Assessing the feasibility of approximating higher-order problem signatures in Artificial Neural Networks with hybrid transfer functions

Adamu A. S[1], M. Tomas[2] and Bargiela A[3]

**[1] Department of Computer Science, University of Nottingham Malaysia Campus
Semenyih, Selangor, Malaysia**

**[2] Department Of Computer Science, University Of Nottingham Malaysia Campus
Semenyih, Selangor, Malaysia**

**[3] Department Of Computer Science, University Of Nottingham Jubilee Campus
Wollatan Road, Nottingham, United Kingdom**

Abstract

Problem signatures are patterns that reveal a glimpse of the computational strategy most likely to be suitable for a given problem. Such a pattern could be the preferred choice of the activation and output functions for a given problem in neural networks that implement transfer functions optimization. We refer to these patterns as first-order signatures. Higher-order signatures capture information on a higher level, such as the likelihood of neural computational paths (i.e. connection between two or more transfer functions) used by the fittest models for specific problems. In addition, it also captures information about their weights.

In this paper, we show that higher-order problem signatures meet our proposed criteria for problem signatures: specifically, that the signatures of the different datasets tested have a lot of neural computation paths in common that makes them similar at a glance, but after thresholding their differences are more apparent. In addition to that, we also show that the signatures were consistent regardless of size of the population (P), number of runs (R), or size of the subsample used for approximating the signatures (N). However, in the case of the subsample size (N), we found that this was provided the sample size was fixed during sampling.

Keywords: *Meta-feature, Neural Network, Optimization, Transfer functions.*

1. Introduction

The transfer functions of Artificial Neural Networks play an important role in learning amongst other components. They enable neural networks to essentially compute decision boundaries in the input space; thus, giving it its ability to classify input data. The shapes and forms of these decision boundaries vary with the type of the transfer function being used. Traditional radial basis transfer functions effectively divide input space in a clustering-like manner. On the other hand, Perceptron's with linear transfer functions form decision boundaries with polygons when connected in multiple layers (i.e. MLP).

The input space of real world problems is complex and is typically not easily separable by hyper geometries produced by canonical neural networks. It is difficult for neural network to project the decision boundaries that accurately defines the problem. One of the simplest solutions to this problem when using canonical neural networks -such as the Multilayer Perceptron (MLP) - is to adapt the complexity of the neural network (e.g. by adding or removing more nodes and connections). However, this increases the risk of either over fitting or under fitting, which in turn results in poor generalization ability. In addition to that, there is also the issue of scalability, which goes hand-in-hand with efficient learning. Both of these characteristics (i.e. generalization ability and scalability) are critical goals for any machine learning system. Transfer functions optimization might hold a more efficient solution to this problem.

Approaches to transfer function optimization in neural networks can generally be classified into two categories; transfer function optimization by *parameterization*, or *hybridization*. Parameterization methods [1]–[3] focus on enhancing the flexibility of the transfer functions, thus enabling them to exhibit a wider range of decision boundary shape and form. Bi-radial transfer functions [4], [5] are an example of this; they could be regarded as variants of radial basis functions that have two centers. This was found to enhance the flexibility of the transfer function. Other studies include that of [2], where they adapted the exponent parameter of a sigmoidal function and evaluated their feed -forward neural network on two function approximation tasks. They found that this can lead to faster learning in FFANN [2]. A similar study was done by [3] where they used a q-exponential function that is capable of reproducing a Cauchy distribution amongst others. They found that the q-Gaussian model was very competitive when compared to other methods including support vector machines (SVM).

The second approach, transfer function optimization by hybridization, generally consists of approaches that use a blend of transfer functions in their neural networks. These are classified as Hybrid Artificial Neural Networks [6]. One example is the work of Gutierrez et al [7], where they used a blend of projection functions (sigmoid and product units) and kernel functions such as the radial basis function. They found that it was better on classification problems when compared to radial basis function (RBF) networks. A related work is Perceptron Radial Basis Net (PRBFN) by Cohen & Itrator [8] which also showed similar results. Maul [9] also proposed the use of projection and kernel functions, in addition to higher-order functions (such as higher-order product) in a framework termed: Neural Diversity Machines (NDM). Neuroevolution was used to optimize the neural networks weights, topology and choice of transfer functions. The results showed significant improvements compared to using multilayer perceptron's (MLP – Matlab Implementation). Other studies include [10] which also found that the resulting neural network models was more compact. The approach optimized the choice of transfer functions for the hidden layer nodes from a set of basis functions, while using either a sigmoidal or Identity output function for the output node in the output layer. A statistical pruning technique was also used to control the models complexity by removing nodes that were considered as not important [10].

Transfer function optimization can lead to increased dimensionality of the search space. This is because there are more possibilities of computational strategies introduced into the computational strategies search space as new transfer functions are added to the pool or flexibility is enhanced. The computational strategies search space consists of ways of projecting decision boundaries given the available transfer functions and their possible topologies. The increased dimensionality subsequently creates more local minima.

The dimensionality of the search space can be increased by both transfer function optimization methods. In the case of parameterization, the dimensionality is increased because there are a lot more parameters to control the shape and form the transfer functions' decision boundary. On the other hand, transfer function optimization by hybridization increases the dimensionality by the number of possible choices for transfer function of each node.

One approach used for handling local minima is to train artificial neural networks using evolutionary algorithms in what is known as Evolutionary Artificial Neural Networks[3], [11]–[19]. This is because evolutionary algorithms do not take into account gradients in their search. Another approach, which we propose in this paper, is to perform some preprocessing to discover the most likely neural computational biased and effective towards the given dataset. The transfer functions can then be restricted to those necessary for reproducing that neural computation strategy[20].

