Can a fuzzy rule extraction find an extremely tiny non-self region?

Akira Imada

[1] Brest State Technical University
Moskowskaja 267, 224017 Brest, Republic of Belarus
akira@bstu.by
http://neuro.bstu.by/ai/akira.html
[2]
[3]

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

This paper is a report of our on-going investigation into how do already proposed methods work on a special situation of what we call a-tiny-island-in-a-lake. To be more specific, we now are exploring a couple of so far proposed approaches for the purpose of searching for an extremely tiny unknown region of non-self data surrounded by an overwhelming amount of known self data partly with an interest from a viw point of *network intrusion detection* where self data imply normal transactions while non-self data imply anomaly.

Assuming our whole universe is n-dimensional Eucledian space all of whose coordinate are in $[-1, 1]$, two differnt regions of non-self data are our targets to be searched for. One is a tiny hyper-retangle all of the coordinates inside are in $[-a, a]$ $(a < 1)$, and the other is a tiny hyper-sphere whose center locates at the origin and radius is $a < 1$. In a *network intrusion detection*, the number of abrormal patterns which we assume as non-self is extremelly fewer thant the number of normal patterns which we assume as self data This is the reason why our target should be extremly *tiny*, In other words, $a \approx 0$ is our condition.

We have so far exploited (i) *artificial immun system* approach, especially a *negative selection* algorithm in which constant or variable sized hyper-sphere detectors detect non-self cells; (ii) *immuno-fuzzy* approach where a set of random fuzzy rules eventually evolves to cover non-self region; (iii) *evolutionary computation* approach where also an evolution of a set of random detectors finally detect non-self; and so on.

In this paper, we study *a fuzzy rule extraction using a neural network* proposed by G. Castellano et al. (2000) [1]. The system they proposed were very clearly descripted and it seems to be very sound and efficient, except for the way in that their data applied by the system. They employed an *Iris-database* in a popular public domain. The database contains 3 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 idea of using artificial data set in such a way in a context of intrusion detection. This is principally because of the two following reasons: Usually, in the context of intrusion detection, (i) the number of non-self (anomaly) data is extremely fewer than the number of self (normal) data; and (ii) we don't know what does a non-self datum look like until it complete its intrusion sucessfully. It would be too late.

Hence our current interest is also two-fold: First, the non-self region should be tiny and secondly, training should be made only by self data. We explore these two points using above mentioned *fuzzy rule extraction by neural network* proposed by G. Castellano et al. (2000) [1].

2 Method

The goal is to classify each of the data from n-dimensional data-set into either of m classes. For the purpose, Castellano et al. [1] used the inference mechanism of a zero-order Takagi-Sugeno fuzzy model; then realizee the idea by a fuzzy neural network model. To train the fuzzy neural network, they emploied a combination of a competitive learning to determine the architecture of the fuzzy neuralnetwork and a gradient descent learning to optimeze the synaptice weights. We, on the other hand, emploies, an evolutionary computation technique to train the network since we already knew the network structure under our current interest and as such, our conceren is just to obtain the solution of weight configuration of the network, and an evolutionaly computation is expected to find it more simply than the prosed approach.

In the following three subsections (i) Takagi-Sugano fuzzy modle, (ii) a realiziztion of the model by fuzzy neural network, and (iii) how we optimize the weight of the fuzzy neural network by an evolutionary computation.

2.1 Takagi-Sugeno Model

Though Castellano et al. [1] very clearly stated the method in their papaer, 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

$$R_k\text{: IF } (x_1 \text{ is } A_1^k) \text{ and } \cdots \text{ and } (x_n \text{ is } A_n^k)$$

$$\text{THEN } (y_1 \text{ is } \nu_{k1}) \text{ and } \cdots \text{ and } (y_m \text{ is } \nu_{km})$$

where R_k is the k-th rule $(k = 1, \cdots H)$, x_i is the i-th variable of the input data $(i = 1, \cdots, n)$, y_j is j-th output variable, A_i^k are fuzzy sets which are usually expressed by linguistic terms such as "midium large" but here expressed by a shape of membership function defined one by one on the corresponding input variable, and ν_{kj} are fuzzy singletons defined on the output variables indicating the likliness of the j-th class infered by 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 input $\mathbf{x}^0 = (x_1^0, \cdots, x_n^0)$ is according to

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

where

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

is the results of application of 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 likliness 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} returning a value ranging from 0 to 1. See, for example, Figure 1 where i-th coordinate of the input $\mathbf{x}$ is evaluated by A_i^k, i-th antiticident of the IF part of the Rule$_k$, which is represented a set of membership function not a usual linguistic term like "Large". The returned membership value of this example in the Figure is 0.71, suggesting, say, *the likliness of if the variable is "Midium Large" is 0.71 ...*

Each of the H rules calculates $\mu_k(\mathbf{x})$ from those n values of $\mu_{ik}(x_i)$. Finally, those H values are conbined to calculate m values of y_j, the result of defuzzification for each of the m classes.

