Learning Opposites with Evolving Rules

H.R. Tizhoosh
Centre for Pattern Analysis and Machine Intelligence
University of Waterloo
Waterloo, Ontario, Canada
Web: http://tizhoosh.uwaterloo.ca/

S. Rahnamayan
Electrical, Computer and Software Engineering
University of Ontario Institute of Technology
Oshawa, Ontario, Canada
Email: shahryar.rahnamayan@uoit.ca

Abstract—The idea of opposition-based learning was introduced 10 years ago. Since then a noteworthy group of researchers has used some notions of oppositeness to improve existing optimization and learning algorithms. Among others, evolutionary algorithms, reinforcement agents, and neural networks have been reportedly extended into their "opposition-based" version to become faster and/or more accurate. However, most works still use a simple notion of opposites, namely linear (or type-I) opposition, that for each $x \in [a, b]$ assigns its opposite as $\breve{x}_I = a + b - x$. This, of course, is a very naive estimate of the actual or true (non-linear) opposite $\breve{x}_{II}$, which has been called type-II opposite in literature. In absence of any knowledge about a function $y = f(\mathbf{x})$ that we need to approximate, there seems to be no alternative to the naivety of type-I opposition if one intents to utilize oppositional concepts. But the question is if we can receive some level of accuracy increase and time savings by using the naive opposite estimate $\breve{x}_I$ according to all reports in literature, what would we be able to gain, in terms of even higher accuracies and more reduction in computational complexity, if we would generate and employ true opposites? This work introduces an approach to approximate type-II opposites using evolving fuzzy rules when we first perform "opposition mining". We show with multiple examples that learning true opposites is possible when we mine the opposites from the training data to subsequently approximate $\breve{x}_{II} = f(\mathbf{x}, y)$.

I. What is the Problem?

It was 10 years ago that "**opposition-based learning**" (OBL) was born. Since then, a modest but growing community of researchers has tried to use OBL to improve diverse optimization and learning techniques. Evolutionary algorithms, reinforcement agents, swarm-based methods, and neural networks, to mention a few, have been extended to employ oppositeness in their processing. Evaluating the error of weights and opposite weights, examining the rewards for actions and counter-actions, and examining the fitness of chromosomes and anti-chromosomes are examples for rethinking existing concepts by embedding opposite entities. Apparently, as a review of literature easily illustrates, using opposites can actually help accelerating many optimization and learning processes (see section III). The main benefit of using opposites seems to be contributing to an accelerated convergence. Apparently by simultaneous consideration of entities and opposite entities, algorithms become capable of jumping over large portions of the solution landscape for difficult problems when they do not exhibit any significance.

Since the first paper was published on OBL in 2005 [32], it has not been really clear how oppositeness can actually be captured with respect to the intrinsic behaviour of the problem at hand. The proposed scheme was that for any given $x \in [a, b]$ the opposite $\breve{x}$ can be given via $\breve{x} = a + b - x$. This, if at all, only makes sense for "linear" functions, in which case no particular algorithmic sophistication is required to begin with. But in spite of its simplicity, many works have reported benefits for using $\breve{x} = a + b - x$, which is called "type I opposite" in context of opposition-based computing [35].

After 10 years, some questions are still unanswered: Is there such thing as type II opposites? If they exist, how can we calculate them? And most importantly, would using type II opposites bring any benefit to existing machine-learning and optimization methods?

II. The Idea

What we call type II (or true) opposite of x, denoted with $\breve{x}_{II}$, is supposed to be meaningful for "non-linear" mappings and relationships, in contrast to type I opposites $\breve{x}^I$ that latently assume a linear relationship between in- and outputs.

Looking at the function $y = f(x_1, x_2, \ldots, x_n)$ in a typical machine-learning or optimization context, one is generally fortunate to receive the output values y for some input variables $x_1, x_2, \ldots, x_n$. However, the function $y = f(\cdot)$ itself is generally unknown otherwise there would be little justification for resorting to machine intelligence. Instead, one has some sort of evaluation function $g(\cdot)$ (error, reward, fitness etc.) that enables us 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.

The **idea** proposed in this paper is to use training data, whenever available, to perform **opposition mining** in order to approximate type II opposites gradually by learning the $x, y, \breve{x}_{II}$-relationship, or more precisely by learning $\breve{x}_{II} = f(x, y)$. Of course, if a large number of training data is available, then one could also just apply them *at once* instead of perpetual/continuous change.

For every input x, in contrast to the type I opposites which are defined according to

$$\breve{x}_I = a + b - x, \qquad (1)$$

we propose to use the more meaningful type II opposites according to

$$\breve{x}_{II} = \{x_i | f(x_i) = y_{\min} + y_{\max} - f(x)\}. \qquad (2)$$

There are several challenges for such an approach: **1)** the range of the output $y = f(\cdot)$ may not be a-priori known, hence we may need to continuously (in an evolving manner) update our knowledge of the output range $[y_{\min}, y_{\max}]$, **2)** the exact output for a type-II opposite may not be present in data. Hence,

the approach should be capable of improving its estimation as data stream provides more clues about the gaps in our estimate of the in- and output relationship, and **3)** sufficient (representative) data may not be available at any given time; another reason in favour of an evolving approach.

In this work, we put forward a concise algorithm to learn opposites via fuzzy inference systems (FIS) with or without evolving rules depending on application whereas the evolving fuzzy systems may offer more versatility in many challenging applications (section IV). Before that, we briefly review the existing literature in section III. Experiments and results, to verify the correctness and the usefulness of the proposed approach are described in section V.

III. REVIEWING PREVIOUS WORKS

In this section, we briefly review the existing literature on opposition-based learning and the evolving fuzzy systems.

