Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1997 Proceedings

Americas Conference on Information Systems (AMCIS)

8-15-1997

The Automatic Clustering of A Rule Base for Its Maintenance

Ook Lee
The Claremont Graduate School, leeo@cgs.edu

Follow this and additional works at: http://aisel.aisnet.org/amcis1997

Recommended Citation
Lee, Ook, "The Automatic Clustering of A Rule Base for Its Maintenance" (1997). *AMCIS 1997 Proceedings*. 265.
http://aisel.aisnet.org/amcis1997/265

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 1997 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

The Automatic Clustering of A Rule Base for Its Maintenance

Ook Lee
Program in Information Science
The Claremont Graduate School
Claremont, CA 91711
(e-mail) leeo@cgs.edu

Abstract

This paper introduces a way of clustering a rule base for the purpose of aiding any future maintenance. The automatic clustering is done implementing the Hopfield neural net. Clustering rules should facilitate the understanding of a rule base, which makes the maintenance job easier.

1. Introduction

The content of a rule base makes its maintenance seem clueless compared to conventional software maintenance. In structured programming languages, the maintainer is able to predict the organization of the content. However, the rules of a rule base have no salient features or landmarks, i.e., every code is composed of the same structure which is "IF....THEN...". Rule base maintenance is difficult partly due to lack of software tools, such as one aiding knowledge base understanding, that can facilitate the maintenance process. The goal of this research is to develop a tool for rule base maintenance. The tool structures the unstructured rule base by clustering rules using a neural network algorithm called a Hopfield net.

2. Constructing the Rule Base Clusterizer

The Rule Base Clusterizer(RBC) should be able to accept input regardless of domain so that the maintainer of existing systems can use it as a tool for facilitating understanding of a rule base. For example, the following rule base could be organized in the particular order without violating the syntax of the programming language.

<Rule#1>: **IF** Functionality **=** Simple **AND** Cost = Low **THEN** Word-Processor = Product-A;	<Rule#2> **IF** Functionality = Complex **AND** Cost = Low **THEN** Word-Processor = Product-B;	<Rule#3> **IF** Functionality = Simple **AND** Cost = High **THEN** Word-Processor = Product-C;	<Rule#4> **IF** Personality = good **AND** Experience = Much **THEN** Candidate = Dole;	<Rule#5> **IF** Personality = bad **AND** Experience = Less **THEN** Candidate = Buchanan;	<Rule#6> **IF** Personality = good **AND** Experience = Less **THEN** Candidate = Forbes;

<Rule#1>, <Rule#4>, <Rule#3>, <Rule#5>, <Rule#2>, <Rule#6>

With this kind of mixed-up organization that may result from undisciplined previous development and maintenance work, the maintainer will spend extra time in figuring out the rules. Our goal is to implement an automatic rule clusterizer that can turn this kind of mixed-up rule base or a somewhat structured one, into a distinctively structured rule base where all the rules are categorized into clusters. For example, the Rule Base Clusterizer will categorize above rule base into as following.

[Rule#1, Rule#2, Rule#3], [Rule#4, Rule#5, Rule#6]

With this kind of clusterization, the maintainer will be able to reduce time spent in comprehending the rule base. This approach is particularly helpful for the maintenance of a very large rule base.

Rules can be characterized in terms of their static distance from one another. Here static refers to the idea that the distance is fixed in terms of the syntactic structure of the rule base. We define the static distance between two rules, i, j to be used in automatic clusterization as follows.

Static Distance(Rule#i, Rule#j) = 1-{ (Number of the same terms in Rule#i and Rule#j) / min(number of terms in Rule#i, number of terms in Rule#j) }

We use the Hopfield net to classify rules automatically, i.e., for clustering the rule base. Based on the static distance matrix, a Hopfield matrix is created and a Hopfield net is implemented from the Hopfield matrix. The Hopfield net converges after a series of iterations. The converged Hopfield net produces the clusters for "related" rules.

3. Hopfield Neural Net

Among the many variations of neural nets, we choose a simple algorithm, the Hopfield net, which is useful in pattern recognition of neurons which are fully connected to other neurons. The Hopfield net algorithm is well described in many text books on neural networks and our implementation is based on the algorithm presented by Fausett(1994). We implemented the algorithm in C++ used on IBM PC-compatible machines, Pascal on Vax VMS miniframes, and GNU C on Sun/UNIX Workstations.