This procedure is realized when we assume a neural network such as depicted in Figure 1. The first layer is made up of n input neurons. The second layer is made up of H

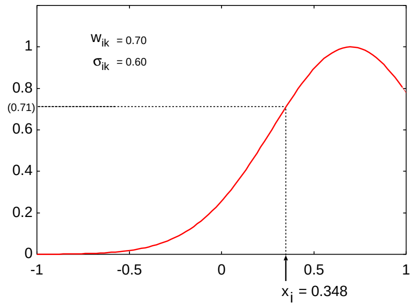

Fig. 1. A fictitious sketch of an evaluation of x_i, i-th entry of the input **x**, by A_i^k, the i-th anticident part of a k-th rule.

groups each contains n neurons where i-th neuron of k-th group has a connection to i-th neuron in the first layer with the synapse which has a pair of weights (w_{ik}, σ_{ik}). Then k-th group in the second layer calculate the value $\mu_k(\mathbf{x})$ from the values its n neurons received. Third layer is made up of m neurons each of which collects the H value from the output of the second layer, that is j-th neuron of the third layer receives the value from k-th output in the second layer with the synapse which has the weight ν_{kj}

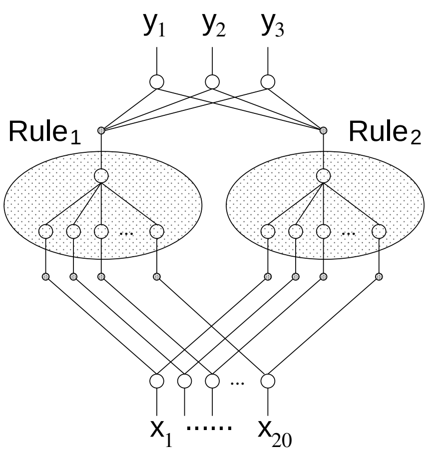

Fig. 2. Architecture of the proposed fuzzy neural network which infers how an input $\mathbf{x} = (x_1, \cdots x_n)$ is likely to belong to the j-th class by generating outputs y_j each of which reflect the degree of the likliness. In this figur 20-dimension data input will infer which of the three classes the input belongs by using two rules.

2.2 How it learns?

Castellano et al. [1] used (1) a competitive learning to determin how many rules are needed and initial weights. Then in order to optimize these initial weights they use (2) 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 already, our conceren is just to obtain 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 higer fitness chromosome are chosen, with *uniform crossover* and ocasional *mutation* by replaceng 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.

To evaluate fitness of each chromosome, we use a measure originally proposed by Lopes et al. [2] and used widlly nowadays in which four quantities, i.e., (i) true-positive, (ii) true-negative, (iii) false positive, and (iv) false negative are used. Here we assume positive sample is non-self and negative sample is self, since detectors is designed to detect non-self cells. Hence, these four terms are defined in a sence that (i) t_p (true positive) — true declaration of positive sample, i.e., non-self declared as non-self (ii) f_p (false positive) — false declaration of positive sample, i.e., self declared as non-self (iii) t_n (true negative) — true declaration of negative sample, i.e., self declared as self (iv) f_n (false negative) — false declaration of negative sample, i.e., non-self declared as self. Under these definitions $d_r = t_p/(t_p + f_n)$ implies detection rate, and $f_a = f_p/(t_n + f_p)$ implies false alarm rate. Then we plot d_r versus f_a, and the resultant graph is called *Receiver Operating Characteristics (OCR)* [3] which reflects a tradeoff between false alarm rate and detection rate.

3 Experiment

Castellano et al. [1] used *Iris* data found in a public domain in a very clever way, writing as follows.

