Proceedings of the 2010 IEEE
International Conference on Information and Automation
June 20 - 23, Harbin, China

Bioinformatics Data Mining Using Artificial Immune Systems and Neural Networks

Shane Dixon, Xiao-Hua Yu

Department of Electrical Engineering
California Polytechnic State University
San Luis Obispo, CA 93407, USA

***Abstract* - Bioinformatics is a data-intensive field of research and development. The purpose of bioinformatics data mining is to discover the relationships and patterns in large databases to provide useful information for biomedical analysis and diagnosis.**

In this research, algorithms based on artificial immune systems (AIS) and artificial neural networks (ANN) are employed for bioinformatics data mining. Three different variations of the real-valued negative selection algorithm and a multi-layer feedforward neural network model are discussed, tested and compared via computer simulations. It is shown that the ANN model yields the best overall result while the AIS algorithm is advantageous when only the "normal" (or "self") data is available.

Index Terms - Artificial immune systems, Real-valued negative selection algorithm, Data mining, Artificial neural networks.

I. INTRODUCTION

Bioinformatics is a fast-growing, data-intensive field that involves the applications of information technology to molecular biology. The purpose of bioinformatics data mining is to discover the relationships and patterns in large bioinformatics databases to provide useful information for biomedical analysis and diagnosis. To accomplish this task, many approaches have been proposed, including algorithms based on artificial immune systems and artificial neural networks.

The biological immune system is a complex adaptive system of cells, molecules, and organs that can recognize foreign substances and then neutralize or degrade them, with or without injury to its own tissues. Over years, the immune system has evolved sophisticated pattern recognition and response mechanisms using its network of chemical messengers for communication. Through an evolutionary learning process, the immune system can recognize an almost limitless variety of infectious foreign cells and substances (known as "non-self" elements), and distinguish them from those native noninfectious cells (known as "self" elements).

The negative selection algorithm (NSA) was first introduced by Stephanie Forrest in 1994 [1]. It is a computational model based on the self/non-self discrimination process performed by the T-cells in natural immune systems. Recently, NSA has attracted the attention of many computational intelligence researchers and has been successfully applied to solve many engineering problems in recent years, such as computer network security analysis, fault detection, and data mining.

In a negative selection algorithm, detectors are generated randomly first, then they are evolved (i.e., eliminated if they match any "self" samples) to obtain a set of trained "mature" detectors. In the testing mode, each unknown data instance is presented to the detector set and classified as either "self" or "non-self". That is, if the unknown data instance matches any detector in the detector set, then it is classified as "non-self" or an anomaly; while on the other hand, if the incoming data instance is not recognized by any detector, it is considered to be a member of the "self" set.

Different negative selection algorithms can be characterized, or distinguished by particular data representation schemes, matching rules and detector generation processes. It is known that negative selection algorithms are often employed to classify data; therefore, they are defined first and foremost by different data representation schemes. The early implementations of negative selection algorithms can only classify binary data. Later on, it was extended to handle data in string representation (characters). The focus of this study concerns real-valued data representation, a more recent topic of research.

The detector generation and elimination mechanisms implemented in a negative selection algorithm are also important characteristics of the algorithm. To date, only random-based generation schemes have been implemented for real-valued vector data representation. Other approaches to detector generation may include genetic algorithms and optimization with aftermath adjustment ([2] [3] [4]).

The central mechanism of a negative selection algorithm is the selection of an appropriate matching rule, or distance measure in the case of real-valued data. The matching rule is a measure of affinity or similarity that two data instances share, and is generally application specific and data representational dependent.

In addition to data representations, detector generation processes and matching rules, there are also a number of other factors that affect the performances of negative selection algorithms. For example, the number of detectors affects the efficiency of generation and detection, and consequently the speed of the algorithm. Linked directly to the accuracy of detection, detector coverage is also an important factor to consider during detector generation. The stopping criteria are often used to determine an adequate number of detectors and their coverage.

