2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

LEARNING OPPOSITES USING NEURAL NETWORKS

Shivam Kalra[†], *Aditya Sriram*[†],*Shahryar Rahnamayan*[‡], *H.R. Tizhoosh*[†]

[†] KIMIA Lab, University of Waterloo, Canada
[‡] Elect., Comp. & Software Eng., University of Ontario Institute of Technology, Canada

ABSTRACT

Many research works have successfully extended algorithms such as evolutionary algorithms, reinforcement agents and neural networks using "opposition-based learning" (OBL). Two types of the "opposites" have been defined in the literature, namely *type-I* and *type-II*. The former are linear in nature and applicable to the variable space, hence easy to calculate. On the other hand, type-II opposites capture the "oppositeness" in the output space. In fact, type-I opposites are considered a special case of type-II opposites where inputs and outputs have a linear relationship. However, in many real-world problems, inputs and outputs do in fact exhibit a nonlinear relationship. Therefore, type-II opposites are expected to be better in capturing the sense of "opposition" in terms of the input-output relation. In the absence of any knowledge about the problem at hand, there seems to be no intuitive way to calculate the type-II opposites. In this paper, we introduce an approach to learn type-II opposites from the given inputs and their outputs using the artificial neural networks (ANNs). We first perform *opposition mining* on the sample data, and then use the mined data to learn the relationship between input x and its opposite $\breve{x}$. We have validated our algorithm using various benchmark functions to compare it against an evolving fuzzy inference approach that has been recently introduced. The results show the better performance of a neural approach to learn the opposites. This will create new possibilities for integrating oppositional schemes within existing algorithms promising a potential increase in convergence speed and/or accuracy.

1. INTRODUCTION

A large number of problems in engineering and science are unapproachable with conventional schemes instead they are handled with intelligent stochastic techniques such as evolutionary, neural network, reinforcement and swarm-based algorithms. However, essential parameter for the end users of aforementioned intelligent algorithms is to yield the solutions within desirable accuracy in timely manner – which remains volatile and uncertain. Many heuristic methods exist to speed up the convergence rate of stochastic algorithms to enhance their viability for complex real-world problems. *Opposite Based Computing (OBC)* is one such heuristic method introduced by Tizhoosh in [1]. The underlying idea is simultaneous consideration of guess and opposite guess, estimate and opposite estimate, parameter and opposite parameter & so on in order to make more educated decisions within the stochastic processes, that eventually results in yielding solutions quickly and accurately.

In essence, learning the relationship between an entity and its opposite entity for a given problem is a special case of *a-priori knowledge*, which can be beneficial for computationally intelligent algorithms in stochastic setups. In context of machine learning algorithm, one may ask, why should effort be spent on extraction of the opposite relations when input-output relationship itself is not well defined? However, various research on this topic has shown that simultaneous analysis of entities and their opposites can accelerate the task in focus – since it allows the algorithm to harness the knowledge about symmetry in the solution domain thus allowing a better exploration of the solutions. *Opposition-based Differential Evolution (ODE)*, however, seems to be the most successful oppositional inspired algorithm so far [2].

There have been two types of opposites defined in literature **1)** *type-I* and **2)** *type-II*. Generally, most learning algorithms have an objective function; mapping the relationship between inputs and their outputs – which may be known or unknown. In such scenario, type-I based learning algorithms deal with the relationship among input parameters, based on their values, without considering their relationship with the objective landscape. On contrary, type-II opposite requires a prior knowledge of the objective function. Until 2015, all papers published on using OBL employed the simple notion of type-I opposites which are conveniently, but naively defined on the input space only, making a latent linearity assumption about the problem domain. Tizhoosh and Rahnamayan [3] introduced the idea of "opposition mining" and evolving rules to capture oppositeness in dynamic environments.