The discrete Hopfield net which uses binary inputs as input neurons refers to a fully interconnected neural net in the sense that each unit is connected to every other unit. Since the discrete Hopfield net requires binary inputs, we need to transform the static distance matrix to a binary one which we call the Hopfield matrix. We create a Hopfield net by setting up a criterion value which determines which binary value should be used. For example, when we set the criterion value as 0.7, every non-diagonal static distance value which is less than 0.7 is replaced with a binary value 1 while the static distance value which is equal or larger than 0.7 is replaced with a binary value -1. The diagonal value of the distance matrix remains as 0 in the Hopfield net. Following is the implementation algorithm for the discrete Hopfield net which is based on Fausett(1994).

<The Hopfield Net Algorithm>

Step 0. Create the Hopfield matrix based on a static distance matrix of rules. Before creating the Hopfield matrix, choose a criterion value which can be any number from 0 to 1 so that the criterion value can be used to transform a static distance matrix to the Hopfield matrix.

Step 1. For each input vector **x,** do Steps 2-6 where vector **x** is the indicator of the given rule. This vector will have 1 only in the given rule number's place and 0s in every other bit. For example, if the given rule is rule #2, vector **x** will be [0 1 0 0 0...00].

Step 2. Set initial activations(output signals) of net(**y** vector)equal to the external input vector **x**:

$y_i=x_i$, (i=1,...n)

Step 3. Do steps 4-6 for each unit y_i

Step 4. Compute net input:

A two-dimensional weight vector **w** represents the Hopfield matrix.

$y\text{-}in_i = x_i + \sum_j y_j w_{ji}$.

j

Step 5. Determine activation(output signal):

$y_i = 1$ if $y\text{-}in_i$ 0

$y_i = 0$ if $y\text{-}in_i < 0$

When $y\text{-}in_i = 0$, y_i does not change.

Step 6. Update the activation vector **y** with the new value of y_i.

Step 7. Output the **y** vector as the cluster to which the given **x** vector(rule number) belongs.

Following is the result of running the Hopfield Net on the previous example rule base.

rule#1

iteration#0 100000(initial value)

iteration#1 100000

iteration#2 110000

iteration#3 111000

iteration#4 111000

iteration#5 111000

iteration#6 111000--->This vector indicates that rule#1 belongs to a cluster of (rule#1, rule#2, rule#3).

rule#2

iteration#0 010000(initial value)

iteration#1 110000

iteration#2 110000

iteration#3 111000

iteration#4 111000

iteration#5 111000

iteration#6 111000--->This vector indicates that rule#2 belongs to a cluster of (rule#1, rule#2, rule#3).

rule#3

iteration#0 001000(initial value)

iteration#1 101000

iteration#2 111000

iteration#3 111000

iteration#4 111000

iteration#5 111000

iteration#6 111000--->This vector indicates that rule#3 belongs to a cluster of (rule#1, rule#2, rule#3).

rule#4

iteration#0 000100(initial value)

iteration#1 000100

iteration#2 000100

iteration#3 000100

iteration#4 000100

iteration#5 000110

iteration#6 000111--->This vector indicates that rule#4 belongs to a cluster of (rule#4, rule#5, rule#6).

rule#5

iteration#0 000010(initial value)

iteration#1 000010

iteration#2 000010

iteration#3 000010

iteration#4 000110

iteration#5 000110

iteration#6 000111--->This vector indicates that rule#5 belongs to a cluster of (rule#4, rule#5, rule#6).

rule#6

iteration#0 000001(initial value)

iteration#1 000001

iteration#2 000001

iteration#3 000001

iteration#4 000101

iteration#5 000111

iteration#6 000111--->This vector indicates that rule#6 belongs to a cluster of (rule#4, rule#5, rule#6).

4. Conclusion

The automatic clusterizer of a rule base was experimented on 3 real-life rule bases with substantial amount of rules and produced adequate clustering that helped understanding and facilitated maintenance.

5. References

1. Fausett, L. Fundamentals of Neural Networks: Architectures, Algorithms, and Applications, Prentice-Hall, Inc., 1994.