In this research, algorithms based on artificial immune systems (AIS) and artificial neural networks (ANN) are

978-1-4244-5704-5/10/$26.00 ©2010 IEEE

employed for bioinformatics data mining. Three different variations of the real-valued negative selection algorithm (i.e., the detectors with fixed radius, the V-detector with variable radius, and the proliferating V-detectors) with five different performance metrics (i.e., the Euclidean distance, the Manhattan distance, the 3-norm Minkowski distance, the partial Euclidean distance, as well as the Chebyshev distance) are studied. As a comparison, a multi-layer feedforward neural network model is also developed and tested.

II. The Real-valued Negative Selection Algorithms

The real-valued negative selection algorithm (RNSA) was first proposed in 2002 [5]. In this algorithm, data (including both training and testing data), detectors, affinity (or performance metrics), and the matching threshold are both represented by real-valued data in an *n*-dimensional real vector space.

The most commonly used distance metric in RNSA is the Euclidean distance, but many others exist. In fact, the selection of an appropriate distance measure is crucial to the overall performance of the algorithm, due to the fact that the entire process of a negative selection algorithm is built upon the concept of affinity or distance. In the detector generation process, the number of detectors and the estimation of detector coverage are both related with the distance metric. During the detection phase, the decision rule to classify the unknown incoming data as either "self" or "non-self" is also based on the distance measure.

One unique feature of the distance metric chosen for a real-valued negative selection algorithm is the impact it has on the shape of the detectors. The detectors are assigned a real-valued threshold in self/non-self discrimination, which can be envisioned as a radius of detection. If the distance between the detector and a data point is less than this threshold, then this sample is "detected" by the detector and thus classified as a member of the non-self set.

Consider a point $(x_1, x_2, x_3, \ldots, x_n)$ and another point $(y_1, y_2, y_3, \ldots, y_n)$ in *n*-dimensional real vector space. The Minkowski distance, or the *m*-norm distance between two points, is defined as ([2] [6]):

$$\left(\sum_{i=1}^{n}\left|x_i - y_i\right|^m\right)^{\frac{1}{m}} \tag{1}$$

where *m* is also called the order of the Minkowski distance.

The commonly used Euclidean distance can be considered as a special case of the Minkowski distance of order 2 (or 2-norm):

$$\sqrt{\sum_{i=1}^{n}\left(x_i - y_i\right)^2} \tag{2}$$

The 1-norm distance is called the Manhattan distance metric, and is simply the absolute value of the difference between the two points in n-dimensional space:

$$\sum_{i=1}^{n}\left|x_i - y_i\right| \tag{3}$$

The 3-norm Minkowski distance metric is similar to the Euclidean distance, except the difference is cubed and the summation is cube-rooted:

$$\left(\sum_{i=1}^{n}\left|x_i - y_i\right|^3\right)^{\frac{1}{3}} \tag{4}$$

The Chebyshev distance is also called the infinity-norm Minkowski distance:

$$\lim_{m\to\infty}\left(\sum_{i=1}^{n}\left|x_i - y_i\right|^m\right)^{\frac{1}{m}} = \max\left(\left|x_i - y_i\right|\right) \tag{5}$$

where $i = 1, 2, \ldots, n$. Note that by taking the limit as *m* approaches infinite, it yields the maximum distance between two points; and thus often simply referred as the maximum distance metric.

A "partial Euclidean distance", or the "Euclidean distance with a sliding window" is also employed in this research (simply referred as the "window" distance metric in section 4). Let's consider an example of two arbitrary points in a four-dimensional real space, i.e., (x_1, x_2, x_3, x_4) and (y_1, y_2, y_3, y_4), and assume the sliding window has a fixed size of 2. First, the Euclidean distance is calculated, but only for (x_1, x_2) and (y_1, y_2). Next, the window of observation "shifts", or "slides" to (x_2, x_3) and (y_2, y_3), and then finally conclude with (x_3, x_4) and (y_3, y_4). Of the three separate distances calculated, only the one with the smallest absolute value is retained (and others are discarded). The partial Euclidean distance determines the smallest distance in a lower-dimensional space (in this case, the dimension is 2) for the data in a higher-dimensional space (in this case, the dimension is 4).