The paper is organized as follows: Section 2 provides a literature review on OBL. Section 3 introduces the idea to use artificial neural network (ANN) to learn the opposites, and provides an overview of type-I and type-II OBL. Finally, Section 5 provides experimental results and analysis and also a comparison of the proposed ANN approach with the evolving fuzzy inference systems, a method recently proposed in [3].

2. BACKGROUND REVIEW

Roughly 10 years ago, the idea of opposition-based learning (OBL) was introduced as a generic framework to improve existing learning and optimization algorithms [1]. This approach has received a modest but growing attention by the research community resulting in improving diverse optimization and learning techniques published in several hundred papers [4]. A few algorithms have been reported to employ "oppositeness" in their processing, including reinforcement learning [5, 6, 7, 8], evolutionary algorithms [9, 10, 11, 12], swarm-based methods [13, 14, 15], and neural networks [16, 17, 18].

The majority of learning algorithms are tailored toward approximating functions by arbitrary setting weights, activation functions, and the number of neurons in the hidden layer. The convergence would be significantly faster towards the optimal solution if these random initializations are close to the result [18]. On the contrary, if the initial estimates are far from the optimal solution – in the opposite corner of the search space, for instance, then convergence to the ideal solution will take considerably more time or can be left intractable [19]. Hence, there is a need to look simultaneously for a candidate solution in both current and opposite directions to increase the convergence speed – a learning mechanism denoted as "opposition-based learning" [1]. The concept of OBL has touched upon the various existing algorithm, and it has proven to yield better results compared to the conventional method of determining the optimal solution. A detailed survey on applications of OBL in soft-computing is discussed by Al-Quanaieer et al. in [4]. The paper discusses the integration of OBL when used for reinforcement learning, neural networks, optimization, fuzzy set theory and fuzzy c-mean clustering. A review on each algorithm states that applying OBL can be beneficial when applied in an effective manner when applications use optimization algorithm, learning algorithm, fuzzy sets and image processing.

Many problems in optimization involve minimization or maximization of some scalar parameterized objective function, with respect to all its parameters. For such problems, OBL can be used as a heuristic technique to quickly converge to the solution within the search space by generating the candidates solutions that have "opposite-correlation" instead of being entirely random.

The concept of "opposite-correlation" can be discussed from type-I and type-II perspectives when an unknown function $y = f(x_1, x_2, \ldots, x_n)$ needs to be learned or optimized by relying on some sample data alone.

Definition 1 *Type-I opposite $\breve{x}_I$ of input x is defined as $\breve{x}_I = x_{\max} + x_{\min} - x$ where $x_{\max}$ is the maximum and $x_{\min}$ is the minimum value of x.*

Computation of type-I opposites are easier due to its linear definition in the variable space. On the contrary, type-II opposition scheme requires to operate on the output space.

Definition 2 *Type-II opposite $\breve{x}_{II}$ of input x is defined as $\breve{x}_{II} = \{x_i | \breve{y}(x_i) = y_{min} + y_{max} - y(x_i)\}$ where y_{max} is the maximum value, and y_{min} is the minimum value of y.*

Type-II opposites may be difficult to incorporate in real-world problems because **1)** their calculation may require a-priori domain knowledge, and **2)** the inverse of the function y, namely $\breve{y}$, is not available when we are dealing with unknown functions $y = f(x_1, x_2, \ldots, x_n)$. The focus of this paper is to develop a general framework to allow ANNs to learn the relationship among the inputs and their corresponding type-II opposites. Validation of the proposed approach is comprised of several benchmark functions. Validation results on the benchmark functions demonstrate the effectiveness of the proposed algorithm as an initial step for future developments in type-II OBL approximations using neural networks.