A. Opposition-Based Learning

Opposition-based learning was introduced in 2005 [32]. Motivated by finding an alternative for random initialization, it was observed that for many algorithms we start with a random guess and hope to move toward an existing solution in a speedy fashion. Examples were named such as the weights of a neural network, the population in genetic algorithms, and the action policy of reinforcement agents. Further, the paper stated that if we begin with a random guess far away from the existing solution, "let say in worst case it is in the opposite location, then the approximation, search or optimization will take considerably more time, or in worst case becomes intractable". The paper advocated that "we should be looking in all directions simultaneously, or more concretely, in the opposite direction" in order to have a higher chance of finding a solution in a shorter time, and established the following definition [32]:

Opposition-Based Learning – Let $f(x)$ be the function in focus and $g(\cdot)$ a proper evaluation function. If $x \in [a, b]$ is an initial (random) guess and $\breve{x}$ is its opposite value, then in every iteration we calculate $f(x)$ and $f(\breve{x})$. The learning continues with x if $g(f(x)) > g(f(\breve{x}))$, otherwise with $\breve{x}$. The evaluation function $g(\cdot)$, as a measure of optimality, compares the suitability of results (e.g. fitness function, reward and punishment, error function).

Of course, after many reports in the past 10 years we know that x should not necessarily be a random guess at the beginning, but it can be a calculated or estimated value by a learning or optimization algorithm at any stage of the process. As well, we may not necessarily need to evaluate $g(f(x))$ and $g(f(\breve{x}))$ in all iterations.

Rahnamayan et al. introduced opposition-based differential evolution (ODE) which seems to be one of the most successful applications of OBL via integration within an existing algorithm [19]-[22]. Among other early works, there are papers on the application of OBL for reinforcement learning [33], [34] where "opposite actions" were defined for specific problems, hence, not engaging in question of type-I versus type-II opposites. Defining opposite fuzzy sets, which may be mistaken with "antonyms" and their applications have been introduced

too [8], [36], [37]. Tizhoosh and Ventresca [35] presented a complete collection of oppositional concepts in machine intelligence in an edited volume, in which the term "type-II opposition" was coined in the chapter *opposition-based computing*. Ventresca and Tizhoosh also extended simulated annealing [38] and introduced the notion of opposite transfer functions for neural networks [39].

Surveys are available to provide an overview of OBL methods. For instance, Al-Qunaieer et al. [7] provide a compact survey of oppositional methods. The most comprehensive survey of opposition-based methods so far has been published by Xu et al. [40].

Type II opposites have not been examined in-depth. However, as we will briefly review them in the following two paragraphs, recently two papers have independently proposed different possibilities for estimating type II opposites.

Mahootchi et al. [15] provided some new ideas for type-II opposites when dealing with reinforcement learning methods such as Q- and *sarsa* learning. Looking at actions a that an agent could take to manipulate its environment, the type-I opposite action $\breve{a}$ was originally defined as 'directional' opposite for grid-world agents, hence "up" was the opposite of "down", for instance. However, in a stochastic environment it makes sense to examine the discounted accumulated rewards $Q(s, a)$ for each state s and estimate the opposite as follows:

$$\breve{a} \in \left\{ \hat{a} | Q^t(i, \hat{a}) \approx \max_b Q^t(i, b) + \min_b Q^t(i, b) - Q^t(i, a) \right\} \tag{3}$$

To obtain new knowledge about the opposites from observations, the authors train an MLP network which is updated every several episodes. For higher accuracy, the approximate function could be updated online when the agent can interact with the stochastic environment. "In other words, after the first training of the network, two sets of action-value functions exist, with the first being used to find the action that the agent should take and the approximate one being used to extract the opposites. This means that there is a mutual relationship between the agent and the opposite agent. This process may assist each agent to accomplish its tasks more efficiently in terms of convergence time and accuracy in real-world applications" [15]. The authors also reported that MLP may not be efficient for large applications due to its computational expense. They suggest that one may substitute MLP with other types of function approximations such as fuzzy inference systems.

Salehinejad et al. [30], examining a type-II extension of opposition-based differential evolution (ODE) also point to the fact that the true opposites should be calculated via $\breve{f}(x) = y_{\min} + y_{\max} - f(x)$ to focus on the output but they use centroid-based method instead of depending on $\min$ and $\max$ specially where the landscape boundaries are unknown. A look-up table can help to find $\breve{f}(x)$, and if not present in the table, its value can be estimated via interpolation. In their algorithm, they use type-I and type-II opposition simultaneously.

B. Evolving Fuzzy Systems

An *evolving fuzzy system* has been initially introduced as an unsupervised method of updating the rule-based structure of a fuzzy inference system (FIS) in a non-iterative way [1],

[2]. The main idea is that FIS can be extended as more data becomes available. The fuzzy rule base evolves (is extended) by adding more rules to a basic rule set formed by the initial fuzzy model or by replacing existing rules with ones that approximate the data better. When data are captured online, its potential is calculated, and the potentials of existing cluster centers are recursively updated. The potential of the new data is compared with the existing centers and one of the following decisions is made: 1) Replace one or more existing clusters with the new data point if its potential is higher than a certain threshold and the new data point is close to an old center, 2) Add the new data as a new cluster if its potential is higher than a certain threshold.

The term "evolving" may imply that instead of retraining the initial system, the system should be updated online when new data become available. However, the evolution of rules in a fuzzy inference system can occur in a variety of ways, and as long as the initial fuzzy model (rule set) is continuously updated, no preference exists with respect to how the actual update is performed. In [3], the Takagi–Sugeno (T-S) type of fuzzy controller with an evolving structure was proposed for online identification of T-S systems. The fuzzy rules of this controller are expanded through the use of data collected during the control process, and is, thus, trained in a non-iterative (recursive) way. There have been approaches that use an iterative approach specially for the application where keeping and re-training the data does not pose a challenge [16]. More recent approaches to evolving fuzzy systems have been proposed [4],[5]. Evolving rules have been used for modelling nonlinear dynamic systems [6] and image classification and segmentation [10][16][17].

Comprehensive treatment and reports on new advances in evolving fuzzy systems can be found in literature [11], [12], [18], [13]. There are other learning approaches one could use. For instance, participatory learning [41] where compatibility between observations and beliefs plays an important role.