2.1. The real-valued negative selection algorithm with fixed detector radius

In the real-valued negative selection algorithm with fixed detector radius proposed by Gonzalez and Dasgupta [5], the detector generation phase begins by randomly generating a preset number of points in *n*-dimensional real space $[0, 1]^n$, with a mean value of ½ (for simplicity, it is assumed that the input data is also normalized within $[0, 1]^n$). In other words, each detector can be envisioned as a hypersphere with a center and fixed radius *r* in an *n*-dimensional space. The detectors are then trained with only self samples; that is, the positions of the detectors are updated through an iterative process. The detectors must remain away from the self points and also remain separated from other detectors in order to maximize the non-self space covering. The new location of the detector is determined by:

$$d(i+1) = d(i) + \eta_i * dir \tag{6}$$

where $d(i)$ is the current position (center) of the detector, $d(i+1)$ is the new position of the detector, η_i is the adaptation rate, *i* is the age of the detector, and *dir* is the direction of moving. Since it is undesirable for the detectors to match self points, the shortest allowable distance for a good detector to the self set is *r* (it is also referred as the threshold for matching).

The adaptation rate η_i can be updated as:

$$\eta_i = \eta_0 e^{-\frac{i}{\tau}} \quad (7)$$

where the preset adaptation rate parameter η_o represents the initial step size used to move the detectors. In order to guarantee that the algorithm will converge to a stable state, it is necessary to decrease this parameter in each iteration in such a way that

$$\lim_{i \to \infty} \eta_i = 0 \quad (8)$$

The direction of moving *dir* can be calculated based on the shortest calculated distance to any self point or detector:

$$dir = \frac{\sum_{i=1}^{n} \left(d_i - c^{nearest} \right)}{\left| \sum_{i=1}^{n} \left(d_i - c^{nearest} \right) \right|} \quad (9)$$

2.2. The V-detector algorithm

In [3], Zhou and Dasgupta proposed a different scheme of detector generation and matching mechanisms for negative selection algorithms. This algorithm (called the V-detector algorithm) includes a new variable parameter, which is the radius of each detector.

The generation phase of the V-detector algorithm begins by randomly generating detector candidates; but instead of generating a full set of detectors, it generates detector candidates one at a time. Each individual candidate is checked using the matching rule determined by the choice of distance metric. If the distance to the nearest self point is less than the threshold value (which is the radius of this nearest self point r_s), the detector is eliminated and a new candidate is generated. If the minimum distance to any self point is greater than the radius of this self point r_s, then the detector is stored temporarily and its radius is recorded as r_d, which is the minimum distance to the nearest self point:

$$r_d = dist\_min \quad (10)$$

This is known as the aggressive approach to assign a detector's radius [6]. Detectors are iteratively generated and assigned a radius based on this mechanism until the stopping criteria is met.

A more conservative approach to detector radius assignment can also be implemented, in which the detector radius r_d is determined by the difference between the minimum distance to the nearest self point and the self radius r_s of the nearest self point [3]:

$$r_d = dist\_min - r_s \quad (11)$$

In this research, both implementations are initially tested and compared. The aggressive strategy produces more accurate results, and consequently was the method chosen for this study.

2.3. The proliferating V-detector algorithm

One of the most recent advances in real-valued negative selection algorithms incorporates the implementation of proliferating variable-sized detectors [7]. During the generation phase, the detector set is filled with an initial set of detectors in the same manner as the generation phase for the V-detector algorithm; the only difference is the assignment of the variable radius r_d. The proliferating V-detector algorithm includes an additional threshold term θ which is also subtracted from the variable radius r_d. Therefore, for the aggressive variable radius approach:

$$r_d = (dist\_min - \theta) \quad (12)$$

while the conservative variable radius approach:

$$r_d = (dist\_min - r_s - \theta) \quad (13)$$

In this study, the aggressive approach is chosen for computer simulations.