In this paper we evaluate the feasibility of discovering unique and consistent computational signatures for problems, which refer to as *problem signatures*. Specifically, our contribution concerns neural networks using transfer function optimization by hybridization.

Computational signatures can potentially be used to understand some of the neural computation strategies evolved in neural networks. In addition, it could also be used for determining the initial architectural state that is most likely best for the neural network before training; thus, improving convergence [20]. The proposed method extends previous contributions by introducing a preprocessing technique inspired by graph theoretical analysis methods used in neuroscience [21], [22]. We adopt these methods for the purpose of approximating the most inclined architectural properties (e.g. choice of transfer functions) for a given problem. This is done by randomly generating a population of neural networks with hybrid transfer functions and evaluating them without training. The fittest N subsample of the neural networks is then used to gather statistical information on the architectural properties of this subsample of the population. Repeated independent runs of subsampling enables us to approximate some architectural properties that seemed to be associated with each problem.

The organization of the paper is as follows: firstly we define the hybrid neural network used for experiments (i.e. Neural Diversity Machine Networks). This is followed by definitions of higher-order problem signatures, thresholding and proposed criteria for problem signatures. The next section describes the experimental setup and is followed by the results section where we reveal the results of the experiments. Afterwards, discussions on the results are made and conclusions drawn in the discussion and conclusion sections, respectively.

In this paper, we refer to *transfer function* as the compound function, $f(g(.))$ which consists of: the *activation function*, $g(.)$ and the *output function*, $f(.)$. We also refer *to neural computation path* as the connection path between two or more nodes. *Signature* is also used synonymously with pattern. However, in the case of signatures we are referring to a specific pattern rather than a generic one.

2. Methodology

In this section, we define higher-order problem signatures and our proposed criteria for problem signatures if they are to be regarded as features of problems. In addition to this, we also define thresholding – a popular tool also used in graph theoretical analysis of the brain [22], [23]. However, prior to this we define Neural Diversity Machines (NDM) [9] –the hybrid Neural Network used for the experiments.

2.1 Neural Diversity Machine Networks

In this paper, we use an NDM [9] as our neural network. A Neural Diversity Machine is essentially an architecture that is flexible in its constraints of the neural networks; primarily, in the choice of transfer function, and network topology. An NDM can adopt any combination of activation and output function. In addition to that, connections between any two nodes are unrestricted. This gives it some flexibility in its bias.

The topology at initialization is a full-connectivity topology with each node from the previous layer connecting to every node in the next layer. It is also worth noting that NDM has a single hidden layer at initialization. The topology and the number of layers of NDMs can be adapted to suite the problem during Neuroevolution. The architecture is also recurrent: the hidden layer is accompanied by a context layer for storing the outputs of the hidden units of the previous time frame; these outputs are used as inputs for the next time frame. The activation functions and output functions used in NDMs are listed in the tables (see table 1 & table 2).

The genetic string of NDMs consists of information about: the fitness gene, connection weights, connection status, bias of nodes, node status, and node transfer functions: i.e. choice of activation and output function (See table 3).

TABLE 1
ACTIVATION FUNCTIONS $a_j = g(W, I)$

Index	Functions	Definitions
1	Inner Product	$a_j = \sum_i^n w_i i_i + bias_i$
2	Euclidean Distance	$a_j = \sqrt{\sum_i^n (w_i - i_i)^2}$
3	Higher-Order Product	$a_j = \prod_i^n c w_i * i_i$
4	Higher-Order Subtractive	$a_j = \sum_i^n \lvert w_1 i_1 - w_i i_i \rvert$
5	Standard Deviation	$a_j = stdDev(w_i i_i \cdots w_n i_n)$
6	Min	$a_j = min(w_i i_i \cdots w_n i_n)$
7	Max	$a_j = max(w_i i_i \cdots w_n i_n)$

Activation functions available in the pool of NDMs for neuroevolution of transfer functions during learning.

TABLE 2
OUTPUT FUNCTIONS $y_j = f(a_j)$

Index	Functions	Definitions
1	Identity	$y_j = k * a_j$
2	Sigmoid	$y_j = \frac{c}{1 + e^{-\kappa * a_j}}$
3	Gaussian	$y_j = e^{\frac{-a_j^2}{\omega}}$
4	Tanh (Hyperbolic tangent)	$y_j = \frac{1 - e^{-\kappa * a_j}}{1 + e^{-\kappa * a_j}}$
5	Gaussian II	$y_j = \begin{cases} a_j = e^{\frac{-a_j^2}{\omega}}, & a_j < \vartheta \\ a_j = 0, & a_j \geq \vartheta \end{cases}$

Output functions available in the pool of NDMs for neuroevolution of transfer functions during learning.

TABLE 3
GENETIC STRING CONTENT

	Information	Definitions
1	Fitness	Cost of the model on the given problem.
2	Connection weights	Weights between all connections in the model.
3	Connection status	Status of the connections, either active or deactivated.
4	Node bias	Bias values of nodes their weights.
5	Node status	Status of the node, either active or deactivated.
6	Node activation function	Activation functions adopted by the nodes.
7	Node output function	Output functions adopted by the nodes.

Some of the contents stored on the genetic string about NDM models.

2.2 Higher-Order problem signatures

In neuroscience, part of the analysis done to understand the brain's network relies on the use of graph theory [21], [22]. Regions of interest in the brain are defined and their structural or functional connectivity are represented in a two dimensional connectivity matrix.

We use a similar approach that also involved some graph theoretical analysis for discovering the signatures of problems. This produced two sorts of higher-order signatures in the form of matrices: coexist-on-path matrix, and connection strength matrix.

