CAN A FUZZY RULE EXTRACTION FIND AN EXTREMELY TINY NON-SELF REGION?

Akira Imada
Brest State Technical University
Moskowskaja 267, 224017 Brest, Republic of Belarus
Email: akira@bstu.by

Keywords: Network-intrusion-detection, fuzzy-neural-network, a-needle-in-a-haystack.

Abstract: This paper reports one snapshot of our on-going experiments in which a common target we call *a-tiny-island-in-a-huge-lake* is explored with different methods ranging from a data-mining technique to an artificial immune system. Our implicit interest is a network intrusion detection where we usually do not know what does an illegal transaction pattern look like until it completed intrusion when it was too late. Hence our first interest is (i) if it is possible to train the intrusion detection system *only using legal patterns.* From this context we assume data floating in the *lake* are normal while ones found on the island is abnormal. Our second concern is then (ii) to study the limit of the size of the detectable area, that is, when we decrease the size of the island shrinking to zero until which size can the detector detect it. In this paper a fuzzy rule extraction implemented by a neural network architecture is employed. for the purpose,

1 INTRODUCTION

A sultan has granted a commoner a chance to marry one of his 100 daughters by presenting the daughters one at a time letting him know her dowry that had been defined previously. The commoner must immediately decide whether to accept or reject her and he is not allowed to return to an already rejected daughter. The sultan will allow the marriage only if the commoner picks the daughter with the highest dowry. — "Sultan's Dowry Problem" [1]

In real world, we have many problems in which it is easy to access to any one of the many candidate solutions which could be the true solution but most likely not, which we don't know in advance.

The ultimate extreme is called *a-needle-in-a-haystack* problem. The needle originally proposed by Hinton & Nowlan (Hinton and Nowlan, 1987) was exactly the one configuration of 20 binary bits. In other words, the search space is made up of 2^{20} points and only one point is the target. No information such as how close is a currently searching point to the needle.

Yet another problem, *a-tiny-flat-island-in-a-huge-lake* — this is a problem we came across when we had explored a fitness landscape defined on all the possible synaptic weight values of a fully-connected spiking neurons to give them a function of associative memory (Imada, 2004). To simplify it we formalized the problem in more general form as follows.

Test-function (A tiny flat island in a huge lake)[2]
Find an algorithm to locate a point in the region A all of whose coordinates are in $[-a, a]$ $(a < 1)$ in an universe of the n-dimensional hypercube all of whose coordinate x_i lie in $[-1, 1]$ $(i = 1, \cdots, n)$.

Our implicit interest is a network intrusion detection where we usually do not know what does an illegal transaction pattern look like until it completes the intrusion when actually it is too late. Hence, our interest

[1]According to the author(s) of the web-page of Cunningham & Cunningham, Inc. (http://c2.com) the problem was probably first stated in Martin Gardner's Mathematical Recreations column in the February 1960 issue of The Scientific American. To explore the problem more in detail, see, e.g., http://mathworld.wolfram.com. We thank Mariusz Rybnik at University Paris XII for suggesting that the problem is reminiscent of our context.

[2]It is not necessarily to be said for the top of the island to be "flat", but the originally this was a test-bed for evolutionary computations, and the fitness of the island region is one and zero in a lake region, that is why.

is to train the intrusion detection system only using legal patterns. From this context, we assume data floating in the *lake* are normal while those found on the *island* are abnormal.

In this paper, we approach the problem from this view point. That is, we take it, more in general, just a pattern classification problem, but under the constraint that we have two classes one of which includes an extremely few patterns while the other includes an almost infinite number of patterns. Or, we might as well take it a task of discrimination of a few of non-self cells as anomaly patterns from enormous amount of self cells which represent normal patterns.

We have so far exploited the following lately reported approaches: (i) *artificial immune system* approach, especially a *negative selection* algorithm in which constant or variable sized hyper-sphere detectors detect non-self cells (see, e.g. (Ji and Dasgupata, 2004)); (ii) *immuno-fuzzy* approach where a set of fuzzy rules is designed to cover non-self region (see, e.g. (J. Gomez and Dasgupta, 2003)); (iii) *evolutionary computation* approach where a set of detectors randomly created at the beginning eventually evolves to detect non-self; and so on.

In this paper, we study *a fuzzy rule extraction using a neural network* proposed by Castellano et al. (Castellano and Fanelli, 2000). The system they proposed were neatly described in the paper, and it seems to be very sound and efficient, except for the way in that their normal/abnormal data are used to train and test the system. They employed an *Iris-flower-database* in a popular public domain. The database contains three different classes of iris family and one class is assumed to be self whilst the other two are assumed to be non-self. The training samples are chosen at random from these two classes and train the system. Then system is tested using the rest of the data in the database. The result was successful. We, however, doubt the real applicability of the idea of using artificial data set in such a way, at least in a context of intrusion detection. This is principally because of the following two reasons: (i) we don't know what does a non-self datum look like until it completes its intrusion successfully; and (ii) the number of non-self (anomaly) data available is extremely fewer than the number of self (normal) data. Hence our current interest is also two-fold. Firstly, (i) training should be made only by self data; and secondly, (ii) the non-self region should be tiny. We explore these two points using two different fuzzy models.