In this paper, we propose a general, simple and efficient approach to learn type-II opposites using fuzzy inference systems. The emphasis is on **opposition mining** (finding opposites or quasi-opposites in the available training data) and **evolving rules** (which perpetually adjust the approximation of the mined opposites).

IV. LEARNING OPPOSITES VIA EVOLVING RULES

A fuzzy inference system (FIS) generally consists of a set of IF–THEN rules of the following form:

IF x_1 is A_1 AND x_2 is A_2 AND $\cdots$ AND x_n is A_n THEN y is B

where $x_i, y \in X$ are variables defined in corresponding universes of discourse X_i and Y, respectively, and A_i and B are fuzzy (sub)sets (in this work we investigate the problems with many inputs and one output only). These rules can be defined by experts. However, in most real-world applications they can be extracted from available data via clustering. A Takagi-Sugeno, or T–S fuzzy inference system operates with fuzzy rules of the following general form [31]:

IF x_1 is A_1 AND x_2 is A_2 AND $\cdots$ AND x_n is A_n THEN $y = f_j(x_1, x_2, \ldots, x_n), j = 1, 2, \cdots, N$

where x_i and y are variables defined in corresponding universes of discourse X_i and Y, respectively, and A_i is a fuzzy (sub)set. The function $f_j(x_1, x_2, \cdots, x_n)$ is a crisp (nonfuzzy) function of xi. In general, the function f_j is defined as the weighted combination of all variables $f_j(x_1, x_2, \cdots, x_n) = w - 0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$.

The output is then calculated by

$$y = \frac{\sum_{j=1}^{N} f_j(x_1, x_2, \cdots, x_n) \mathcal{T}_{i=1}^{m_j} \mu_j(x_i)}{\sum_{j=1}^{N} \mathcal{T}_{i=1}^{m_j} \mu_j(x_i)} \tag{4}$$

where N is the number of fuzzy rules, n is the number of inputs (features), μ_j is the membership value of the ith input x_i for the jth rule, $1 \leq m_j \leq n$, and $\mathcal{T}$ is a T-norm representing the logical conjunction.

In order to learn the type II (true) opposites with evolving fuzzy rules, we first have to sample the problem at hand to find existing (quasi-)opposites. This is the **opposition mining** stage that provides the data for clustering and the subsequent rule extraction. One generally assumes that the more data is available the better the approximation becomes because we expect that more data increases the probability for opposition mining to find more representative samples for rule generation in order to more accurately approximate $\breve{x}_{II} = f(x, y)$. We assume that the range of input variables is known, $x_i \in [x_{\min}^i, x_{\max}^i]$, but the range of output, $y_j \in [y_{\min}^j, y_{\max}^j]$, may be a-priori unknown. Since we are looking for the true (type-II) opposites, we need to calculate the (quasi-)opposite of the output for each sample. Subsequently, we find the closest value to that point and select its corresponding input as the opposite of the given input. If the inflow of data continues, the reliability of the mined opposites as training data increases which would result in higher accuracies for the approximated true opposites.

As preparation for designing the learning procedure, let us look at Figure 1:

- Calculating type I opposites is straightforward (top diagram). For any given x we calculate its opposite $\breve{x} = a + b - x$ (denoted with ox on the x-axis) which can then be used to estimate the value of $f(\breve{x})$ (denoted with f(ox) on the y-axis). However, as it is apparent from the diagram, calculation of ox is completely detached from the output.

- Type II opposites, in contrast, are based on a more realistic (or more intuitive) understanding of opposite-ness (bottom diagram in Figure 1). For any given x (①), we first need an estimate or evaluation of $f(x)$ (②). Then, we find the opposite of $f(x)$, namely $\breve{f}(x)$ or of(x) in Figure 1 (③). Any input that can produce outputs like of(x) is the type II opposite of x (ox1, ox2 and ox3 on the x-axis) (④). Of course, for non-monotonic functions, we may get multiple opposites for each input. In most cases, one of those opposites might suffice to exploit the potential benefits of OBL when validated with the available objective function.

Algorithm 1 describes the steps how we learn the opposites based on the same steps in the bottom diagram of Figure 1. The approach consists of two distinct stages:
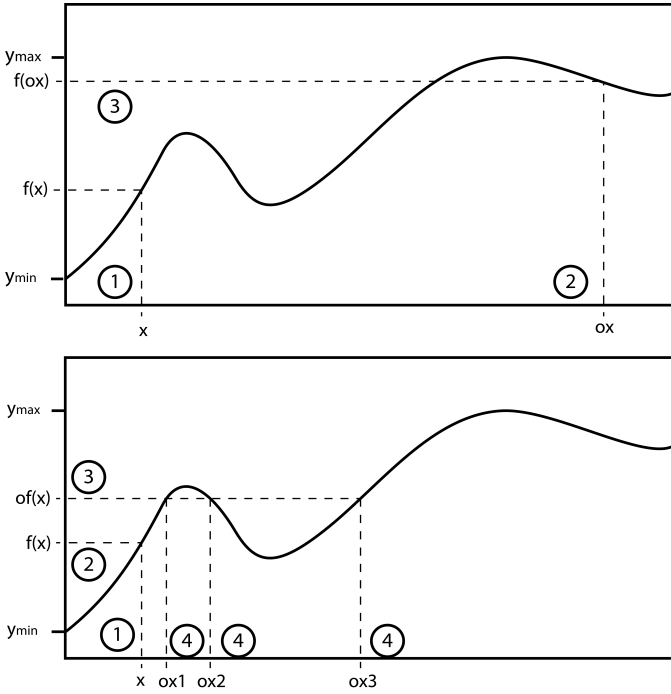
Fig. 1. Type I versus Type II Opposition. TOP: 1) variable x is given, 2) opposite of x, xo, is calculated via a+b-x . BOTTOM: 1) variable x is given, 2) f(x) is calculated, 3) opposite of f(x), namely of(x) is calculated, 4) opposite(s) of x are found: ox1,ox2 and ox3.