After the generation phase concludes, the proliferation stage begins to proliferate (or clone) new detectors from the detector set initially created from the generation stage. These new detectors are referred to as offspring. At the beginning of the proliferation stage, the algorithm already has a set of detectors D from the previous generation stage. In the *i-th* iteration, it selects one of those detectors whose center and radius are x_i and r_i from the set D, and creates new offspring located at a distance r_i from x_i. In two dimensional vector space, the original detector is regarded as a circle of radius r_i in the non-self region centered around x_i, and the offspring detectors will be located along the circle's circumference at a location $x_i + \hat{u}r_i$, where $\hat{u}$ is some unit direction vector [7]. The offspring's radius is set to be equal to the minimum distance from its center to the nearest self point, but can also be modified to include the additional threshold θ, as in the previous discussions.

Offspring coverage is controlled in the same manner as the detector generation phase of the V-detector algorithm. Since a new detector has additional coverage value only when another does not already cover the space, only those offspring detectors which are not covered will be retained for the detection phase. The detectors in D are selected for proliferation in a sequential manner, with the unit vectors û are kept to be either parallel (+1) or anti-parallel (-1) to each dimension.

The proliferation stage may not only involve one stage of proliferation. Several stages of proliferation, where the offspring from one stage is allowed to proliferate in the next stage, are often desirable. Maintaining the threshold θ initially high during the first the first generation stage, and lowering it towards zero in a stepwise manner during subsequent proliferation stages, can result in much better coverage of the non-self subspace. This is because decrementing the threshold θ at the end of each stage creates a gap between the self/non-self boundaries. This gap can then be filled by the offspring detectors of the next proliferation stage. Steadily decreasing the gap by lowering θ will result in increasingly smaller, but strategically placed offspring to proliferate around the

self/non-self boundary region. To ensure full coverage of the non-self subspace, the threshold θ must be set to zero during the last stage of proliferation [3].

III. The Neural Network Model

In this section, a multi-layer feedforward artificial neural network (ANN) model for bioinformatics data classification is discussed. It is well known that ANN can learn the input-output mapping of a system through an iterative training and learning process, and thus is an ideal candidate for pattern recognition and data analysis.

The ANN model has an input layer, an output layer, and one or more hidden layer(s). There are n inputs for the n-dimensional input data; and an output which indicates the class of input vector ("self" or "non-self"). That is, the neural network model is a multi-input, single-output system. The activation function for each hidden neuron is chosen as the sigmoid function:

$$f(x) = \frac{1}{1+e^{-x}} \tag{14}$$

The weights of the neural network are initialized at random, and then updated using the back-propagation algorithm to minimize the following objective function:

$$J(k) = \frac{1}{2}\left[d(k) - y(k)\right]^2 \tag{15}$$

where d is the desired output (class) and y is the output of neural network, k is the index of a training pair.

$$W(k+1) = W(k) + \Delta W \tag{16}$$

where

$$\Delta W = \mu \frac{\partial J}{\partial W} \tag{17}$$

where μ is the learning rate.

On-line learning approach employed in this study. An input sample pattern is fed into the network, resulting in an error signal at the output. The error signal is then back propagated through the network in order to adjust the synaptic weights of each neuron. The above procedure repeats until all input samples within the training set have been exhausted. The order of the training samples is then randomly rearranged and another training pass is conducted, until the maximum number of iterations reached or the error signal is reduced to an acceptable level. To remain consistent with the negative selection algorithm, the output of neural network is also bounded between [0, 1], with a decision threshold of 0.5. That is, the input data is classified either as "1" (self) if y • 0.5, or "0" (non-self) if $y < 0.5$.