2 METHODS

Two fuzzy models are as follows.

2.1 Preliminary Experiment — Immuno-Fuzzy Model

A set of fuzzy rules is used to cover the non-self patterns. As already mentioned, self/non-self sells are represented by n-dimensional real valued vectors each of whose coordinate lies in $[-1, 1]$. That is, the self/non-self space is $[-1, 1]^n$, and a self/non-self pattern is represented by a vector $(x_1, \cdots, x_n)$ where $x_i \in [-1, 1]$. Then a fuzzy rule to detect non-self patterns is

If x_1 *is* T_1, $\cdots$, *and* x_n *is* T_n *then* $\mathbf{x}$ *is non-self*

where T_i is a fuzzy linguistic terms which is either of

{*Low, Low-Middle, Middle, Middle-High, or High*}.

Each of T_i maps the x_i to a real value between 0 and 1, expressing the degree to how it is likely to the linguistic value. This is calculated by a membership function, which is defined here using fixed shaped triangular and trapezoidal fuzzy membership functions.

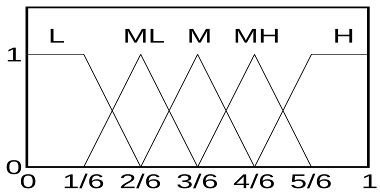

Figure 1: Five fixed shaped membership functions each of which describes how likely a coordinate is either Low (L), Middle-Low (ML), Middle (M), Middle-High (MH), or High (H). Note that the coordinates in $[-1, 1]$ are translated into $[0, 1]$ with interpretation being intact.

Then a genetic algorithm evolves these fuzzy rules, with chromosomes being $(T_1, \cdots, T_n)$, starting with those chromosomes randomly created. To be more specific, our chromosome is made up of n integer genes whose value is chosen from $\{0, 1, 2, 3, 4\}$. The fitness of a rule is evaluated by applying the rule to all the self patterns $\mathbf{x} = (x_1, \cdots, x_n)$, one by one, and calculated as

$$\text{fitness}(R) = 1 - \max_{\mathbf{x} \in Self}\{\min_{i=1,\cdots,n}\{\mu_{T_i}(x_i)\}\}$$

which implies how the rule covers the non-self space.

2.2 A Fuzzy Neural Network Approach

The goal is to classify the data taken from the n-dimensional data-set into either of the pre-defined m classes. For the purpose, Castellano et al. (Castellano and Fanelli, 2000) used the inference mechanism of the zero-order Takagi-Sugeno fuzzy model; then realized the idea by a fuzzy neural network model. To train the fuzzy neuronal network, they employed a combination of (i) *a competitive learning* to determine the architecture of the fuzzy neural network at first and (ii) *a gradient descent learning* to optimize the synaptic weights afterwards. We, on the other hand, employ an evolutionary computation technique to train the network, since we already know the optimal network structure under our current interest, and as such, our concern is just to obtain the solution of weight configuration of the fuzzy neural network.

In the following three sub-subsections, Takagi-Sugeno fuzzy model, a realization of the model by fuzzy neural network, and how we optimize the weight of the fuzzy neural network by an evolutionary computation are described more in detail.

2.2.1 Takagi-Sugeno Model.

Though Castellano et al. (Castellano and Fanelli, 2000) stated the method very clearly in their paper, let us briefly describe it with an intention of making this paper self-contained. Takagi-Sugeno fuzzy inference model is made up of a set of H rules, such as

R_k: IF (x_1 is A_1^k) and $\cdots$ and (x_n is A_n^k)
THEN (y_1 is ν_{k1}) and $\cdots$ and (y_m is ν_{km})

where R_k is the k-th rule ($k = 1, \cdots H$), x_i denotes the i-th variable of the input data ($i = 1, \cdots, n$), y_j denotes the j-th output variable ($j = 1, \cdots, m$), A_i^k denotes a fuzzy set which is usually expressed by a linguistic term such as *"Medium-Large"* but here expressed by a shape of membership function defined one by one on the corresponding input variable, and ν_{kj} denotes a fuzzy singleton each defined on the output variables indicating the likeliness of how the input belongs to the j-th class according to the k-th rule.