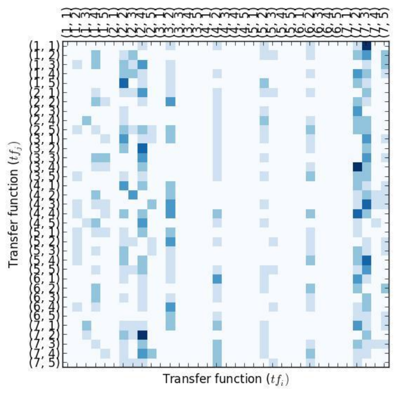


Fig.1. A coexist-on-path matrix showing the frequency of use for computation paths (from the hidden to the output layer) in the Top N neural network models from a randomly initialized population of Neural Diversity Machines evaluated on the Iris dataset (where N, number of solution samples with the fittest cost).

The coexist-on-path matrix represents information about the frequency of connections between any two transfer functions, and the direction of that connection (see Fig.1). The transfer function is represented as a tuple (a, b) where, a and b are the indices of the activation function and output function, respectively (see table 1 & table 2). The direction of the connection is read from the y-axis to the x-axis. Darker regions indicate higher values, while lighter regions indicate lower values. In essence, this measure only grabs the likelihood of two transfer functions coexisting on the same path in the subset of the most accurate subset of models evaluated without training.

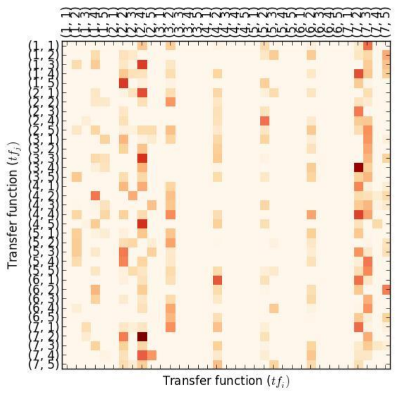


Fig. 2. An example of a connection strength matrix showing the weights of some neural computation paths (from the hidden to output layer) accumulated over the process of extracting signatures from the fittest (i.e. Top N) of randomly initialized NDM networks evaluated on the Iris dataset.

Connection strength is similar to coexistence-on-path in representation, though it grabs information about the accumulated weight between connections of any two transfer functions (see Fig. 2). In essence, these two tools enable us to glimpse at some part of the neural computation strategies at work by knowing about their neural computation paths. The pattern extracted by these tools is what we refer to as a problem signature. In this case, these signatures are higher-order problem signatures since they are signatures that reveal some information about connection paths and their direction. More atomic signatures, which we refer to as first-order problem signatures [20] reveal information on the combination of activation and output functions. However, the focus of this paper is on higher-order problem signatures.

In this paper, we used subsampling to select solutions from a population of randomly initialized models of NDM networks. We select N ($N = [1,2,3,4]$) of the top NDM models after randomly generating architectures from the pool of transfer functions and evaluating them without training on a given problem. Due to NDMs having a hybrid and diverse pool of transfer functions, it is possible to have a variation of neural network architectures with different transfer functions which in turn implies that there are more computation strategies available. By computation strategies in the context of classification problems, we mean ways of projecting decision boundaries that can divide the input space to some extent into the respective classes that describes the underlying principle of the problem that generated the datasets.

The potential significance of problem signatures to efficient learning in neural networks with hybrid transfer functions is that it could be used to determine a suitable initial state for training from which the complexity can be increased as required in a manner similar to that of NEAT [14]. This has the potential to improve convergence in addition to revealing interesting information about the relationship between problems in the context of computational strategies. Thus, it can potentially be used as a measure to estimate the distance between problems. Using Multidimensional Scaling (MDS), one might be able to visualize the problem-computational strategy space.

2.3 Problem signature criteria

As is apparent in the visualizations (see Fig. 1 & Fig. 2) of the coexist-on-path and connection strength matrices, there is a consistent pattern: some strongly shaded vertical regions that show some neural computation paths being more likely to be found in the elite subsample of NDM models than others.

The working hypothesis in this paper is that these signatures would differ between problems that are not related (i.e. in terms of the computation strategy applicable to solving them), and be similar between related problems. We also speculated that the difference or similarity between problems would be proportional to the degree of their relationship. Finally, we also expected the signatures for a problem to be consistent.

In summary, we hypothesize that the criteria for problem signatures to be feasible as a reliable description of the computational strategy for problems are as follows:

1. *Consistency* of signatures belonging to a problem.
2. *Discrimination/difference* between signatures belonging to non-related problems.
3. And vice versa; *similarity* between signatures of problems that are related.

2.4 Thresholding

Thresholding is another common technique used in various analysis [21] to reveal more pronounced features by filtering out some that are below the threshold. The thresholding function implemented is given by the simple equation below:

$$M'(i,j) = \begin{cases} 0, & M(i,j) < \theta \\ M(i,j), & M(i,j) \geq \theta \end{cases}$$

Where, M is a NxN matrix, and θ is the threshold.

In this paper, thresholding was done to filter out the values that were below the mean (μ) to reveal information that could be of interest. The mean was chosen because it is relative to the range of intensities in the matrix. Thus, it makes it unlikely to filter out some information that might be important or not filter out signatures that are not useful.

We also used another parameter; the *plus value* (α), which can be used to increase the threshold (θ). Such that,

$$\theta = \mu + \alpha$$

Thus: when $\alpha = 0$, then $\theta = \mu$.

3. Experimental setup

The datasets used for the experiments were: Iris, Sonar and XOR dataset. The Iris and Sonar datasets were retrieved from the UCI machine learning repository [24].