There is a major distinction between the negative selection and neural network algorithm which must be addressed at this point. While a negative selection algorithm, by design, requires training of only one class of data, the neural network algorithm must be trained with samples from both classes of data. In bioinformatics data mining, it is very common that one class of data is dominant over the other class. For example, a database may contain large amount of testing results from the "normal" population while the "abnormal" samples are just a small portion of the entire dataset. Under this circumstance, the negative selection algorithms may outperform the neural network model.

IV. Simulation Results

In this section, three different real-valued negative selection algorithms and a multi-layer feedforward neural network model are tested and compared via computer simulations. The data used in this research is from the "biomedical dataset" reported by Larry Cox in 1982 in the CMU StatLib datasets archive [8]. In a study to develop screening methods to identify the carriers of a rare genetic disorder disease, four measurements (m_1 , m_2 , m_3 , and m_4) were taken from human blood samples. The data contains 209 observations, with 134 samples are considered to be "normal" (or free of disorders) and 75 samples are identified as the "carriers" of the disorder.

Two performance metrics are employed to evaluate the effectiveness of each algorithm, i.e., the detection rate and false alarm rate [9]. The detection rate (DR) is defined as the number of correctly identified non-self samples divided by the total number of non-self samples. This yields a percentage of correctly identified non-self points, signifying how well the algorithm detected anomalies. Conversely, the false alarm rate (FA) is calculated as the number of self samples classified incorrectly divided by the total number of self samples. This produces a percentage of self samples classified incorrectly, signifying how poorly the algorithm misclassifies self data as an anomaly. A figure of merit (FOM) is proposed by authors to determine an overall final score for the performance of the algorithm, which is defined as the difference between the false alarm rate and the detection rate (DR-FA). It is a method of comparing how well the algorithm detects anomalies while simultaneously penalizing it for self misclassifications.

The neural network model is tested for two cases. The first is the case in which the network is trained with the same set of data as the negative selection algorithm (i.e., only self data is included), while in the latter case the network is trained with mixed samples including both self and non-self data. The results from training with only self data clearly demonstrate that the neural network fails to classify input data. Since the network is only trained with self data, the desired output for all training data is always the same (i.e., "1"). That implies that the network is basically trained to only output a "1"; and thus any new unknown data is always classified as a "1". In other words, the detection rate is a constant zero, i.e., all non-self data is consistently classified as "self".

In the second case, the neural network model is trained with 97 randomly selected samples (with 64 of them being "normal" and 33 of them being "carrier"), and then tested with the rest of samples in the database. The FOM of neural network for the testing data is 46.99%.

The computer simulation results of different real-valued

negative selection algorithm are summarized in tables 1, 2, and 3, where "FD" represents the algorithm with fixed-radius detectors, "VD" represents the algorithm with variable-radius detectors, and "PVD" is for the algorithm that is with the proliferating detectors; "Euclidean", "Manhattan", "Window", "3-Norm", and "Max" indicate the different distance metrics used for simulation, as discussed in section 2. The last row, "Average" is an arithmetic average value of the performance of each algorithm with different metrics.

TABLE I
The Detection Rate (%) of Each Algorithm

	FD	VD	PVD
Euclidean	32.30	30.06	50.55
Manhattan	39.63	28.96	77.00
Window	74.54	66.71	46.28
3-Norm	32.88	30.68	47.17
Max	34.51	28.86	37.23
Average	42.77	37.05	51.65

TABLE II
The False Alarm Rate (%) of Each Algorithm

	FD	VD	PVD
Euclidean	5.74	5.00	11.22
Manhattan	7.42	4.46	17.32
Window	15.53	13.15	9.43
3-Norm	6.04	5.40	9.91
Max	6.86	6.70	7.71
Average	8.32	6.94	11.12