> *The validity of our approach to fuzzy inference and rule extraction has been tested on the well-known benchmark Iris data problem. The classification problem of the Iris data consists of classifying three species of iris flowers (setosa, versicolor and virginica). There are 150 samples for this problem, 50 of each class. A sample is a four-dimensional pattern vector representing four attributes of the iris flower (sepal length, sepal width, petal length, and petal width).*

However, we doubt this way of using *Iris* data as an artificial data at least in the context of network intrusion detection. One reason of our doubt is that illegal patterns are usually unknown and it cannot be represented by certain specific patterns, if not al all. The other reason is the sparseness of this data set. What if the system meets a pattern which does not belong to either of the two classes? See another report of us [4] regarding this issue.

Our target problem is what we call *a-tiny-flat-island-in-a-huge-lake* which 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 [5]. To simplify it, we formalized the problem in more general form as follows.

Testfunction 1 (A tiny island in a huge lake - 1) *Find an algorithm to locate a point in the region A all of whose coordinates are in $[-a, a]$ $(a \approx 0)$ in an universe of the n-dimensional hypercube all of whose coordinate x_i lie in $[-1, 1]$ $(i = 1, \cdots, n)$.*

Or:

Testfunction 2 (A tiny island in a huge lake -2) *Find an algorithm to locate a point in the region A all of whose coordinates are in the hyper-sphere whose radius is a $(a \approx 0)$ and its center locates at the orgin in an universe of the n-dimensional hypercube all of whose coordinate x_i lie in $[-1, 1]$ $(i = 1, \cdots, n)$.*

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 controll the size of the island by changing the value a. Hence it is easy to guess that only one inference rule is enough to classify input into either of two classes. The architecture of the fuzzy network is, therefore, twenty inputs, one rules, and two outputs.

4 Results and Discussion

Though our experiments have sometimes reversed our expectations depending on parameters determined, we are obtaining a series of successful results, such as shown in Figure 3 where an example of obtained membership function corresponding to one anticident of the rule (Left), as well as one of the two output singletons of the same experiment (Right). Training Samples are from the assumed legal data exist in the *lake* to identify the illegal data exists in the *tiny island* difined as $a = 0.1$.
In the Figure, although only one example of membership fuction out of 20 other 19 membership fuctions 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).

That is the input belongs to the abnormal class.

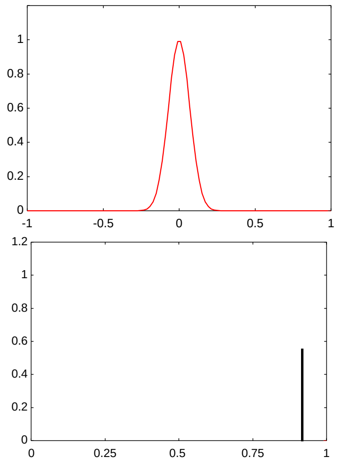

Fig. 3. An example of experimental result of a membership function of one anticident membership function of a rule (Left), and one of the two output singletons of the same experiment (Right). Training Samples are from the assumed legal data exist in the *lake*, while the illegal data is assumed to be 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*. When we increase the difficulty of the problem by making the size of the island shrink to zero, it becomes what they call *a-needle-in-a-haystack*. As far as we know, this issue has resisted to be fully solved and still remains open. Though our results so far has not been matured yet, we hope a lot of experiments which might result in positive obserbasions in considering how we design a network intrusion detection system await our exploration.

References

1. G. Castellano and A. M. Fanelli(2000) ”Fuzzy Inference and Rule Extraction using a Neural Network.” Neural Network World Journal Vol. 3, pp. 361–371.
2. H. S. Lopes, M. S. Coutinho, W. C. Lima (1997) “An Evolutionary Approach to Simulate Cognitive Feedback Learning in Medical Domain.” Genetic Algorithms and Fuzzy Logic Systems. World Scientific, pp. 193–207.
3. F. Provost, T. Fawcett, and R. Kohavi (1989) “The case against accuracy estimation for comparing induction algorithms.” Proceedings of international conference on machine learning, pp. 445–453.
4. A. Imada (2005) “When One Iris Flower is Normal Then are Others Abnormal?” Proceedings of International Conference on Pattern Recognition and Information Processing,
5. A. Imada (2004) “How a Peak on a Completely-flatland-elsewhere can be Searched for? — A Fitness Landscape of Associative Memory by Spiking Neurons.” Proceedings of Advanced

Computer Systems (ACS) and Computer Information Systems and Industrial Management Applications (CISIM), Vol.2, pp. 171–150.