

Quasi-morphism and Comprehensibility of Rules in Inductive Learning

Wiphada Wettayaprasit,^{1,2,3} Chidchanok Lursinsap,² and Cheehung Henry Chu¹

¹*Center for Advanced Computer Studies, The University of Louisiana at Lafayette,
Lafayette, LA 70504, U.S.A.*

²*Advanced Virtual and Intelligent Computing Research Center, Department of Mathematics,
Faculty of Science, Chulalongkorn University, Bangkok, 10330, Thailand*

³*Department of Computer Science, Faculty of Science, Prince of Songkla University,
Hatyai, Songkla, 90110, Thailand*

Email: wwettayaprasit@yahoo.com, lchidcha@chula.ac.th, cice@cacs.louisiana.edu

Abstract

We present a model of creating a hierarchical set of rules that encode generalizations and exceptions derived from induction learning. The rules use the input features directly and are therefore comprehensible to the users. Learning is performed by a feedforward neural network and the rules are extracted from the trained network. A pattern classification task is used to demonstrate the efficacy of our approach. We show that the rules have similar classification performance while being more comprehensible to the users.

Keywords: Neural networks, induction learning, rule-based systems, rule extraction.

1 Introduction

Inductive learning systems do so from examples. Decision trees [1] and neural networks are popular methods for modeling induction. Holland's classifier system [2], another model of induction, uses rules that can adapt to the world with which it interacts. The rules in a classifier system form a hierarchy that can be used to model the world with increasing precision. While this feature does not depend on the process by which a classifier system is trained, the comprehensibility of the rules does. The classifier system as originally proposed contains rules that are initialized as random binary strings and are adapted using a genetic algorithm. The

resultant rules can be as difficult to interpret as the connections and weights of a neural network.

In this paper, we propose to use a neural network to capture the information that must be learned from the environment for a given task, such as the classification of feature vectors. This is then in turn used as the "world" that is modeled by a rule-based system similar in concept to the classifier system. A default hierarchy, referred to as a quasi-morphism, of rules are extracted from the trained neural network. These rules are condition-action (or if-then) pairs, with the condition part described in terms of the input features directly, and not functions of those features. This improves the comprehensibility of the rules.

Classifier systems model the world using a hierarchy of rules. This hierarchy is used to represent exceptions to generalizations and facilitates a tradeoff between precision and generalizations. The comprehensibility of rules is an important property of rules and it is usually hampered by the need for precision and for handling exception cases. A rule that is sufficiently general, such as "if the temperature is between 293 and 298 kelvin, then the development time is 8 minutes," is comprehensible to most users. When variables are combined: "if the temperature is between 293 and 298 kelvins and the film is of type 1 and the development tank size is large, then development time is 11 minutes," the rules can still be understood. Transformations of variables not otherwise related can cloud the comprehensibility: "If one and a half times the normalized temperature minus 2.39 times the film type is greater than 4, then the development time is 8 minutes." A more natural way is to use more than one

rule, such as “If the temperature is between 293 and 298 kelvins, the development time is 8 minutes,” to describe a general, or default, situation and “if the development tank size is large, then increase the development time by 50%,” to handle exceptional cases.

Rule extraction from neural networks has been studied since the early days of neural network research. Classification frameworks for rule extraction algorithms were reviewed in [5,6]. The methods for extracting symbolic knowledge from a trained neural network can be grouped into the so called open-box and black-box approaches. The open-box approach methods are often analytical in nature. They can extract rules by directly interpreting the strengths of connection weights in the trained network [8]. Other methods extract rule by analyzing the activation values of the hidden units [3,11,12]. The black-box approach is often based on generate-and-test using observed input/output behaviors. This has been used to extract rules from fuzzy neural networks [5,6] and from recurrent networks [14].