3. THE IDEA

Type-II (or true) opposite of x, denoted with $\breve{x}_{II}$, is more intuitive when compared to type-I opposite in context of "non-linear" functions. When looking at function $y = f(x_1, x_2, ..., x_n)$ in a typical machine-learning setup, one may receive the output values y for some input variables $x_1, x_2, \ldots, x_n$. However, the function $y = f(\cdot)$ itself is usually unknown otherwise there would be little justification for resorting to machine-learning tools. Instead, one has some sort of evaluation function $g(\cdot)$ (error, reward, fitness, etc.) that enables to assess the quality of any guess $\hat{x_1}, \hat{x_2}, \ldots, \hat{x_n}$ delivering an estimate $\hat{y}$ of the true/desired output y.

Tizhoosh and Rahnamayan introduced the idea of *opposition mining* as an approach to approximate type-II opposites for training the learning algorithms using fuzzy inference systems (FIS) with evolving rules in [3]. Evolving FIS has received much attention lately [20, 21] which is now being used for modeling nonlinear dynamic systems [22] and image classification and segmentation [23, 24, 25]. However, learning opposites with evolving rules are observed to be sensitive to the parameters used and encounters difficulty in generalizing a large variety of data. The proposed method in this paper uses opposition mining for training artificial neural network to approximate the relationship between the input x and its type-II opposite $\breve{x}_{II}$. This methodology can, of course, be extended for various applications; hence, if a large training dataset is available, then one can apply them at once instead of incremental changes. A graphical representation of the type-II opposites, described in Definition 2, is shown in Fig. 1: **1)** Given variable x, **2)** The corresponding $f(x)$ is calculated, **3)** from which the opposite of $f(x)$, namely $of(x)$ is determined, and **4)** the opposite(s) of x are found to be: $ox1, ox2$ and $ox3$. However, there are various challenges using this approach, which include: **1)** the output range of $y = f(\cdot)$ may not be a-priori known, thereby an updated knowledge on the

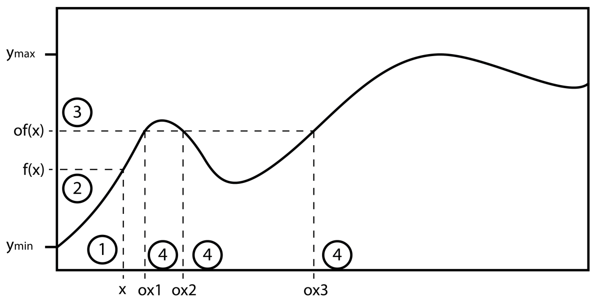

Fig. 1. Type-II opposites [Adopted from [3]].

output range of $[y_{min}, y_{max}]$ is needed, **2)** the precise output of the type-II opposite may not be present in the given data; thereby, a method to determine gaps in the dataset is needed to estimate the relationship between the input and the output, and **3)** it is difficult to generalize over a high dimension/range of data. In this paper, we put forward a concise algorithm to learn opposites via neural networks – which, as outlined in Section 5, is proven to yield better results when compared to recently introduced FIS approach.

4. LEARNING OPPOSITES

Neural networks with sufficient number of hidden layers can learn any bounded degree polynomials with good accuracy [26]. Therefore, ANNs make a good candidate for learning the nonlinear relationship between the inputs x and their type-II opposites $\breve{x}_{II}$.

In order to learn type-II opposites, we first need to sample the (quasi-)opposites from the given input-output data. The first stage of our algorithm is *opposition mining*, which provides the data that can be subsequently used by ANN for performing a nonlinear regression. One generally assumes that the more data is available the better the approximation becomes for $\breve{x}_{II} = f(x)$ as we have more (quasi-)opposites for training the ANN. We assume that the range of input variable is known, $x_i \in [x^i{}_{min}, x^i{}_{max}]$ but the range of output, $y_j \in [y^j{}_{min}, y^j{}_{max}]$, may be apriori unknown. Since we are approximating type-II opposites, we need to generate the (quasi-)opposite data from given training data. Our approach consists of two distinct stages:

Opposition Mining – The training data is sampled to establish the output boundaries. Depending on a specific oppositeness scheme, all data points in training data are processed to find (quasi-)opposites of each input. At the end of the opposite mining, we have a corresponding (quasi-)opposite (approximate of type-II opposite) for every input point in training data as outlined in Algorithm 1. There are different schemes for calculating the opposition. Given a sample of random variable $x \in [x_{min}, x_{max}]$ with mean $\bar{x}$, the opposite of x

Algorithm 1 Opposition Mining

Input: x inputs, y outputs & T_i opposition scheme
Output: type-II opposites $\breve{x}_{II}$ in same order as input x

```
 1: procedure OPPOSITION MINING($x, y, T_i$)
 2:     $y_{max} \leftarrow max(y)$
 3:     $y_{min} \leftarrow min(y)$
 4:     $\bar{y} \leftarrow mean(y)$
 5:     for $i \in [1, length(y)]$ do
 6:         $y_I \leftarrow y(i)$
                ▷ Choosing opposition scheme based on value of $T_i$
 7:         if $T_i = 0$ then
 8:             $oppY \leftarrow y_{max} + y_{min} - y_i$
 9:         else if $T_I = 1$ then
10:             $oppY \leftarrow \left(y_I + \frac{y_{max}+y_{min}}{2}\right) \% y_{max}$
11:         else
12:             $oppY \leftarrow 2\bar{y} - y_i$
13:         end if
                ▷ In case oppY goes out of boundary range
14:         if $oppY \notin [y_{max}, y_{min}]$ then
15:             $oppY \leftarrow y_{max} + y_{min} - y_i$
16:         end if
                ▷ Getting index of element in y closest to oppY
17:         $oppY_{idx} \leftarrow argmin(|y - oppY|)$
18:         $\breve{x}_{II}(i) \leftarrow x(oppY_{idx})$
19:     end for
        return $\breve{x}_{II}$
20: end procedure
```

can be calculated as follows:

$$T_1: \quad \breve{x}_I = z_{max} + x_{min} - x \tag{1}$$

$$T_2: \quad \breve{x}_I = \left(x + \frac{x_{min} + x_{max}}{2}\right) \% \quad x_{max} \tag{2}$$

$$T_3: \quad \breve{x}_I = 2\bar{x} - x \tag{3}$$

In scheme T_3, calculated opposite $\breve{x}_I$ may go out of the boundaries of the variable range. Therefore, for the experiments purposes, we solved the boundary violation problem by switching the scheme to T_1 whenever necessary (Algorithm 1, Line 14). It is important to note that, opposites calculated with any of the above schemes in output space when projected back on to the variable space are known as type-II (true) opposites.

Learning the Opposites – ANN is employed to approximate the function $\breve{x}_{II} = f(x)$ that maps input and its type-II opposite. The network is trained on the data collected from the opposition mining step, and can be retrained progressively as more data comes in or it can be used to predict the type-II opposites for a given input x. In the following sections, we report the results of some experiments to verify the accuracy of our algorithm, its superiority over existing FIS based technique and some discussion on the usefulness of type-II opposites for machine-learning algorithms.

5. EXPERIMENTS AND RESULTS

We have performed two experiment series to test the various aspects of our algorithm including – opposition mining, learning type-II using ANN and prediction accuracy of the trained ANN model versus evolving fuzzy rules model. The experiments deal with the approximation accuracy of type-II opposites and application of type-II opposites for some standard optimization scenarios respectively. For comparing the approximation accuracy, we used the 8 benchmark functions for generating the data required for the opposition-mining and subsequent training of ANN and evolving fuzzy rules models; which are taken from [3]. It is important to note that, benchmarks function have been intentionally kept simple and mostly monotonic in defined ranges, in order avoid the surjective relationship of inputs and their (quasi-)opposites during the opposition mining stage, thus allowing to extract the most feasible patterns in the data to be used with learning algorithms. However, in order to learn the type-II opposites across any general non-monotonic functions, it would be required to decompose the function in question into monotonic piece-wise ranges and subsequently perform the type-II approximation procedure on each of the piece separately. We calculated the approximation error of our algorithm for every benchmark function against different oppositeness schemes. We compared the results against the recently published ones by approximating type-II using evolving fuzzy rules [3].