TABLE III
The Figure of Merit (%) of Each Algorithm

	FD	VD	PVD
Euclidean	26.56	25.07	39.33
Manhattan	32.20	24.50	56.68
Window	59.01	53.56	36.85
3-Norm	26.89	25.28	37.27
Max	27.65	22.19	29.53
Average	34.46	30.12	39.93

From the simulation data presented in the tables, we can conclude that the proliferating detectors algorithm performs the best among the three different AIS algorithms, with the highest average detection rate (51.65%) and the average FOM index (39.93%). The variable-radius detectors algorithm yields the lowest average detection rate (37.05%), average false alarm rate (6.94%), as well as the average FOM (30.23%). This may be due to the fact that the parameters used in the algorithm are not well-selected. On the other hand, it is also observed that this algorithm requires the shortest simulation run-time, and thus may be selected for certain applications when quick solutions are needed in real-time. If time constraint is not a concern, then the proliferating V-detector algorithm can be the better choice.

The performance of each AIS algorithm also differs if different distance metrics are used. For example, by choosing an appropriate distance measure (i.e., "window"), the fixed-radius detectors algorithm can yield satisfactory performance (with a FOM of 59.01%). In fact, both the fixed-radius detectors and variable-radius detectors perform significantly better (with higher detection rate and overall FOM) if the "window" metric is chosen; while for the proliferating V-detector algorithm, the "Manhattan" is definitely a better choice.

V. Conclusions

In this research, algorithms based on artificial immune systems (AIS) and artificial neural networks (ANN) are employed for bioinformatics data mining. Three different variations of the real-valued negative selection algorithm and a feedforward neural network model are tested and compared via computer simulations. Though the neural network model yields the best overall result (with a FOM of 46.99%), the AIS algorithm is advantageous when only the "normal" (or "self") data is available while the neural network has to be trained using data of both "self" and "non-self" sets. The accuracy of an AIS algorithm may be further improved by optimizing (i.e., fine-tuning) the values of some of the parameters used in the algorithm. More testing will be conducted in the future.

Acknowledgment

This work was supported in part by the Leonard Transportation Center, under Award # GT 90907.

References

[1] Forrest, S., Perelson, A., Allen, L., R., "Self-nonself discrimination in a computer", In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA. 1994. pp 202–212

[2] Zhou, J., "Negative Selection Algorithms: from the Thymus to V-detector", Ph. D. dissertation, University of Memphis, Memphis, TN, USA, August 2006.

[3] Zhou, J., Dasgupta, D., "Revisiting Negative Selection Algorithms", Issue 15.2. Evolutionary Computation Journal. July, 2007.

[4] Gonzalez, F., Dasgupta, D., Nino, L. F., "A Randomized Real-Valued Negative Selection Algorithm", Second International Conference on Artificial Immune Systems. United Kingdom. September, 2003.

[5] Gonzalez, F., Dasgupta, D., Kozma, R., "Combining Negative Selection and Classification Techniques for Anomaly Detection", Volume 1, Congress on Evolutionary Computation. Honolulu, Hawaii. May, 2002. pp.705-710.

[6] Zhou, J., Dasgupta, D., "Applicability Issues of the Real-Valued Negative Selection Algorithms", Genetic and Evolutionary Computation Conference (GECCO). Seattle, Washington. July, 2006.

[7] Das, S., Gui, M., Pahwa, A., "Artificial Immune Systems for Self-Nonself Discrimination: Application to Anomaly Detection", Studies in Computational Intelligence (SCI) 116, 2008. pp229–246.

[8] "Statlib datasets archive", World Wide Web, http://lib.stat.cmu.edu/datasets.
[9] Gonzalez, F., Dasgupta, D., "Anomaly Detection Using Real-Valued Negative Selection", Journal of Genetic Programming and Evolvable Machines. Volume 4, Issue 4. December, 2003. pp383-403.
[10] Haykin, S., *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1999.
[11] *Neural Network Toolbox User's Guide*, The Mathworks.