The rule extraction process itself can be classified depending on its relationship with the network learning. Rule extraction can be considered as a preprocessing step, such as the insertion of knowledge into the network to set the bias link before training [9]. The extraction can be used to modify the network structure or its components. Examples of this approach include rule extraction by successive regularization [10] and the use of a staircase function in place of the sigmoid function [15]. Most often, rule extraction is applied after the network training is over. These postprocessing methods include most of the open-box approach methods, such as the M-of-N rule extraction [8].

Our emphasis in the present paper is on the structure of the rule set that is extracted. We use rule extraction methods that have an open-box, postprocessing approach. In the following, we elaborate on the default hierarchy of rules and the proposed procedures to set one up. In Section 3, we illustrate our method using a pattern classification task. We draw our conclusions in Section 4.

2 Quasi-morphism of rules

A rule in a classifier system provides a set of expectations that are true only so long as they are not contradicted by more specific rules. In the absence of additional information these default rules provide the best model of the situation. Rules can therefore be arranged in hierarchies ordered by default expectations based on their relations. Useful generalizations that may be drawn upon for modeling the world and those exceptions to these generalizations can be represented. Where there are few exceptions, the lower level

exceptions can accommodate them. Where there are many exceptions, generalizations will be tentative and weak. Hence, we can associate the modeling uncertainty with a generalization rule.

In a classifier system, categorizations are used to divide the world it is modeling into equivalence classes. A rule maps an equivalence class from one time step to another class in the next time step. If the categorization is a many-to-one mapping, it is a homomorphism. The rules of a classifier system are meant to model transitions in the real world. Since there are always uncertainties and exceptions to the modeling, another level is needed to correct for these exceptions. Once this new category is created, the set of rules must also be refined to account for this change. This layered model is called a quasi-morphism (or q-morphism). In a q-morphism, the upper layers provide default hierarchies while the lower level rules are evoked on exceptions. A complete model is constructed by the progressive refinement of the q-morphism.

Classifier systems and multilayered neural networks are similar from the point of view of using successive steps in transforming an input to an output. The input layer in a neural network transforms the input vector to a vector comprising the hidden layer activation values. This plays an equivalent role to a rule in the classifier system. Upper layers in a neural network transform the hidden layer activation values to other activation value vectors, ultimately to the network output. Classifier systems can therefore be viewed as generalizations of neural networks in that they do not specify the manner in which the mappings are made, whereas a neural network layer transforms an input vector to an output vector by applying a nonlinearity to the weighted sums of the input.

In a classifier system, an if-then rule is fired when its condition part is satisfied. The actions specified by the then-part of the rules at each step are matched with the conditions of the rules at the next. These actions can be considered as messages, which act as a communication channel among the rules. In a neural network, there is usually only a single upper layer to receive the input from a lower layer, so there is no explicit matching of or competition among several for the input. Nevertheless, the upper layer produces an output that varies depending on the input. In this sense, the activation patterns of the lower layer in a neural network can be viewed as the communication channel between the layers. As such, they reveal much about the information processing of the network.

A research question that arises is: How does one create a quasi-morphism of rules that are comprehensible to the users for a given inductive learning task? Our approach is that we can consider the

learning ability of a neural network to be adequate and hence the solution is to extract the quasi-morphism of comprehensible rules from the network.

2.1 Rules from neural networks

Our system first derives so-called crisp rules from the neural network. A certainty factor is then evaluated for each rule. Rules that provide certainty that the premise of the input vectors interval will represent only the elements in any given class are then determined.

While our model can be adapted to different architectures and to different tasks, we concentrate on the multilayered feedforward network and on pattern classification tasks. In our experiments, we use a fully connected architecture with an input layer that handles continuous numeric input data, a hidden layer, and an output layer. The backpropagation learning rule is used to train the network.