For each of the datasets, the problem signature was extracted by generating a random population of genetic strings with gene values ranging from -1.0 to 1.0. The size of the population generated, P is fixed. Some of the architectural properties encoded include, but are not limited to the following: the connection weights, the status of connections (i.e. on or off), status of nodes (i.e. switched on or off), the activation and output function of each node and its bias, and fitness of the gene.

These genes are then decoded and evaluated on the dataset without training. A subsample of the solutions (particularly, the top N solutions) are then selected for extracting problem signatures. The process of extracting signatures simply records statistics of the likelihood and direction of connections between transfer functions (i.e. coexist-on-path) and the connection weights between them (i.e. connection strength) from the samples. This is repeated for a number of runs, R.

The experiments carried sought to answer if the following affected the signatures integrity:

1. Subsample size **N** ($1 \leq N \leq 4$).
2. Population size at initialization, **P** ($P = [100, 200, 300, 400, 500, 700, 1000, 1500]$).
3. Number of subsampling runs, **R** ($R = [5, 10, 15, 25]$).

We also applied thresholding to see if the change in plus value, α ($0.0 \leq \alpha \leq 2.0$) affects the signatures.

4. Results

In this section, we present the results of our preliminary feasibility assessments of problem signatures on the Iris, Sonar and XOR datasets.

4.1 Signatures' Discriminatory/Similarity Criteria and Thresholding

It was observed that some neural computation paths were generally used for all the datasets consistently (i.e. for Iris, Sonar, and XOR – see Fig. 3, Fig. 4 & Fig. 5).

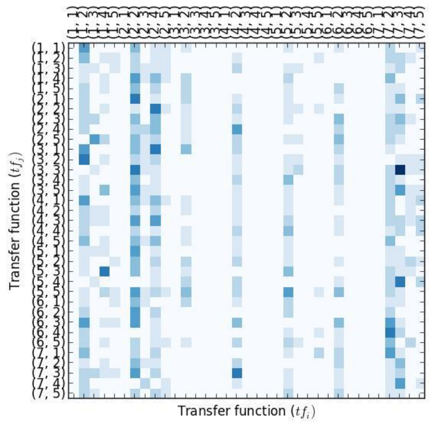


Fig. 3. Coexist-on-path matrix of the Iris dataset showing vertical patterns indicating the connectivity patterns usually used.

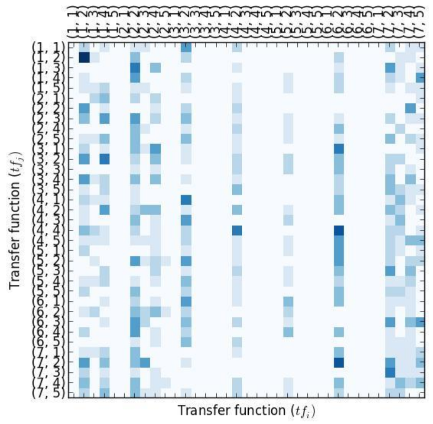


Fig. 4. Coexist-on-path matrix of the Sonar dataset also showing a similar pattern as the Iris, however with varied intensities at specific regions.

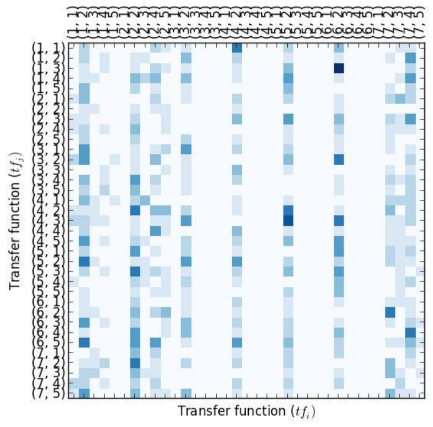


Fig. 5. Coexist-on-path matrix of the XOR dataset also showing a similar pattern with some varied regions.

However, by applying the thresholding function; the neural computation paths unique to each problem were more apparent. Thus, suggesting that the computational strategies used for problems were likely different. The illustrations (Fig. 6 & Fig. 7) show how the correlation between problems decreases as the threshold (θ) is increased.

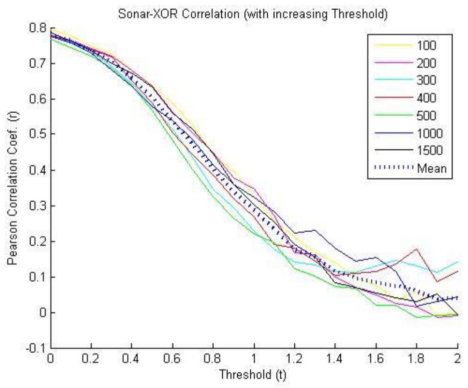


Fig. 6. Sonar-XOR correlation as threshold is increased by increasing the plus value α. The illustration shows the results of the varying sizes of the population size, P ($P = [100, 200, 300, 400, 500, 700, 1000, 1500]$)

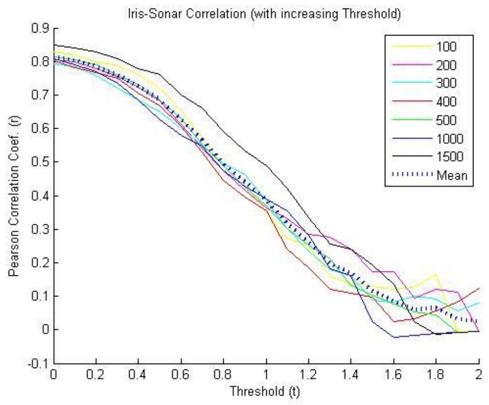


Fig. 7. Iris-Sonar correlation as threshold is increased by increasing the plus value α ($\alpha = [0.1, 0.2, 0.3 \dots 2.0]$).