5.1. Comparing with Evolving Fuzzy Rules

The results for 8 benchmark functions (used in [3]) are summarized in Table 1. Green cells represent the performance of ANN is statistically significant with 95% confidence (unless otherwise stated), whereas red cells show the significance of evolving fuzzy rules based approach for the respective opposition scheme. Cells marked gray represent the best results achieved using any deployed method or opposition scheme for a given benchmark function. The error in approximation of the type-II opposite $\breve{x}^*_{II}$ is inferred by comparing the value of the function at approximated opposite x_{II} and true opposite value of the function $\breve{y}^*_{II}$ at given input x. It is important to note that $\breve{y}^*_{II}$ can be calculated if input x, opposition scheme T_i and function f are known:

$$error(\breve{x}_{II}) \propto error(\breve{y}_{II}) = |\breve{y_{II}}^* - f(\breve{x}_{II})| \qquad (4)$$

The results are reported in Table 1. Overall T_1 seems to be the best oppositeness scheme. As long as y_{max}, y_{min} does not change for the given sample data, T_1 and T_2 schemes allow continuous training of ANN. The approximation seems to perform better overall except for the linear functions and functions with square root power. The ANN approach seems to generalize much better for the logarithm function at higher values of x where output changes much slower than change in x.

5.2. Optimization Problems

In this experiment, we test three standard optimization functions which are commonly used in literature of global optimization – Ackely, Bulkin and Booth functions [3].

Ackley Function – The Ackley function is given by

$$f(x_1, x_2) = 20\left(1 - \exp\left(-0.2\sqrt{0.5(x_1^2 + x_2^2)}\right)\right) - \exp\left(0.5 * (cos(2\pi x_1) + cos(2\pi x_2))\right) + \exp(1)$$

The global minimum is 0 at $(3, 0.5)$ with $x_1, x_2 \in [-35, 35]$.

Bulkin Function – The Bulkin function is given by

$$f(x_1, x_2) = 100\sqrt{||x_2 - 0.01x_1^2||} + 0.01||x_1 + 10||$$

The global minimum is 0 at $(-10, 0)$ with $x_1 \in [-15, -5]$ and $x_2 \in [-3, 3]$.

Booth Function – The Booth function is given by

$$f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

The global minimum is 0 at $(1, 3)$ such that $x_1, X_2 \in [-10, 10]$.

We train ANN and evolving fuzzy rules method for type-II opposites using $n_s = 1000$ samples from each of the benchmark function. This experiment enables to verify three majors points: **1.)** test whether the fundamental statement of OBL holds, namely that – *simultaneous consideration of a guess and opposite guess provides faster convergence to the solution in learning and optimization processes*, **2.)** test whether type-II opposites provides any advantage over type-I, and **3.)** test if ANN based type-II approximation provides any superiority over recently introduced evolving fuzzy rule approach.

To conduct this experiment, we generate two random input samples x_1^r and x_2^r and we calculate the error (distance from the global minimum). Then, we approximate the opposites $\breve{x}_1^r$ and $\breve{x}_2^r$ and calculate the error again. In order to find the solution, we chose the strategy that yields less error and continues the process. We should expect to have a reduction or no change in error at every iteration since we deliberately choose the outcome with the least error. We carry out the experiment for both type-I $\breve{x}_I^r$ and type-II opposites $\breve{x}_{II}^r$ generated by each of the candidate methods. By recording the average error after $0.1 * n_s$ iterations for multiple runs of the experiments, we can test whether considering type-II opposite from either of the candidate methods have any statistical significance over type-I in yielding outcome closer to the global minimum. However, the focus is more on comparing the two candidate approaches for approximating type-II (proposed and evolving fuzzy rule-based), to see which one provides better estimates of type-II to bring the optimization process closer to the global optima. The results for Ackley, Booth, and Bulkin functions are shown in Table 2. The

Table 1. Comparison between error in approximation of type-II opposite using trained ANN mode versus evolving fuzzy rules.