Consider a neural network that is trained to perform a binary classification task. The polarity of a neuron's activation value represents the neuron's decision at the level of that particular neuron. A hidden unit can therefore be considered to have learned a portion of the training data and so it has a more provincial view, while the output unit can be considered to take a more global view by integrating the multiple provincial decisions from the hidden layer neurons. Suppose we have a set of input feature vectors $[x_0 \ x_1 \ \dots \ x_{N-1}]$. Each vector generates an activation value for a particular hidden neuron. We can group all the input vectors into two sets based on the polarity of the hidden neuron activation value. In each set, if a particular feature x_i has its values clustered to form an interval, then that feature can be said to have support that can be partitioned to make the decision. Such an interval is then considered to *define* a class with respect to x_i .

More generally, we make the following observations. A class can be defined by a set of intervals. If Class A is defined by a set $\{[l_i, u_i]_j : 1 \leq j \leq a\}$ with respect to x_i then the conditions of an input vector to be in Class A can be written as $[l_i, u_i]_1 + [l_i, u_i]_2 + \dots + [l_i, u_i]_a$. The notation $+$ denotes the logical OR. Let $\{[l_i, u_i]_1, [l_i, u_i]_2, \dots, [l_i, u_i]_{a_j}\}$ be a set, of size a_j , of intervals of element x_i . If Class A can be defined by a set of intervals of elements $\{x_i : 1 \leq i \leq n\}$ then the conditions of an input vector to be in Class A can be written as $([l_1, u_1]_1 + [l_1, u_1]_2 + \dots + [l_1, u_1]_{a_1}) * ([l_2, u_2]_1 + [l_2, u_2]_2 + \dots + [l_2, u_2]_{a_2}) * \dots * ([l_n, u_n]_1 + [l_n, u_n]_2 + \dots + [l_n, u_n]_{a_n})$ where $*$ denotes logical AND and $+$ denotes logical OR.

Based on these observations, we can proceed to extract the so-called crisp rules from a trained neural network. After training is complete, we compute the

activation value of each hidden neuron for each input vector. By associating the polarity of the activation values with respect to each element x_i , we can find the set $\{[l_i, u_i]\}$. Rules from the set $\{[l_i, u_i]\}$, for $1 \leq i \leq n$ are then found and are called crisp rules.

This procedure for extracting crisp rules work well if all activation values with respect to a particular input feature for every input vector in a specific class is either positive or negative. In reality, this may not always be possible, when there are input vectors in a class that generate activation values of both polarities for some input features. We consider an interval to be *vague* if it covers more than one class of input vectors; otherwise, the input vector interval is a non-vague interval. Inclusion of these vague intervals in a rule is often justified to improve the comprehensibility of the rules. They are often found between the intervals of the input vectors to represent a class.

A second step in our rule hierarchy construction is to determine a certainty factor (CF) for each rule [9]. A *Certainty Factor* with respect to a Class C, denoted by CF_C , is the percentage of the input vectors correctly classified under a particular crisp rule. The certainty factor is used to measure the precision of the crisp rules and how well the rule performed. A *CF Rule* is a rule of class C that is extracted with a CF_C assigned to it. A *definite* rule is one that has the CF_C of class C equals to one. An input vector that satisfies the condition of a definite rule is definitely in the class specified by the rule.

To extract definite rules, we compute the CF_C for each rule from the set of crisp rules of class C. For each class C, we then determine all vague intervals. If some vague intervals are obtained, modify all crisp rules containing the corresponding input feature to exclude the vague intervals so that the CF_C can be set to one.

2.2 Quasi-morphism Construction

Given a set of rules, we organize them based on their generality in their conditions. Our notation of a default hierarchy is

$$\{ X > Y > \dots > Z \}$$

where X, Y, ..., Z are sets of rules. Sets of rules X and Y are denoted $X > Y$ when the rules in X is more general than those in Y. Rules are in the same set if they are at the same level of generality. A decrease in generality should of course result in an increase in accuracy as measured by the certainty factor.