Opposition Mining (Lines 7 to 24) – The training data is sampled to established the output boundaries. Depending on a specific oppositeness scheme, all data points are then processed to find the (quasi-)opposite of each input and the corresponding input $\breve{x}_{II}$, which is the type-II opposite. At the end of the opposition mining we have n_s opposites.

We also look at different ways of calculating opposition in general. Given any function $y = f(x)$ with variables $x \in [x_{\min}, x_{\max}]$ with a sample average of $\bar{x}$, one may calculate oppositeness for the input domain in different ways [32], [35], [25], [24] (% denotes the modulo operator):

$$T_1 : \quad \breve{x}_I = x_{\min} + x_{\max} - x \quad (5)$$

$$T_2 : \quad \breve{x}_I = \left(x + \frac{x_{\min} + x_{\max}}{2}\right) \% x_{\max} \quad (6)$$

$$T_3 : \quad \breve{x}_I = 2\bar{x} - x \quad (7)$$

Analogously, given the output $y \in [y_{\min}, y_{\max}]$ with an average $\bar{y}$, one may calculate its opposite $\breve{y}_I$ in different ways:

$$T_1 : \quad \breve{y} = y_{\min} + y_{\max} - y \quad (8)$$

$$T_2 : \quad \breve{y} = \left(y + \frac{y_{\min} + y_{\max}}{2}\right) \% y_{\max} \quad (9)$$

$$T_3 : \quad \breve{y} = 2\bar{y} - y \quad (10)$$

For T_3 scheme, we may go out of the existing boundary of the variables sometimes. For the experiments, we solved this problem by switching to scheme T_1 whenever necessary. It is paramount to emphasize that any opposite calculate based on $\breve{y}$ is a type-II opposite.

Learning the Opposites (Lines 26 to 30) – Here any learning and approximation method may be used to employ the results of opposition mining. As for our approach, namely the fuzzy inference systems (FIS), in- and outputs are clustered to extract rules. Performing fuzzy inference with these rules will then approximate the type-II opposites for new (unseen) inputs.

In following section, we report the results of some experiments to verify the superiority of type II over type I opposites.

Algorithm 1 Learning the opposites via Evolving Rules

1: ————— Initialization —————
2: Set FIS mechanism, e.g. Takagi-Sugeno
3: Determine the number of samples n_s
4: Set the number of clusters n_c
5: Select oppositeness scheme $T_i, i \in \{1,2,3\}$
6: ————— **Opposition Mining** —————
7: Get sample points $< x_1^i, x_2^i, \ldots, y^i > (i = 1, 2, \ldots, n_s)$
8: Calculate $y_{\min}$, $y_{\max}$, and $\bar{y}$
9: **for** $i = 1 : n_s$ **do**
10:    $thisY \leftarrow y(i)$;
11:    Use T_i to estimate the opposite of the ouput:
12:      $T_i = T_1: oppY \leftarrow y_{\min} + y_{\max} - thisY$
13:      $T_i = T_2: oppY \leftarrow \left(thisY + \frac{(y_{\min} + y_{\max})}{2}\right) \% y_{\max}$
14:      $T_i = T_3: oppY \leftarrow 2\bar{y} - thisY$
15:    $minDiff \leftarrow \infty$ (a large number)
16:    **for** $j = 1 : n_s$ **do**
17:      $thatY \leftarrow y(j)$
18:      $thisDiff \leftarrow |oppY - thatY|$
19:      **if** $thisDiff < minDiff$ **then**
20:        $minDiff \leftarrow thisDiff$
21:        $\breve{x}_{II}(i) \leftarrow x(j)$
22:      **end if**
23:    **end for**
24: **end for**
25: ———— Train FIS ————
26: Set $< x_1^i, x_2^i, \ldots, y^i >$ as input
27: Set $< \breve{x}_{1,II}^i, \breve{x}_{2,II}^i, \ldots >$ as output
28: Partition the data into n_c clusters
29: Extract fuzzy rules
30: Save the FIS data

V. EXPERIMENTS AND RESULTS

We conduct multiple experiments to demonstrate the usefulness of opposition mining and learning opposites with evolving fuzzy rules. We examine 9 simple functions with known inverse functions to verify the correctness of the algorithm, and to establish a better understanding of the approach. Using functions with known inverse relations is a very reliable testing vehicle because the true opposites can be easily calculated for validation. As well, we employed Matlab™ "**genfis3**" function to implement a clustering-based Takagi-Sugeno fuzzy inference system (note: Matlab™ "**genfis3**" was trained incrementally to achieve the rule evolution). We experimented with number of clusters n_c to be 30 or 60 clusters (with relatively similar results) and set the number of maximum iterations to maxIter= 5000. The fuzzy exponent was set to $m = 2$ and the clustering error threshold was fixed at $\epsilon = 0.00001$.

A. First Experiment Series

The results for 9 functions are summarized in Table I (also see Figure 2). The simplicity of the functions enable us to easily formulate the inverse function such that we can exactly calculate the **true opposites**. With exception of $f_7(x)$, the inverse of all other test functions are given in Table I:

$$f_7^{-1}(x) = \cfrac{1}{9\left(\frac{y}{2} + \left(\left(\frac{y}{2} - \frac{29}{54}\right)^2 - \frac{1}{729}\right)^{\frac{1}{2}} - \frac{29}{54}\right)^{\frac{1}{3}}} \quad (11)$$
$$+ \left(\frac{y}{2} + \sqrt{\left(\left(\frac{y}{2} - \frac{29}{54}\right)^2 - \frac{1}{729}\right)} - \frac{29}{54}\right)^{\frac{1}{3}} - \frac{1}{3}$$

For every function 100 random samples were drawn in the opposition mining stage. The error for $\breve{x}_I$ and $\breve{x}_{II}$ is calculated through direct comparison with the true opposite $\breve{x}^*$ available through inverse functions: $\text{error}(\breve{x}_I) = |\breve{x}^* - \breve{x}_I|$ and $\text{error}(\breve{x}_{II}) = |\breve{x}^* - \breve{x}_{II}|$.