Another observation was that the coexist-on-path and connection strength matrices were strongly correlated. A Pearson correlation coefficient (r) test on both of them was found to confirm this strong correlation for all the datasets tested (see table 4). This also means that the relationship between the two is linear. This is because the Pearson correlation coefficient measures degree of linear relationships.

TABLE 4
PEARSON CORRELATION COEFFICIENT

	Iris	Sonar	XOR
Pearson Correlation Coefficient (r)	0.978	0.982	0.982

2D Pearson correlation coefficient (r) between the coexist-on-path and connection strength matrices for the Iris, Sonar and XOR datasets. A correlation above 0.5 is usually considered to be a strong correlation.

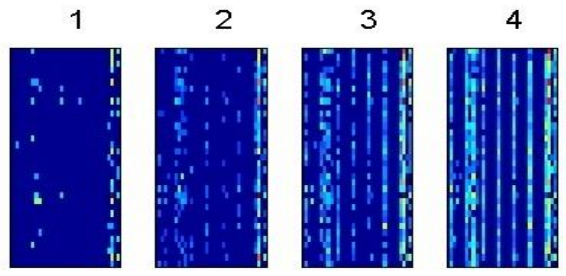


Fig. 8. Heat map showing the visualization of the coexist-on-path matrix for the Iris dataset as the subsample size, N increases; more vertical patterns can be seen as N increases indicating heavier use of some neural computation paths.

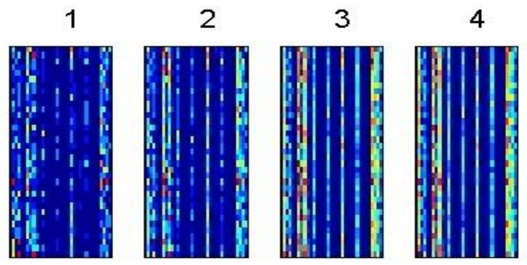


Fig. 9. Heat map showing the visualization of the coexist-on-path matrix for the Sonar dataset as the subsample size, N increases; a similar pattern can be observed here as well.

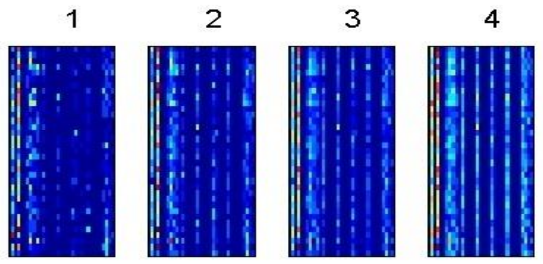


Fig. 10. Heat map of the coexist-on-path matrix for the XOR dataset as the subsample size, N increases: the pattern of increasing intensities is also noticeable here.

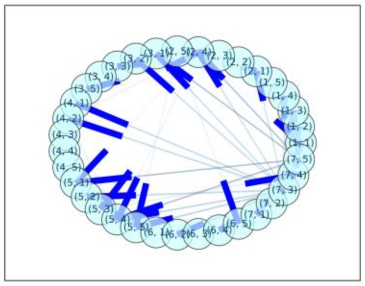
4.2 Signature Consistency and Size of Subsample

Tests made with various population sizes, P $(100, 200, 300, 400, 500, 700, 1000, 1500)$ showed that it had no significant effect on the signatures. Additionally, another experiment that involved adjusting the number of runs, R $(5, 10, 15, 20, 25)$ also showed no significant distortion of the signatures.

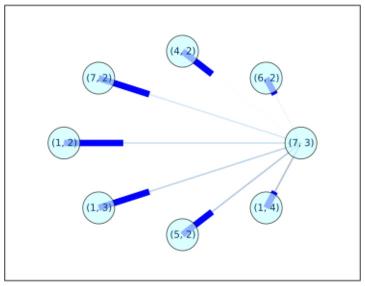
The size of the subsample N (i.e. $N = [1, 2, 3, 4]$) had some effect on the signature, i.e. the vertical shaded regions are increased as the subsample size is increased (see Fig. 8, Fig. 9

& Fig. 10); this further suggests that other NDM models in the population were using different computational strategies. However, the consistencies of the signatures were maintained: specifically, neural computation paths were still consistent as the subsample size was increased. It appeared to be that only neural computation paths were being detected. To further analyze this, we use some graph theory to analyze both the coexist-on-path and connection strength matrices.

By using a weighted Networkx [25] directed graph we can visualize the coexist-on-path and connection strength matrices all together with the help of Matplotlib [26]. However, the graph was cluttered with lots of nodes and connections (see Fig. 11a, Fig. 12a & Fig. 13a), so we used thresholding to filter out some of them. This was done by only focusing on the node with the highest connection density (i.e. largest frequency of being connected to) as the sole terminal node. In other words, we only considered neural computational paths with connections to the node with the most connection density as their terminal node. Furthermore, only nodes connected more frequently than average were included. This thresholding operation was necessary for confirming that different problems did indeed use different neural computation paths. It also helps in unveiling more interesting results. The results post thresholding can be seen in the illustrations below (see Fig. 11b, Fig. 12b & Fig. 13b). The directionality of the connection is represented by the thicker end being the origin to the thinner end of the line being the target.