We consider an example with a two-dimensional feature space to illustrate the above terms. Suppose the two classes A and B occupy, respectively, rectangles [1,

$3] \times [20, 40]$ and $[2, 4] \times [10, 30]$. Examples of crisp rules for the input vectors are

rule(c1) if $(1 \leq x_1 \leq 3 \text{ and } 20 \leq x_2 \leq 40)$ then class A;

rule(c2) if $(2 \leq x_1 \leq 4 \text{ and } 10 \leq x_2 \leq 30)$ then class B.

The vague intervals are $[2, 3]$ for x_1 and $[20, 30]$ for x_2 . The crisp rules $c1$ and $c2$ are transformed to CF rules by assessing their respective certainty factors. The existence of the vague intervals means that they will have certainty factors less than 1. Suppose each class has 30 samples and that classes A and B have, respectively, 7 and 3 samples in the vague region. Using $c1$, all Class B samples are correctly classified but the 3 Class B samples are incorrectly classified; the CF is hence calculated as 0.91. The CF rules are as follows:

rule(cf1) if $(1 \leq x_1 \leq 3 \text{ and } 20 \leq x_2 \leq 40)$ then class A
($CF_A = 0.91$);

rule(cf2) if $(2 \leq x_1 \leq 4 \text{ and } 10 \leq x_2 \leq 30)$ then class B
($CF_B = 0.81$).

From the vague intervals, we can arrive at the following definite rules:

rule(d1) if $(1 \leq x_1 < 2 \text{ and } 20 \leq x_2 \leq 40)$ or $(2 \leq x_1 \leq 3 \text{ and } 30 < x_2 \leq 40)$ then class A ($CF_A=1$);

rule(d2) if $(3 < x_1 \leq 4 \text{ and } 10 \leq x_2 \leq 30)$ or $(2 \leq x_1 \leq 3 \text{ and } 10 \leq x_2 < 20)$ then class B ($CF_B=1$).

From this set of rules ($cf1$, $cf2$, $d1$, and $d2$), a default hierarchy can be constructed as:

$$\{ \{ cf1, cf2 \} > \{ d1, d2 \} \}.$$

Thus far, as illustrated in this example, the definite rules are refinements of the crisp rules, so that they do not contradict the CF rules; i.e., an input vector that satisfies the condition of a CF rule and that of a crisp rule will have the same classification. Using this default hierarchy, the first level rules $cf1$ and $cf2$ can be used to handle input vectors with their respective certainty factors. If further processing is available or desirable, a second level of rules $d1$ and $d2$ can be used to make a definite classification. There are obviously some input vectors that can only be handled by $cf1$ or $cf2$.

In our example, the decision regions for the two classes overlap at the region $[2, 3] \times [20, 30]$, i.e., the region defined by the vague intervals. Unless we overtrain a neural network, the classification accuracy using the training data set will not attain 100%. A neural network with sufficient number of hidden layers or units or both can almost always be trained to any data sets. In this case, the higher level rules will be used to focus on the regions defined by the vague intervals, i.e., those inputs that can be handled by the first level rules but which do not match the conditions of the second level rules.

Since the rules are derived from the trained neural network, the performance of the rules as measured by the training data can at best match that of the trained neural network. If the neural network can attain 100% training data classification, a set of rules, however obtuse, can be found to match the performance. In the interest of having comprehensible rules, our current approach is to isolate the regions defined by the vague intervals and use a separate neural network to train the input vectors in those regions, followed by our rule extraction process again. This set of rules will be incorporated into the original default hierarchy at higher levels. A quasi-morphism can then be of the form:

$$\{ \{ cf01, cf02 \} > \{ d01, d02, \{ \{ cf11 \} > \{ d11, d12 \} \} \} \}$$

An alternate, practical way to assign levels in the default hierarchy to the rules is to use the number of conditions; in this way, the rule sets might be:

$$\{ \{ cf01, cf02 \} > \{ d01, d02, cf11 \} > \{ d11, d12 \} \}.$$

The ambiguous areas are akin to the exception cases in the quasi-morphism or default rule hierarchy. To improve the accuracy of the classification rules, or, equivalently, to allow the rules to model the environment with further precision, those areas can be isolated and the rule extraction process described above can be applied to those ambiguous areas. The result is an additional set of crisp rules that apply to the ambiguous areas. These rules form a quasi-morphism.