Analysis of the Results –

- Type-II opposite is clearly more accurate than type-I opposite with exception of linear function $f_3(x)$.

- The opposition scheme T_1 seems to be the best representation of oppositeness. T_2 is the best oppositeness scheme only for $f_8(x)$.

- If we only examine type I opposites, T_3 is better than T_1 and T_2 for 7 out of 9 test functions.
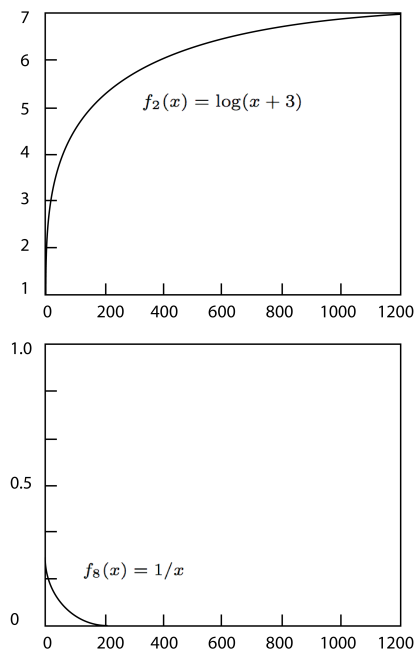

Fig. 2. Examples for the test functions: $f_2(x)$ (top) and $f_8(x)$ (bottom) according to equations provided in Table I. The simplicity of the functions allows us to calculate the inverse (hence the true opposites) but they are still difficult enough to demonstrate the limitations of the type-I opposition as large errors in Table I demonstrate.

B. The Second Experiments Series

In the second experiment series, we looked at the effect of evolving rules: What happens if data is fed into the system one-by-one or block-wise? Intuitively, one expects that through clustering of more data, a more compact rule base can be generated which can more accurately approximate the true opposites.

As Figure 3 shows some representative examples, the average error and standard deviation of approximated opposites decrease over time as the estimated values by evolving rules become more and more representative of the true opposites. We also observed a typical *pinnacle point* at which the error reaches its maximum when one observes the evolving opposites long enough (Figure 4). This appear to be due to the intrinsic nature of evolving steps as at the beginning there are not enough data points to extract *good* rules, thus resulting in large errors.

C. Third Experiment Series

In this section, we test 3 standard optimization functions which are commonly used in the literature of global optimization. For our convenience and without any loss of generality, we use only 2D functions, two variables x_1 and x_2. We learn the opposites with $n_s = 1000$ samples and then test the function optimization with $0.1 \times n_s$ new samples. Two random variables are generated and used to call FIS for their opposites. The combination of these two may provide a lower error as seen for the examined functions. Now, as we have established the superiority of evolved type II opposites over type I opposites in previous sections, here we also examine type II opposites in conjunction with random samples (also see [27], [26]). Here we would like to verify whether the fundamental statement of OBL is true: Simultaneous consideration of guess and opposite guess provides lower errors at each iteration of learning and optimization processes.

Given a function $f(x_1, x_2) = 0$ we generate two random samples x_1^r and x_2^r and we calculate the error. Then, we approximate the opposites $\breve{x}_1^r$ and $\breve{x}_2^r$ and calculate the error. As generally proposed in OBL schemes, we can simultaneously look at all results and continue with the best results depending on the evaluation of the function call. That means we continue with the results of x_1^r and x_2^r if $\text{Error}(f(x_1^r, x_2^r)) < \text{Error}(f(\breve{x}_1^r, \breve{x}_2^r))$ otherwise with $\breve{x}_1^r$ and $\breve{x}_2^r$. The opposites $\breve{x}_i^r$ can be type I (as used in existing literature) or type II (as we are proposing). The question is which one can provide a statistically significant benefit when used in conjunction with the initial random guess x_i^r.

Ackley Function – The Ackley function is given as

$$f(x_1, x_2) = 20\left(1 - \exp\left(-0.2\sqrt{0.5(x_1^2 + x_2^2)}\right)\right) - \quad (12)$$
$$\exp\left(0.5(\cos(2\pi x_1) + \cos(2\pi x_2))\right) + \exp(1).$$

The global minimum is $f(x_1, x_2) = f(3, 0.5) = 0$ where the search domain is $-35 < x_i < 35$. The results for Ackley function are given in Table II. The random guesses (first column) and their type I opposites (third column) were statistically the same (null hypothesis could not be rejected). Their simultaneous consideration therefore does not provide

TABLE I. THE RESULTS FOR TEST FUNCTIONS (BEST RESULTS HIGHLIGHTED): FOR EVERY FUNCTION BOTH TYPE I AND TYPE II OPPOSITES ARE CALCULATED USING THE THREE DIFFERENT OPPOSITENESS SCHEME. THE AVERAGE ERROR m AND ITS STANDARD DEVIATION σ IS CALCULATED USING DIRECT COMPARISON WITH THE KNOWN TRUE OPPOSITE VIA INVERSE FUNCTION.