Benchmark Function	Opposition Scheme	Proposed Approach ($\mu \pm \sigma$)	Fuzzy based Approach ($\mu \pm \sigma$)	p-value	Welch's Test Results
$f(x) = (2x+8)^3$	T_1	0.76 ± 0.85	4.41 ± 2.44	0.0005	Proposed approach
	T_2	9.53 ± 12.99	11.96 ± 12.82	0.3398	Not significant
	T_3	4.65 ± 9.95	6.82 ± 10.62	0.3214	Not significant
$f(x) = log(x+3)$	T_1	18.95 ± 18.00	30.05 ± 20.87	0.11	Proposed approach $\approx$ 80% confidence
	T_2	10.02 ± 15.73	11.20 ± 18.48	0.4398	Not significant
	T_3	2.98 ± 9.85	6.39 ± 7.87	0.2020	Not significant
$f(x) = 2 * x$	T_1	0.19 ± 0.26	0.01 ± 0.01	0.9693	Fuzzy approach
	T_2	6.24 ± 11.08	21.03 ± 14.31	0.0091	Proposed approach
	T_3	0.30 ± 0.58	0.25 ± 0.65	5759	Not significant
$f(x) = x^2$	T_1	0.49 ± 0.55	3.04 ± 1.72	0.0005	Proposed approach
	T_2	8.46 ± 12.56	15.02 ± 14.31	0.1456	Not significant
	T_3	3.61 ± 7.98	4.41 ± 6.38	0.4039	Not significant
$f(x) = \sqrt{x}$	T_1	0.60 ± 1.15	0.04 ± 0.13	0.9188	Fuzzy approach
	T_2	2.91 ± 7.68	18.99 ± 16.33	0.0074	Proposed approach
	T_3	2.70 ± 5.91	3.74 ± 4.22	0.3287	Not significant
$f(x) = x^{3/2}$	T_1	0.37 ± 0.31	1.68 ± 1.03	0.0014	Proposed approach
	T_2	5.16 ± 10.52	17.89 ± 14.92	0.0212	Proposed approach
	T_3	2.15 ± 4.27	2.66 ± 4.12	0.4022	Not significant
$f(x) = x^3 + x^2 + 1$	T_1	1.36 ± 2.84	4.42 ± 2.72	0.0122	Proposed approach
	T_2	9.84 ± 12.57	11.82 ± 12.95	0.3656	Not significant
	T_3	5.13 ± 10.79	6.31 ± 9.99	0.4022	Not significant
$f(x) = \frac{\sqrt{x+1}}{3}$	T_1	1.63 ± 3.50	0.06 ± 0.11	0.9061	Fuzzy approach
	T_2	4.27 ± 8.17	18.20 ± 16.79	0.0173	Proposed approach
	T_3	2.11 ± 5.15	3.74 ± 4.61	0.2332	Not significant

Table 2. Errors values for Ackely, Booth and Bulkin test functions.