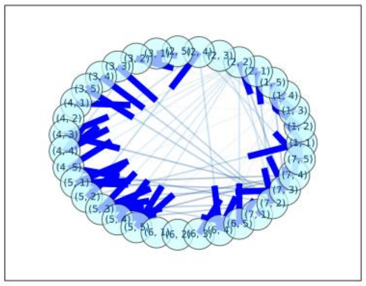


(a)

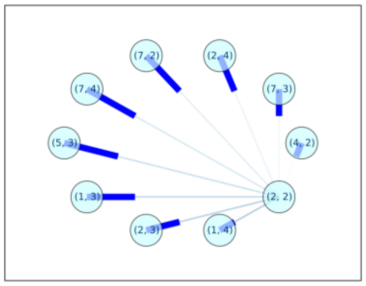


(b)

Fig. 11. The graph for the Iris dataset: (a) the graph before thresholding, (b) the graph after thresholding. The *Thickness* of the connection represents the *connection strength*.

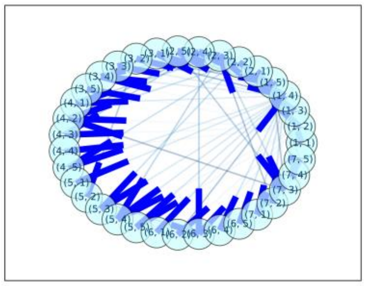


(a)

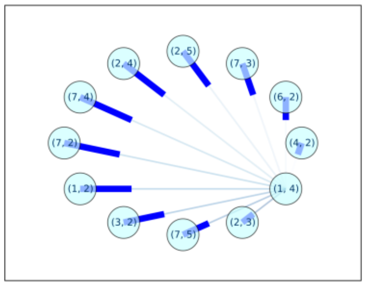


(b)

Fig. 12. Graph for the Sonar dataset: (a) graph prior to thresholding, (b) graph post thresholding.

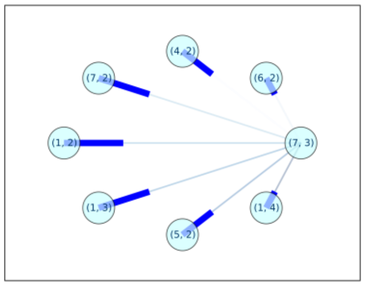


(a)

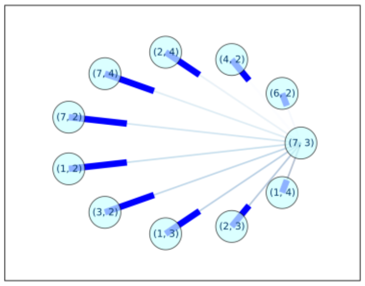


(b)

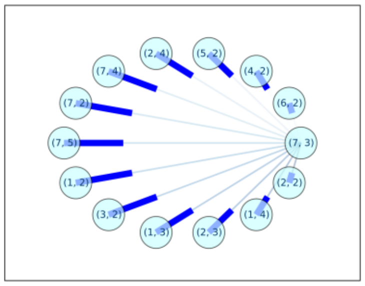
Fig. 13. Graph for XOR dataset: (a) pre-thresholding, (b) post-thresholding.



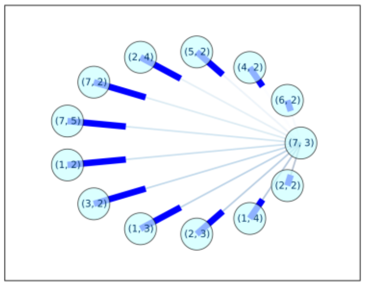
(a) N = 1



(b) N = 2



(c) N = 3



(d) N = 4

Fig. 14. Graphs for the Iris datasets as the subsample size N is increased: (a) N = 1, (b) N = 2, (c) N = 3, (d) N = 4.

5. Discussion

In terms of the signatures discriminatory/similarity criteria, higher-order problem signatures were found to have better discriminatory ability as the threshold was increased (see Fig.6 & Fig. 7). This was done by increasing the plus value α (i.e. specifically between the range of $0.0 \leq \alpha \leq 2.0$). The results were consistent regardless of the population size, P. The correlation decreases almost linearly as the threshold was increased.

The fact that the signatures were similar (i.e. vertical shaded regions of the graphs indicating some similarity in neural computation paths and the strong correlation prior to thresholding) for all the problems suggest that some neural computation paths are generic in their mode of use for the problem tested. Consequently, it also shows that other neural computation paths were more suitable for specific problems – as shown after thresholding filtered out the generic ones. This suggests that different computational strategies were being used as can be seen in the illustrations (see Fig. 11b, Fig. 12b & Fig. 13b).

As seen in the results, the pattern reveal vertical patterns suggesting the heavy connection of other transfer functions to a specific subset of transfer functions.

For vertical patterns to appear in the matrices (i.e. coexist-on-path and connection strength) there has to be a set of vertices, V_V that is a subset of the all the vertices set V, which consists of vertices that have a set of edges, E_V consisting of connections that connects others vertices from the global set V connects to a significant number of vertices in the set of vertices, V_V.

The fact that vertical patterns were more prominent in the results of all the datasets is as a result of the fully connected topology of the initialized population of NDMs which also had a single hidden layer. Each solution is initialized with random transfer functions, weights, and bias values; however, in this experiment they are fully connected at initialization and there was no training operation applied to change this topology. Given the fact that the population used had single layers, and that statistics of problem signatures was only captured for the hidden to output layer connections; the neural computation paths presented in the Figures (Fig. 11b, Fig. 12b and Fig. 13b) represent to a large extent the most biased and fittest models (or instances of computational strategies) at initialization towards the given datasets.

Regardless of the commonality of the architectural constraints used for all the datasets, there was bias of some neural computation paths for each problem (see Fig. 11b, Fig. 12b and Fig. 13b).