Function and its inverse	Oppositeness scheme	Type-I Opposites $m \pm \sigma$ for 30 runs	Type-II Opposites $m \pm \sigma$ for 30 runs
$f_1(x) = (2x+8)^3$ $f_1^{-1}(x) = (y^{1/3}-8)/2$	T_1	378.91 ± 178.48	$\mathbf{22.62 \pm 24.01}$
	T_2	337.20 ± 303.62	78.12 ± 96.86
	T_3	234.27 ± 187.97	155.25 ± 228.39
$f_2(x) = \log(x+3)$ $f_2^{-1}(x) = \exp(y)-3$	T_1	478.06 ± 270.07	$\mathbf{18.95 \pm 29.47}$
	T_2	470.85 ± 296.33	24.00 ± 51.03
	T_3	529.12 ± 1881.12	557.06 ± 2051.47
$f_3(x) = 2x$ $f_3^{-1}(x) = y/2$	T_1	$\mathbf{0 \pm 0}$	0.02 ± 0.01
	T_2	$\mathbf{0 \pm 0}$	210.66 ± 137.20
	T_3	0.51 ± 2.90	0.82 ± 0.53
$f_4(x) = x^2$ $f_4^{-1}(x) = \sqrt{y}$	T_1	285.91 ± 117.22	$\mathbf{21.77 \pm 19.91}$
	T_2	248.87 ± 297.11	118.34 ± 129.50
	T_3	152.22 ± 129.68	120.17 ± 185.21
$f_5(x) = \sqrt{x}$ $f_5^{-1}(x) = y^2$	T_1	313.05 ± 140.76	$\mathbf{0.028 \pm 0.018}$
	T_2	317.94 ± 335.52	161.85 ± 156.19
	T_3	96.79 ± 99.41	101.09 ± 156.69
$f_6(x) = x^{3/2}$ $f_6^{-1}(x) = y^{2/3}$	T_1	183.72 ± 72.06	$\mathbf{15.38 \pm 13.35}$
	T_2	171.61 ± 280.48	150.75 ± 142.32
	T_3	83.11 ± 73.16	59.28 ± 98.56
$f_7(x) = x^3 + x^2 + 1$ (for $f_7^{-1}(x)$ see Eq.11)	T_1	380.63 ± 177.82	$\mathbf{22.98 \pm 26.77}$
	T_2	328.18 ± 297.96	75.55 ± 93.71
	T_3	131.57 ± 85.13	130.22 ± 177.85
$f_8(x) = 1/x$ $f_8^{-1}(x) = 1/y$	T_1	503.26 ± 292.20	3.46 ± 13.49
	T_2	496.60 ± 289.02	$\mathbf{0.56 \pm 0.27}$
	T_3	434.52 ± 338.03	39.72 ± 163.92
$f_9(x) = \sqrt{(x+1)}/3$ $f_9^{-1}(x) = 9y^2 - 1$	T_1	306.25 ± 138.12	$\mathbf{0.03 \pm 0.02}$
	T_2	299.37 ± 324.00	159.08 ± 165.29
	T_3	95.63 ± 92.46	97.60 ± 147.43

any benefit. In contrast, the type II opposites (second column) are not only different but also exhibit lower errors.

TABLE II. ERRORS FOR ACKELY FUNCTION

	x_1^r and x_2^r	$\breve{x}_{1,II}^r$ and $\breve{x}_{2,II}^r$	$\breve{x}_{1,I}^r$ and $\breve{x}_{2,I}^r$
1. run	284.89 ± 263.32	10.35 ± 9.21	284.89 ± 263.32
2. run	20.45 ± 2.39	2.09 ± 2.09	20.45 ± 2.39
3. run	20.61 ± 1.50	3.90 ± 0.41	20.61 ± 1.50
4. run	20.24 ± 2.27	2.40 ± 1.19	20.24 ± 2.27
5. run	20.63 ± 1.71	4.06 ± 1.01	20.63 ± 1.72

Booth Function – The Booth function is given as

$$f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2. \quad (13)$$

The global minimum is $f(x_1, x_2) = f(1, 3) = 0$ where the search domain is $-10 < x_i < 10$. The results for Booth function are given in Table III. Apparently, the random guesses (first column) and their type I opposites (third column) are the

same distribution. Their simultaneous consideration therefore does not provide any benefit. In contrast, the type II opposites (second column) are different. The simultaneous consideration of guess and opposite guess does provide a plausible benefit.

TABLE III. ERRORS FOR BOOTH FUNCTION

	x_1^r and x_2^r	$\breve{x}_{1,II}^r$ and $\breve{x}_{2,II}^r$	$\breve{x}_{1,I}^r$ and $\breve{x}_{2,I}^r$
1. run	448.78 ± 460.46	1930 ± 612.73	432.54 ± 444.90
2. run	403.22 ± 425.78	2000 ± 575.18	400.88 ± 424.10
3. run	470.40 ± 556.03	1875 ± 717.06	374.51 ± 388.70
4. run	382.35 ± 405.19	2103 ± 545.19	400.07 ± 465.64
5. run	422.20 ± 483.02	2058 ± 609.94	415.63 ± 455.30