Runs	x_1^r, x_2^r	$x_{1,II,ANN}^r, x_{2,II,ANN}^r$	$x_{1,II,FIS}^r, x_{2,II,FIS}^r$	$x_{1,I}^r, x_{2,I}^r$
Ackley Function				
1. run	487.70 ± 516.362	108.00 ± 139.03	117.43 ± 150.53	198.91 ± 222.677
2. run	377.43 ± 455.85	108.65 ± 136.91	116.62 ± 138.02	155.24 ± 187.24
3. run	512.31 ± 529.517	156.66 ± 162.61	143.05 ± 153.73	238.00 ± 246.19
4. run	374.87 ± 388.63	124.20 ± 124.50	123.83 ± 134.19	154.55 ± 139.38
5. run	415.84 ± 465.70	146.94 ± 157.14	128.90 ± 145.57	175.28 ± 218.05
Booth Function				
1. run	424.53 ± 502.66	302.66 ± 274.64	337.39 ± 304.37	183.97 ± 197.39
2. run	420.03 ± 443.28	303.23 ± 318.19	330.74 ± 328.68	191.13 ± 166.01
3. run	391.28 ± 445.53	318.19 ± 276.40	328.68 ± 280.56	166.01 ± 195.55
4. run	338.94 ± 406.02	292.60 ± 290.77	123.83 ± 295.41	154.55 ± 174.13
5. run	430.72 ± 496.36	323.67 ± 286.38	338.95 ± 289.79	187.19 ± 211.87
Bulkin Function				
1. run	120.40 ± 41.00	47.26 ± 18.99	63.89 ± 24.04	101.21 ± 38.46
2. run	119.20 ± 50.42	53.78 ± 15.48	68.58 ± 26.75	97.11 ± 42.71
3. run	126.78 ± 45.16	43.09 ± 17.24	72.83 ± 29.55	103.427 ± 38.58
4. run	118.39 ± 47.31	49.17 ± 20.12	64.02 ± 30.43	94.27 ± 39.96
5. run	128.58 ± 44.49	48.23 ± 20.96	66.43 ± 30.63	100.40 ± 38.16

first column consists of random guesses, the second column contains type-II opposite guesses estimated by ANN, and the third column contains type-II opposite guesses using the FIS approach; the last column is type-I opposites. By performing the t-test, results of type-II opposites obtained by ANN are statistically significant compared to the FIS approach. Type-II is performing better except for the Booth function.

6. CONCLUSION

Ten years since the introduction of opposition-based learning, the full potential of type-II opposites is still largely unknown. In this paper, we put forward a method for learning type-II opposites with ANNs. The core idea in this paper is to utilize the (quasi-)opposite data collected from opposition-mining to learn the relationship between input x and its type-II opposite $\breve{x}_{II}$ using neural networks. We tested the proposed algorithms with various benchmark functions and compared it against the existing fuzzy rules-based approach. We showed the correctness of fundamental statement of OBL scheme by utilizing type-II opposites on three of the famous global optimization problems. One of the major hurdles for existing type-II approximation methods (including proposed in this paper) is when the function in question is highly non-monotonic or periodic in nature. In those circumstances, the relationship between x and $\breve{x}_{II}$ becomes surjective, causing discontinuities in opposition mining. This makes it difficult for any learning algorithm difficult to fit such discontinuous data. There is a potential for improvement in future works where non-monotonic functions can be decomposed into monotonic piece-wise intervals; each of the intervals can then be trained separately.

7. REFERENCES

[1] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *null*. IEEE, 2005, pp. 695–701.

[2] S. Rahnamayan, H. R. Tizhoosh, and M. Salama, "Opposition-based differential evolution," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 1, pp. 64–79, 2008.

[3] H. R. Tizhoosh and S. Rahnamayan, "Learning opposites with evolving rules," in *Fuzzy Systems (FUZZ-*

IEEE), 2015 IEEE International Conference on. IEEE, 2015, pp. 1–8.

[4] F. S. Al-Qunaieer, H. R. Tizhoosh, and S. Rahnamayan, "Opposition based computinga survey," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010, pp. 1–7.

[5] H. R. Tizhoosh, "Reinforcement learning based on actions and opposite actions," in *Int. Conf. on Artificial Intelligence and Machine Learning*, 2005, p. 9498.

[6] M. Mahootchi, H. Tizhoosh, and K. Ponnambalam, "Opposition-based reinforcement learning in the management of water resources," in *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*. IEEE, 2007, pp. 217–224.