In the case of the Iris, the signatures after thresholding and visualization as a graph revealed that the node with the most input connection density was one that used an unusual combination of a Max with Gaussian output function. Some of the nodes that connected to it often used; Euclidean distance with sigmoid, Inner product with Gaussian, Max with Gaussian II, Standard deviation with sigmoid, and Euclidean distance with Tan. The ones that connected to it the least used: Higher-Order subtractive with sigmoid, and Min with Sigmoid. In earlier studies [9], it was observed that the Max and Min functions were used as relay functions when combined with a linear output function, such as Identity for some synthetic 2D datasets. In other occasions involving real world datasets such as the Australian credit card, they have been found to be utilized as filter-like functions when the difference between the ranges of input parameters is large. The filtering function enabled the neuron to filter out all other inputs parameter. In the case of the Iris, this seems to be the case; assuming the neuron with this transfer function is fed directly from the input layer and that the weights are not significantly different from each other, then the sepal length parameters could be the input parameter being favored over the others. This is because the sepal lengths' mean and min is greater than the mean and min values of the other parameters, which include: sepal width, petal length and petal width. If the output function for the node is a Gaussian, it means that the sepal length values closer to the Gaussian function's center would get higher outputs, while those with values farther away from the center would get the lowest outputs. If the output function was a Gaussian II (Gauss II), then there would be a bunch of values that would result in high outputs. This is because the Gauss II in essence flattens the peak of the Gaussian function; this in turn means the center is much wider. In other words, a neural computation node using Max with Gaussian/Gaussian II functions could be providing normalized outputs of petal lengths. A node with Min is also likely to act in the same way, except that it would likely be working with the input parameter with the lowest mean, which is petal width in this case. Other similar variants such as Max/Min with Sigmoid/Hyperbolic Tangent have a related functionality, except that their normalization is by squashing the outputs within the range of their outputs.

In the case of the Sonar dataset, the results reveals that the highest contributor nodes to the most densely connected to node, which was one that used an Euclidean distance with sigmoid, included nodes using: HO Product with sigmoid (HO Unit), HO Subtractive with Sigmoid (HO Unit-variant) Euclidean distance with Gaussian (RBF Unit), Inner product and tanh (Perceptron), and Euclidean distance and Hyperbolic Tangent.

As for the XOR dataset, the most likely node to have the highest input connection density from other nodes was one that used an inner-product with a hyperbolic tangent (Perceptron node). The nodes most likely to connect to it in a path included nodes that used the following transfer functions: Euclidean distance or Max as the activation function with Gaussian/Gaussian II or Sigmoid/Tan as the output function. Basically, they were mostly Radial Basis function unit, and filter functions (such as Max with Gaussian/Gaussian II or Sigmoid/Tan). The Euclidean distance with sigmoid/Tanh function can be classified as a function that creates new information; in this case the summed distance between the input vector and the weight vector. The summed distance is then squashed by a sigmoid/Tanh. In the case of the XOR, the Euclidean distance could be valuable for finding simple computational strategies for the dataset. Assuming the weights between the input and hidden layer are all either $(1,1)$ or $(0,0)$; the Euclidean distance for the input values $(1,1)$ and $(0,0)$ would both produce a number of even parity, while $(1,0)$ and $(0,1)$ would produce a number of odd parity. This is valuable information that can be used to discriminate between the two classes for the given dataset.

In summary, a lot of these "unpopular" transfer functions are in essence performing some sort of filtering to the inputs. Specifically, they are usually being used to relay the normalized values of particular inputs from the input vector, while ignoring the rest. It would be interesting to see how a layer dedicated to evolving filter functions as preprocessors might improve the scalability and generalization of artificial neural networks. Each input node could be connected to a single preprocessing unit in the preprocessing layer, which can evolve any sort of transformation function from the activation and output functions pool. Multiple layers of these preprocessors with each layer having different connection topologies could also be explored. This is likely to appear in future works.

Another interesting observation was the relationship of the size of the subsample and the higher-order problem signatures. It can be seen in the illustration (see Fig. 14.) that as the size of the subsample increases, so do the neural computation paths in the graph. The fact that there is a significant number of the neural computation paths still consistent in the graphs as N grows, suggests that higher-order problem signatures are consistent (see Fig. 11, Fig. 12, & Fig. 13). The difference is that new neural computation paths are introduced. Keeping in mind that the subsamples are chosen according to fitness, it can be speculated that neural computation paths have fitness too or at least an associated fitness. However, measuring this is another topic. Consequently, it can also be speculated that neural computation strategies - made up of these correction paths - also have fitness that can be measured too. However, this is not in the scope of this paper and might appear in future works.

Considering the results of the size of the subsample (N), population size (P), and number of runs (R): we have found that the signatures are consistent. In addition, we have also shown that thresholding can be used to make them dissimilar/similar. However with regards to consistency with respect to the size of the subsample (N); the higher-order signatures are consistent given that the size of the subsample is kept fixed during sampling. Varying the size might results in some dormant signatures to become more pronounced. Thus, introducing new neural computation paths which could some cause inconsistencies between signatures from different runs.

6. Conclusion

In conclusion, we have shown that higher-order problem signatures described have met the problem signatures criteria proposed (see section 2.3) for the datasets tested: specifically, that these signatures have a lot of neural computation paths in common that makes them similar at a glance, but after thresholding their difference are more apparent. As for signature consistency; the signatures were found to be consistent regardless of size of the population (P), number of runs (R), or subsample size (N). However, in the case of the subsample size (N), we have found that new neural computation paths that were dormant can be introduced into the signatures as the size is increased. Thus, it can be regarded as consistent if sample size is fixed during sampling. In other words, making the subsample size (N) variable during sampling process could introduce some inconsistency.

In a nutshell, we have found that problem signatures could be analogous to thumb prints: similar at a glimpse because of the same pattern of whirls radially moving out from the center of the thumb, but they different when we are more specific of regions to look at. Future works might tell.