3 Experiments and results

The Iris database, widely reported in machine learning studies, was used to further illustrate our approach. The Iris data set consisted of three classes of flowers with 50 patterns each. One class (Setosa) was linearly separable while the other two (Versicolor and Virginica) were not. The data consisted of real values and had a feature dimensionality of four. The four features were the sepal length (x_1), sepal width (x_2), petal length (x_3), and petal width (x_4) [7].

The network structure we used had four input units, four hidden nodes, and one output node. Network training was by the Stuttgart Neural Network Simulator (SNNS) software. The data set was divided into two groups of training set and test set. The training set contained 120 randomly selected input vectors (40 input vectors from each class) and the test set contained 30 input vectors (10 from each class). There were three network structures to be trained, viz. the Setosa network structure, the Versicolor network structure, and the Virginica network structure. In the training process, the Setosa network structure set the data in the class of

Setosa to have the output value equal one while the data not in the class of Setosa had the output value set to zero. This training process was also applied to the other two network structures.

The Setosa and Versicolor network structures had 100% classification accuracy while the Virginica network had 99% accuracy for the training data sets. After the networks were trained, rules were extracted (Fig. 1 and 2). These rules were then used to classify the complete data sets. The rules from the Setosa network structure achieved 100% accuracy. The Versicolor rules achieved 94% accuracy on all of the data set with eight data out of 150 classified into the wrong group. The Virginica rules achieved 88% accuracy on all of the data set with eighteen data out of 150 classified into the wrong group. Overall, the average classification accuracy of the rules was 94% using the complete data set.

R1: If $(1.0 \leq x_3 \leq 1.9)$ or $(0.1 \leq x_4 \leq 0.6)$ Then Setosa
R2: If $(3.0 \leq x_3 \leq 5.1)$ and $(1.0 \leq x_4 \leq 1.8)$ Then Versicolor
R3: If $(4.5 \leq x_3 \leq 6.9)$ and $(1.4 \leq x_4 \leq 2.5)$ Then Virginica

Fig. 1. Crisp rules for the Iris data set.

R1: If $(1.0 \leq x_3 \leq 1.9)$ or $(0.1 \leq x_4 \leq 0.6)$
Then Setosa (CF_{st}=1)
R2a: If $(3.0 \leq x_3 \leq 5.1)$ and $(1.0 \leq x_4 \leq 1.8)$
Then Versicolor (CF_{vs}=0.94)
R2b: If $(3.0 \leq x_3 < 4.5)$ and $(1.0 \leq x_4 < 1.4)$
Then Versicolor (CF_{vs}=1)
R3a: If $(4.5 \leq x_3 \leq 6.9)$ and $(1.4 \leq x_4 \leq 2.5)$
Then Virginica (CF_{vg}=0.88)
R3b: If $(5.1 < x_3 \leq 6.9)$ and $(1.8 < x_4 \leq 2.5)$
Then Virginica (CF_{vg}=1)

Fig. 2. CF rules for the Iris data set.

The crisp rules and the CF rules are shown in figures 1 and 2, respectively. The rules contain only the petal length (x_3) and the petal width (x_4) in their premises. Therefore, we can conclude that the sepal length (x_1) and sepal width (x_2) can be eliminated. Furthermore, the Setosa is linearly separable and we can use either the petal length or the petal width to represent the class of Setosa. These rules can be organized into a quasi-morphism as follows:

$$\{\{R1, R2a, R3a\} > \{R2b, R3b\}\}.$$

We verified that this rule hierarchy has the same classification accuracy as the original trained neural network.