A_i^k is defined by Gaussian membership functions

$$\mu_{ik}(x_i) = \exp\{-(x_i - w_{ik})^2/\sigma_{ik}^2\}.$$

Then defuzzification for an input $\mathbf{x}^0 = (x_1^0, \cdots, x_n^0)$ is via the equation:

$$y_j^0 = \{\sum_{k=1}^{H}(\mu_k(\mathbf{x}^0) \cdot \nu_{kj})\} / \sum_{k=1}^{H} \mu_k(\mathbf{x}^0)$$

where

$$\mu_k(\mathbf{x}^0) = \prod_{i=1}^{n} \mu_{ik}(x_i^0)$$

is the results of application of the Larsen product operator.

In other words, the procedure of inference is as follows. When an input $\mathbf{x} = (x_1, \cdots x_n)$ is given, each of the H rules evaluates the $\mathbf{x}$ and output the likeliness of the class, from one class to the next, to which $\mathbf{x}$ belongs to. The evaluation by k-th rule of x_i is by the corresponding membership function $\mu_{ik}(x_i)$ which is specified by giving two parameters w_{ik} and σ_{ik} so that it returns a value ranging from 0 to 1. See, e.g., Fig. 1 where the i-th coordinate of the input $\mathbf{x}$ is evaluated by A_i^k, the i-th antecedent of the *IF* part of the Rule$_k$, which is represented by a membership function not by a usual linguistic term like *"Small"*. The returned membership value in this example in the figure is 0.71, suggesting, say, "The likeliness of if the variable is "Medium Large" is 0.71."
Using those n values of $\mu_{ik}(x_i)$, each of the H rules

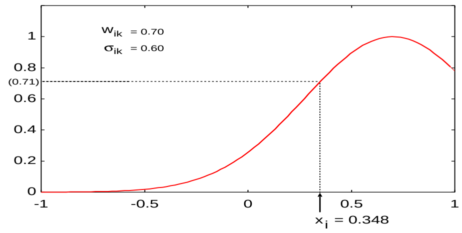

Figure 2: A fictitious sketch of an evaluation of x_i, the i-th entry of the input $\mathbf{x}$, by the i-th antecedent part of the k-th rule A_i^k.

calculates $\mu_k(\mathbf{x})$, and finally these H values are combined to calculate m values of y_j, the resultant defuzzified value for each of the m classes.

2.2.2 Fuzzy Neural Network Implementation.

The procedure described in the previous sub-subsection can be realized when we assume a neural network architecture such as depicted in Fig. 2. The 1st layer is made up of n input neurons. The 2nd layer is made up of H groups of a neuronal structure each contains n neurons where the i-th neuron of the k-th group has a connection to the i-th neuron in the 1st layer with a synaptic connection which has a pair of weights (w_{ik}, σ_{ik}). Then k-th group in the second layer calculates the value $\mu_k(\mathbf{x})$ from the values which are received from each of the n neurons in the first layer. The 3rd layer is made up of m neurons each of which collects the H values from the output of the second layer, that is j-th neuron of the 3rd layer receives the value from k-th output in the second layer with the synapse which has the weight ν_{kj}

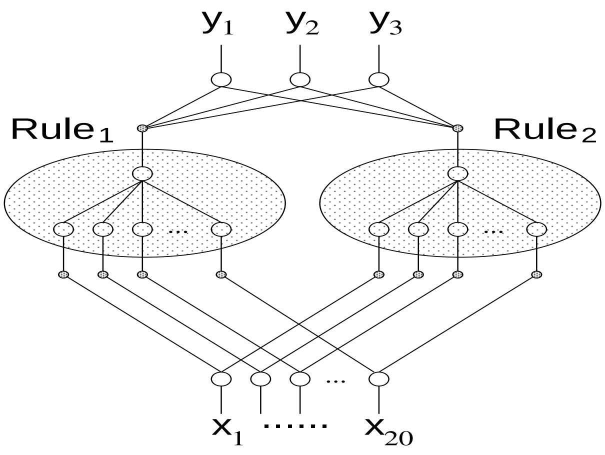

Figure 3: Architecture of the proposed fuzzy neural network which infers how an input $\mathbf{x} = (x_1, \cdots x_n)$ is likely to belong to the j-th class by generating outputs y_j each of which reflect the degree of the likeliness. In this example, a 20-dimension data input will be inferred to which of the 3 classes the input belongs by using 2 rules.

2.2.3 How it learns?

Castellano et al. (Castellano and Fanelli, 2000) used (i) a competitive learning to determine how many rules are needed under initial weights created at random. Then, in order to optimize the initial random weight configuration, they use (ii) a gradient method performing the steepest descent on a surface in the weight space employing the same training data, that is, supervised learning.