7. Future works

In the future we plan to introduce more variations of the higher-order problem signatures. Specifically, we intend to use two more coexist matrices that capture information not based on path, but on coexistence in layer or in the network model as a whole. This should provide rich information that would reveal more relationships between transfer functions and their role in neural network models in the context of various problems.

Acknowledgment

We would like to acknowledge the role of the family and friends of all the authors in support of this humble piece of contribution to knowledge.

References

[1] W. Duch and N. Jankowski, “Bi-radial transfer functions,” *Proc. Second Conf. neural networks their Appl.*, pp. 131–137, 1996.

[2] P. Chandra and Y. Singh, “A case for the self-adaptation of activation functions in FFANNs,” *Neurocomputing*, vol. 56, pp. 447–454, Jan. 2004.

[3] F. Fernández-Navarro, C. Hervás-Martínez, P. a Gutiérrez, and M. Carbonero-Ruz, “Evolutionary q-Gaussian radial basis function neural networks for multiclassification.,” *Neural Networks*, vol. 24, no. 7, pp. 779–84, Sep. 2011.

[4] N. Jankowski, “Flexible transfer functions with ontogenic neural networks,” *Toru, Pol.*, vol. 1, no. 6, pp. 1–6, 1999.

[5] W. Duch and N. Jankowski, “Transfer functions: hidden possibilities for better neural networks,” *9th Eur. Symp. Artif. Neural Networks*, pp. 81–94, 2001.

[6] P. A. Gutiérrez and C. Hervás-Martínez, “Hybrid Artificial Neural Networks : Models , Algorithms and Data,” *Lect. Notes Comput. Sci.*, vol. 6692, no. PART 2, pp. 177–184, 2011.

[7] P. Gutiérrez, C. Hervás, M. Carbonero, and J. Fernández, “Combined projection and kernel basis functions for classification in evolutionary neural networks,” *Neurocomputing*, vol. 72, no. 13–15, pp. 2731–2742, 2009.

[8] S. Cohen and N. Intrator, “A Hybrid Projection-based and Radial Basis Function Architecture: Initial Values and Global Optimisation,” *Pattern Anal. Appl.*, vol. 5, no. 2, pp. 113–120, Jun. 2002.

[9] T. Maul, “Early experiments with neural diversity machines,” *Neurocomputing*, Mar. 2013.

[10] N. Jankowski and W. Duch, “Optimal transfer function neural networks,” in *In 9th European Symposium on Artificial Neural Networks*, 2001, no. I, pp. 101–106.

[11] X. Yao, “Evolving Artificial Neural Networks,” in *Proceedings of the IEEE*, 1999, vol. 87, no. 9, pp. 1423–1447.

[12] S. Nolfi and D. Parisi, “Evolution of Artificial Neural Networks,” *Handb. Brain Theory Neural Networks*, 2002.

[13] X. Yao and Y. Liu, “Towards designing artificial neural networks by evolution,” *Appl. Math. Comput.*, vol. 91, no. 1, pp. 83–90, 1998.

[14] K. O. Stanley and R. Miikkulainen, “Efficient Reinforcement Learning Through Evolving Neural Network Topologies,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, 2002.

[15] X. Yao, “A review of evolutionary artificial neural networks,” *Int. J. Intell. Syst.*, vol. 8, no. 1, pp. 539–567, 1993.

[16] H. Abbass, “Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective

optimization," in *The 2003 Congress on Evolutionary Computation 2003 CEC 03 (2003)*, 2003, vol. 3, no. 4, pp. 2074–2080.

[17] G. Brown, "Diversity in neural network ensembles," 2004.

[18] A. Chandra and X. Yao, "Ensemble Learning Using Multi-Objective Evolutionary Algorithms," *J. Math. Model. Algorithms*, vol. 5, no. 4, pp. 417–445, Mar. 2006.

[19] M. M. I. Islam, X. Yao, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles.," *IEEE Trans. Neural Networks*, vol. 14, no. 4, pp. 820–34, Jan. 2003.

[20] A. S. Adamu, T. H. Maul, and A. Bargiela, "On Training Neural Networks with Transfer function Diversity," in *International Conference on Computational Intelligence and Information Technology (CIIT 2013)*, 2013.

[21] M. Rubinov and O. Sporns, "Complex network measures of brain connectivity: uses and interpretations.," *Neuroimage*, vol. 52, no. 3, pp. 1059–69, Sep. 2010.

[22] O. Sporns, "Graph theory methods for the analysis of neural connectivity patterns," in *Neuroscience Databases*, 2003, pp. 169–183.

[23] E. Bullmore and O. Sporns, "Complex brain networks: graph theoretical analysis of structural and functional systems.," *Nat. Rev. Neurosci.*, vol. 10, no. 3, pp. 186–98, Mar. 2009.

[24] K. Bache and M. Lichman, "{UCI} Machine Learning Repository." 2013.

[25] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using {NetworkX}," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 2008, pp. 11–15.

[26] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007.

Adamu A.S received his B.S (Hons.) degree in computer science from the University of Nottingham – Malaysia Campus in 2011 and is currently a PhD candidate at the same university with the faculty of computer science. His current research interests include: neural networks, optimization and some other machine learning fields such as image processing.

M. Tomas received his M.Sc. degree in Computer Science from Imperial College London, United Kingdom; and a Ph.D. in computer science from the University Malaya, Malaysia. Currently, he is an assistant professor with University of Nottingham - Malaysia Campus, Malaysia.

Bargiela A. was conferred full professorship from the president of Poland in October 2005. He has worked with several universities including, Nottingham Trent University, and the University of Durham. Currently he is an associate professor with University of Nottingham - Jubilee Campus, United Kingdom.