therefore be generally acceptable. Also note that the advent of association rule mining arose due to the need for market basket analysis. As such, association-based classifiers using the implicit conjunction operator might not have the additional expressiveness and flexibility in handling the complex logic inherent in most classification data sets.

Data Set	CBA	NACfull	NACstd	Data Set	CBA	NACfull	NACstf
anneal	3.6 (34)	**0.3** (18.0)	1.0 (15.3)	horse	18.7 (97)	**14.2** (38.4)	19.4 (31.3)
auto	27.2 (54)	**16.9** (34.2)	19.4 (31.3)	hypo	1.7 (35)	1.2 (44.3)	**0.9** (46.7)
breast-w	4.2 (49)	**3.3** (40.7)	3.9 (39.1)	iono	8.2 (45)	6.5 (21.4)	**5.4** (23.3)
cleve	16.7 (78)	**16.2** (42.1)	17.8 (40.1)	iris	7.1 (5)	4.0 (6.3)	**4.0** (6.2)
crx	14.1 (142)	**13.3** (82.0)	15.5 (68.0)	labor	17.0 (12)	**6.7** (3.1)	6.7 (7.2)
diabetes	25.3 (57)	23.7 (20.8)	**23.3** (18.0)	lymph	19.6 (36)	**12.0** (21.7)	12.7 (23.8)
german	26.5 (172)	**24.8** (63.4)	25.0 (66.1)	pima	27.6 (45)	**23.8** (19.9)	24.1 (17.2)
glass	27.4 (27)	**21.5** (24.2)	23.3 (21.3)	sonar	21.7 (37)	**14.0** (21.6)	19.0 (23.1)
heart	18.5 (52)	**14.2** (38.5)	14.5 (23.6)	vehicle	31.3 (125)	**29.2** (34.7)	30.7 (31.0)
hepati	15.1 (23)	**13.6** (15.1)	13.6 (15.4)	wine	8.4 (10)	1.1 (5.1)	**0.6** (4.7)

Table 6. Experimental results depicting classification errors (%) of different associative classifiers. Figures in bold indicate the lowest error. The numbers inside brackets denotes the average number of CARs/NARs in the classifier.

7 Conclusion and Future Work

The adoption of the genetic programming paradigm has future implications in that it enables encoding of prior knowledge. Very often, this is in the form of general consensual knowledge, that is, knowledge having both high support and high confidence. Such knowledge can then be encoded as NAR(s) which could either be kept intact throughout the rule generation and classifier building process, or at most, minimally adapted. These NARs will make up the foremost rules of the classifier, leaving the *neulonet* associative classifier to come up with the latter NARs which would make up the interesting exceptions. The successful integration of GP *neulonet* learning with associative classification has also opened up a host of opportunities for similar endeavours. In fact, any data classification platform that separates rule generation from classifier construction are potential candidates for further exploration. The *net rule* library will be expanded and more elaborate form of decision logic will be tested. We envision an even more exciting horizon by integrating associative classification with fuzzy *neulonets* [1].

References

1. Teh, H.H.: Neural Logic Network, A New Class Of Neural Networks. World Scientific, Singapore (1995)
2. Tan, C.L., Quah, T.S., Teh, H.H.: An artificial neural network that models human decision making. Computer **29** (1996) 64–70
3. Tan, C.L., Chia, H.W.K.: Genetic construction of neural logic network. In: Proceedings of INNS-IEEE International Joint Conference on Neural Networks. Volume 1., Piscataway, N.J. (2001) 732–737
4. Tan, C.L., Chia, H.W.K.: Neural logic network learning using genetic programming. In: Proceedings of Seventeenth International Joint Conference on Artificial Intelligence, Menlo Park, Calif. (2001) 803–808
5. Gaudet, V.C.: Genetic programming of logic-based neural networks. In Pal, S.K., Wang, P.P., eds.: Genetic Algorithms for Pattern Recognition. CRC Press, Boca Raton (1996)
6. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI Magazine **17** (1996) 37–54
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, Mass. (1992)
8. Schlimmer, J.C.: Concept acquisition through representational adjustment. PhD thesis, Department of Information and Computer Science, University of California, Irvine, CA (1987)
9. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. (1993) 207–216
10. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining. (1998) 80–86
11. Kishore, J.K., Patnaik, L.M., Mani, V., Agrawal, V.K.: Application of genetic programming for multicategory pattern classification. IEEE Transactions on Evolutionary Computation **4** (2000) 242–258
12. Blake, C., Merz, C.: UCI repository of machine learning databases (1998)