Here, on the other hand, we use a simple genetic algorithm, since our target space is specific enough to know the network structure in advance, i.e., only unique rule is necessary. Our concern, therefore, is just obtaining the solution of weight configuration of the network. That is to say, all we want to know is a set of parameters w_{ik}, σ_{ik} and ν_{kj} $(i = 1, \cdots n)$, $(k = 1, \cdots H)$, $(j = 1, \cdots m)$ where n is the dimension of data, H is the number of rules, and m is the number of outputs. Hence our chromosome has those $n \times H \times m$ genes. Starting with a population of chromosomes whose genes are randomly created, they evolve under *simple truncate selection* where higher fitness chromosome are chosen, with *uniform crossover* and occasional *mutation* by replacing some of a few genes with randomly created other parameters, expecting higher fitness chromosomes will be emerged. These settings are determined by trials and errors experimentally.

3 EXPERIMENT

An experiment was carried out in the 20-dimensional space. Our assumption is normal data exist in the *lake* region while abnormal data in the *island* region. We control the size of the island by changing the parameter value a. Furthermore, it is easy to guess that only one inference rule is enough to classify an input into either of the two classes. The architecture of the fuzzy network is, therefore, 20 input nodes, 1 rules, and 2 output nodes.

4 RESULTS AND DISCUSSION

Though our experiments have sometimes reversed our expectations depending on parameter setting, we are obtaining a series of successful results.

4.0.4 Where is the tiny island?

In the preliminary experiment using a five fixed shaped triangular and trapezoidal fuzzy membership functions, evolution converges to the chromosome

$$\{M, M, M, \cdots, M\}$$

which implies

IF x_1 is Middle, and $\cdots$, and x_{20} is Middle THEN no-self.

However, this holds only on the condition that the island is fairly large.

4.0.5 How tiny island can be detected?

In Fig. 3, we showed an example of obtained membership function corresponding to one antecedent of the rule (Left), as well as one of the output singletons obtained in the same experiment (Right). Training Samples are from the assumed legal data exist in the *lake* region to identify the illegal data exists in the *tiny island* region defined as $a = 0.1$. In the figure, although only one example of membership function is shown out of 20 others, the other 19 membership factions are more or less similar to the one shown in the figure. This suggest that

R_1: IF all of x_n is near the origin THEN (y_1 is HIGH) and (y_2 is LAW).

Namely, the input belongs to the abnormal class.

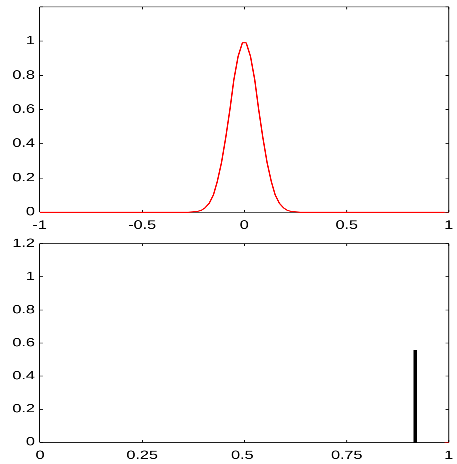

Figure 4: An example of experimental result of a membership function of one antecedent membership function of a rule (Left), and one of the two output singletons of the same experiment (Right). Training samples are from the assumed legal data exist in the *lake*, while the illegal data is assumed to be in the *tiny island* defined as $a = 0.1$.

5 SUMMARY

In this paper, we have reported our on-going investigations, that is, how already proposed methods work on a special situation of what we call *a-tiny-island-in-a-huge-lake*, suggesting to use it a test-function to design a network intrusion system. When we increase the difficulty of the problem by making the size of the island shrink to zero, it will become what they call *a-needle-in-a-haystack*. As far as we know, this issue has resisted to be fully unveiled and still remains open. Though our results so far has not been matured yet, we hope a lot of experiments await our exploration which might result in useful observations in considering how we design a network intrusion detection system.

REFERENCES

Castellano, G. and Fanelli, A. M. (2000). Fuzzy inference and rule extraction using a neural network. In *Neural Network World Journal Vol. 3, pp. 361–371*.

Hinton, G. E. and Nowlan, S. J. (1987). How learning can guide evolution. In *Complex Systems, 1, pp. 495–502*.

Imada, A. (2004). How a peak on a completely-flatland-elsewhere can be searched for? — a fitness landscape of associative memory by spiking neurons. In *Advanced Computer Systems (ACS) and Computer Information Systems and Industrial Management Applications (CISIM), Vol.2, pp. 171–150*.

J. Gomez, F. G. and Dasgupta, D. (2003). An immuno-fuzzy approach to anomaly detection. In *IEEE International Conference on Fuzzy Systems, Vol. 2, pp. 1219-1224*.

Ji, Z. and Dasgupata, D. (2004). Augmented negative selection algorithm with variable-coverage detectors. In *Congress on Evolutionary Computation. pp. 1081–1088*.