Bukin4 Function – The Bukin4 function is given as

$$f(x_1, x_2) = 100\sqrt{||x_2 - 0.01x_1^2||} + 0.01||x_1 + 10||. \quad (14)$$

The global minimum is $f(x_1, x_2) = f(-10, 0) = 0$ where the search domain is $-15 < x_1 < -5$ and $-3 < x_2 < 3$. The
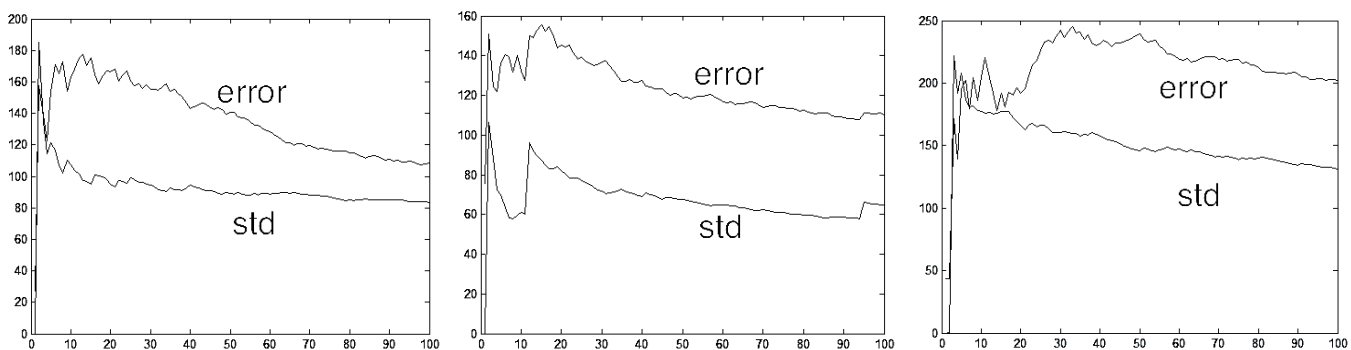
Fig. 3. The errors and standard deviations (std) for three different runs: In each run opposites are first trained offline with 100 samples (not shown here). The diagrams show the error of estimating the true opposites based on $f_1(x)$ (Table I) with 100 new opposites online whereas the rules are clustered and evolved with $101, 102, \cdots, 199, 200$ samples, respectively.
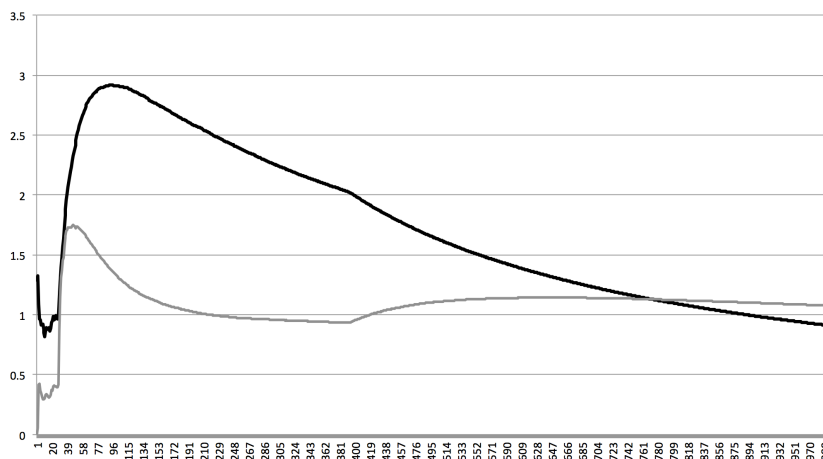

Fig. 4. The evolving opposites gradually become more accurate after processing each additional data point. The average error (black) and standard deviation (gray) decrease with each new sample. At the beginning there may be fluctuations due to lack of sufficient number of data to extract reasonable clusters and hence meaningful rules. Standard deviation may grow larger than mean if a small sample contains variables with large range.

results for Bukin4 function are given in Table IV. Apparently, the random guesses (first column) and their type I opposites (third column) are the same distribution (null hypothesis could not be rejected). Their simultaneous consideration therefore does not provide any benefit. In contrast, the type II opposites (second column) are both different and deliver lower errors.

TABLE IV. RESULTS FOR BUKIN4 FUNCTION

	x_1^r and x_2^r	$\breve{x}_{1,II}^r$ and $\breve{x}_{2,II}^r$	$\breve{x}_{1,I}^r$ and $\breve{x}_{2,I}^r$
1. run	290.14 ± 271.39	1.75 ± 1.59	290.14 ± 271.39
2. run	317.69 ± 253.78	7.34 ± 9.6	317.69 ± 253.78
3. run	301.05 ± 301.05	1.26 ± 1.03	301.05 ± 289.75
4. run	299.18 ± 262.70	49.06 ± 34.26	299.18 ± 262.70
5. run	282.59 ± 274.73	6.43 ± 5.01	282.59 ± 274.73

As apparent from the three tables, the simultaneous consideration of random guess and its opposite, as already demonstrated in existing literature, has clear benefit as always one of them is much closer to the solution. This shows that the learned type-II opposites do satisfy the fundamental assumption of OBL, namely that the simultaneous consideration of a guess and its opposite delivers a shorter path to the solution.

VI. SUMMARY

Ten years after introducing opposition-based learning, there exists no general algorithm to generate opposites for learning and optimization purposes. In this paper, taking the 50th anniversary of fuzzy sets as a pretext, we introduced an approach to learn true (type II) opposites via evolving fuzzy inference systems. The core idea in this paper is "opposition mining" to extract (quasi-)opposites by processing the available training data. This makes the learning of opposites possible as it extracts the necessary data (specially the desired outputs) for any learning scheme to approximate. The evolving fuzzy rules can then capitalize on these data points and refine the estimate of opposition with perpetual incorporation of future data points. We tested the proposed algorithm with simple test functions, examined its evolving aspect, and investigated three widely used optimization benchmark functions as well to verify both the correctness and usefulness of type II opposites.

We used the Takagi-Sugeno fuzzy systems in this work to learn the true opposites. This approach does employ a t-norm and hence triggers axis-parallel rules. However, It has been clearly shown in recent literature that arbitrarily rotated rules are able to outperform the conventional axis-parallel ones [13]. This is certainly a possibility for potential improvement

in future works. As well, in this work we only investigated the problems with one output. The extension to multiple output systems will certainly be of interest.

The authors are investigating the benefits of the proposed approach for complex learning and optimization problems, and hope to report the results in foreseeable future.

REFERENCES

[1] P. Angelov and R. Buswell, Evolving rule-based models: A tool for intelligent adaptation, in Proc. Joint 9th Int. Fuzzy Syst. Assoc. World Congr. 20th North Amer. Fuzzy Inf. Process. Soc. Int. Conf., vol. 2, pp. 1062–1067, 2001.

[2] P. Angelov and R. Buswell, Identification of evolving fuzzy rule-based models, IEEE Trans. Fuzzy Syst., vol. 10, no. 5, pp. 667–677, Oct. 2002.

[3] P. Angelov, A fuzzy controller with evolving structure, Inf. Sci., vol. 161, nos. 1–2, pp. 21–35, 2004.

[4] P. Angelov, E. Lughofer, and X. Zhou, Evolving fuzzy classifiers using differentmodel architectures, Fuzzy Sets Syst., vol. 159, no. 23, pp. 3160–3182, 2008.

[5] P. Angelov and X. Zhou, Evolving fuzzy-rule-based classifiers from data streams, IEEE Trans. Fuzzy Syst., vol. 16, no. 6, pp. 1462–1475, 2008.