[7] H. R. Tizhoosh, "Opposition-based reinforcement learning," *JACIII*, vol. 10, no. 4, pp. 578–585, 2006.

[8] M. Mahootchi, H. R. Tizhoosh, and K. Ponnambalam, "Oppositional extension of reinforcement learning techniques," *Information Sciences*, vol. 275, pp. 101–114, 2014.

[9] S. Rahnamayan, H. R. Tizhoosh, and M. Salama, "Opposition-based differential evolution algorithms," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, 2006, pp. 2010–2017.

[10] H. Salehinejad, S. Rahnamayan, and H. R. Tizhoosh, "Type-ii opposition-based differential evolution," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 1768–1775.

[11] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "A novel population initialization method for accelerating evolutionary algorithms," *Computers & Mathematics with Applications*, vol. 53(10), pp. 1605–1614, 2007.

[12] S. Rahnamayan and H. R. Tizhoosh, "Image thresholding using micro opposition-based differential evolution (micro-ode)," in *IEEE World Congress on Computational Intelligence Evolutionary Computation*, 2008, pp. 1409–1416.

[13] C. Zhang, Z. Ni, Z. Wu, and L. Gu, "A novel swarm model with quasi-oppositional particle," in *Information Technology and Applications, 2009. IFITA'09. International Forum on*, vol. 1, 2009, pp. 325–330.

[14] H. Jabeen, Z. Jalil, and A. R. Baig, "Opposition based initialization in particle swarm optimization (o-pso)," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, 2009, pp. 2047–2052.

[15] M. Kaucic, "A multi-start opposition-based particle swarm optimization algorithm with adaptive velocity for bound constrained global optimization," *Journal of Global Optimization*, vol. 55(1), pp. 165–188, 2013.

[16] M. Rashid and A. R. Baig, "Improved opposition-based pso for feedforward neural network training," in *Information Science and Applications (ICISA), International Conference on*, 2010, pp. 1–6.

[17] M. Yaghini, M. M. Khoshraftar, and M. Fallahi, "Hiopga: a new hybrid metaheuristic algorithm to train feedforward neural networks for prediction," in *Proceedings of the International Conference on Data Mining*, 2011, pp. 18–21.

[18] Q. Xu, L. Wang, N. Wang, X. Hei, and L. Zhao, "A review of opposition-based learning from 2005 to 2012," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 1–12, 2014.

[19] M. Ventresca and H. R. Tizhoosh, "A diversity maintaining population-based incremental learning algorithm," *Information Sciences*, vol. 178(21), pp. 4038–4056, 2008.

[20] P. Angelov, E. Lughofer, and X. Zhou, "Evolving fuzzy classifiers using different model architectures," *Fuzzy Sets and Systems*, vol. 159(23), pp. 3160–3182, 2008.

[21] P. P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *Fuzzy Systems, IEEE Transactions on*, vol. 16, no. 6, pp. 1462–1475, 2008.

[22] J.-C. de Barros and A. L. Dexter, "On-line identification of computationally undemanding evolving fuzzy models," *Fuzzy sets and systems*, vol. 158(18), pp. 1997–2012, 2007.

[23] E. Lughofer, "On-line evolving image classifiers and their application to surface inspection," *Image and Vision Computing*, vol. 28(7), pp. 1065–1079, 2010.

[24] A. A. Othman and H. R. Tizhoosh, "Evolving fuzzy image segmentation," in *Fuzzy Systems, IEEE International Conference on*, 2011, pp. 1603–1609.

[25] A. A. Othman, H. R. Tizhoosh, and F. Khalvati, "Efisevolving fuzzy image segmentation," *Fuzzy Systems, IEEE Transactions on*, vol. 22(1), pp. 72–82, 2014.

[26] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang, "Learning polynomials with neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1908–1916.