4 Conclusions and Future Work

We presented a model of creating a quasi-morphism of comprehensible rules from a trained neural network. The extracted rules are more comprehensible and are capable of having the same classification performance as the original network. While the comprehensibility of rules are relevant in fields such as machine learning, it can be subsidiary in cognitive modeling. Our rules are extracted through the analysis of hidden units; the analysis is typically localized to specific units. A more global approach has been shown to take advantage of the distributed nature of neural networks [13].

Our ongoing work includes further testing of the model using tasks that are larger in scale and complexity. Through this, we hope to address concerns such as whether the quasi-morphism can represent arbitrarily complicated rules, and whether regularities that are not localized to specific hidden units can be extracted.

Acknowledgments

This work is supported in part by Thailand's National Science and Technology Development Agency as well as a scholarship granted by Thailand's Ministry of University Affairs, and in part by the U.S. Louisiana Board of Regents' Information Technology Research program. The authors further thank the reviewers for their constructive suggestions.

References

- [1] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, Calif., 1993.
- [2] J.H. Holland, K.J. Holyoak, R.E. Nisbett, P.R. Thagard, *Induction*, MIT Press, Cambridge, Mass., 1989.
- [3] W. Wettayaprasit and C. Lursinsap, "Rule extraction based on hidden neural activation interval projection on each dimensional axis," in *Proc. Artificial Neural Network in Engineering Conf.*, St. Louis, Mo., Nov. 2001, pp. 95-100.
- [4] Y. Hanyashi, R. Setiono, and K. Yoshida, "Diagnosis of hepato-biliary disorders using rules extracted from artificial neural

- networks,” in *Proc. IEEE International Fuzzy Systems Conference*, Seoul, Korea, Aug. 1999, pp. 344–348.
- [5] L. Fu, “Knowledge discovery by inductive neural networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 6, pp. 992–998, Nov./Dec., 1999.
- [6] A. Gupta, D. Paer, and S.M. Lam, “Generalized analytic rule extraction for feedforward neural networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 6, pp. 985–991, Nov./Dec. 1999.
- [7] I. A. Taha, and J. Ghosh, “Symbolic interpretation of artificial neural networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 3, pp. 448-463, May/June 1999.
- [8] R. Setiono, “Extraction M-of-N rules from trained neural networks,” *IEEE Transactions on Neural Networks*, vol. 11, no. 2, pp. 512-519, March 2000.
- [9] J. W. Shavlik, “A Framework for Combining Symbolic and Neural Learning,” Computer Sciences Department, University of Wisconsin-Madison, Tech. Rep. 1123, November 1992.
- [10] M. Ishikawa, “Rule extraction by successive Regularization,” *Neural Networks*, vol. 13, pp. 1171-1183, 2000.
- [11] R. Setiono and H. Liu, “Symbolic representation of neural networks,” *IEEE Computers*, vol. 29, pp. 71-77, 1996.
- [12] M. R. W. Dawson, D. A. Medler, D. B. McCaughan, L. Willson, and M. Carbonaro, “Using extra output learning to insert a symbolic theory into a connectionist network,” *Minds And Machines*, vol. 10, pp. 171-201, 2000.
- [13] D. A. Medler, D. B. McCaughan, M. R. W. Dawson, and L. Willson, “When local isn’t enough: Extracting distributed rules from networks,” in *Proc. 1999 Int. Joint Conf. Neural Networks*, pp. 305i-305vi.
- [14] A. Vahed and C. W. Omlin, “Rule extraction from recurrent neural networks using a symbolic machine learning algorithm,” in *Proc. ICONIP’99*, vol. 2, pp. 712-717.
- [15] G. Bologna, “Rule extraction from multi layer perceptron with staircase activation functions,” in *Proc. 2000 Int. Joint Conf. Neural Networks*, vol. 3, pp. 419-414.