[6] J. de Barros and A. Dexter, On-line identification of computationally undemanding evolving fuzzy models, Fuzzy Sets Syst., vol. 158, no. 18, pp. 1997–2012, 2007.

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

[8] F. S. Al-Qunaieer, H.R. Tizhoosh, S. Rahnamayan, Oppositional fuzzy image thresholding, IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–7, 2010.

[9] H. Salehinejad, S. Rahnamayan, H.R. Tizhoosh, Type-II opposition-based differential evolution, 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1768–1775, 2014.

[10] E. Lughofer, On-line evolving image classifiers and their application to surface inspection, Imag. Vis. Comput., vol. 28, no. 7, pp. 1065–1079, 2010.

[11] E. Lughofer, Evolving Fuzzy Systems – Methodologies, Advanced Concepts and Applications, Springer, Berlin Heidelberg, 2011.

[12] E. Lughofer, Evolving Fuzzy Systems – Fundamentals, Reliability, Interpretability, Useability, Applications, in: Handbook of Computational Intelligence, editor: P. Angelov, World Scientific, 2015.

[13] E. Lughofer, C. Cernuda, S. Kindermann and M. Pratama, Generalized Smart Evolving Fuzzy Systems, Evolving Systems, on-line and in press, 2015, DOI:10.1007/s12530-015-9132-6.

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

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

[16] A.A. Othman, H.R. Tizhoosh, Evolving fuzzy image segmentation, IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1603–609, 2011.

[17] A.A. Othman, H.R. Tizhoosh, F. Khalvati, EFIS–evolving fuzzy image segmentation, IEEE Transactions on Fuzzy Systems, 22 (1), 72–82, 2014.

[18] M. Pratama and S.G. Anavatti and P. Angelov and E. Lughofer, PANFIS: A Novel Incremental Learning Machine, IEEE Transactions on Neural Networks and Learning Systems, vol. 25 (1), pp. 55–68, 2014.

[19] S. Rahnamayan, H.R. Tizhoosh, M.M.A., Salama, Opposition-based differential evolution algorithms. IEEE Congresson Evolutionary Computation, Vancouver, Canada, pp. 2010–2017, 2006.

[20] S. Rahnamayan, Opposition-based differential evolution. Ph.D thesis, University of Waterloo, Waterloo, Canada, 2007.

[21] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, A novel population initialization method for accelerating evolutionary algorithms. Comput. Math. Appl. 53(10),1605–1614, 2007.

[22] S. Rahnamayan, H. R. Tizhoosh, Image thresholding using micro opposition-based differential evolution (micro-ODE), IEEE Congress on Evolutionary Computation, CEC 2008, pp. 1409–1416, 2008.

[23] S. Rahnamayan, G.G. Wang, Center-based sampling for population-based algorithms. IEEE Congress on Evolutionary Computation, Trondheim, Norway, pp.933–938, 2009.

[24] S. Rahnamayan, J. Jesuthasan, F. Bourennani, H. Salehinejad and G.F. Naterer. "Computing opposition by involving entire population." IEEE Congress on Evolutionary Computation (CEC), pp. 1800–1807, 2014.

[25] S. Rahnamayan et al., "Centroid Opposition-Based Differential Evolution." International Journal of Applied Metaheuristic Computing (IJAMC) 5.4: 1-25, 2014.

[26] S. Rahnamayan, H. Gary Wang, and M. Ventresca. "An intuitive distance-based explanation of opposition-based sampling." Applied Soft Computing 12.9: 2828–2839, 2012.

[27] S. Rahnamayan, H.R. Tizhoosh, and M.M.A. Salama. "Opposition versus randomness in soft computing techniques." Applied Soft Computing 8.2: 906–918, 2008.

[28] S. Rahnamayan, H. R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution, IEEE Transactions on Evolutionary Computation, vol. 12, issue 1, pp. 64–79, 2008.

[29] F. Sahba, H.R. Tizhoosh, M.M.A. Salama, Application of opposition-based reinforcement learning in image segmentation, IEEE Symp.on Comp. Intell. Image and Signal Processing, pp. 246–251, 2007.

[30] H. Salehinejad, S. Rahnamayan, H.R. Tizhoosh, Type-II opposition-based differential evolution. IEEE Congress on Evolutionary Computation (CEC), pp. 1768–1775, 2014.

[31] T. Takagi and M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE transactions on systems, man, and cybernetics, vol. 15, no. 1, pp. 116132, 1985.

[32] H.R. Tizhoosh, Opposition-Based Learning: A New Scheme for Machine Intelligence; Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation - CIMCA2005, Vienna - Austria, vol. I, pp. 695–701, 2005.

[33] H.R. Tizhoosh, Reinforcement learning based on actions and opposite actions, International Conference on Artificial Intelligence and Machine Learning, pp. 94–98, 2005.

[34] H.R. Tizhoosh, Opposition-Based Reinforcement Learning, JACIII, vol.10, issue 4, pp. 578–585, 2006.

[35] H.R. Tizhoosh, M. Ventresca, Oppositional Concepts in Computational Intelligence. Springer, XII, Hardcover. ISBN: 978-3-540-70826-1, 2008.

[36] H.R. Tizhoosh, Opposite Fuzzy Sets with Applications in Image Processing, IFSA/EUSFLAT, pp. 36–41, 2009.

[37] H.R. Tizhoosh, F. Sahba, Quasi-global oppositional fuzzy thresholding, IEEE Int. Conf. on Fuzzy Systems, pp. 1346–1351, 2009.

[38] M. Ventresca, H.R. Tizhoosh, Simulated annealing with opposite neighbors, IEEE Symposium on Foundations of Computational Intelligence, FOCI 2007, pp. 186 –192, 2007.

[39] M. Ventresca, H.R. Tizhoosh, Opposite transfer functions and backpropagation through time, IEEE Symposium on Foundations of Computational Intelligence, pp. 570–577, 2007.

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

[41] R.R. Yager, A model of participatory learning. IEEE Transactions on Systems, Man and Cybernetics, vol. 20, n. 5, pp. 1229–1234, 1990.