Memetic Comp. (2011) 3:101–110
DOI 10.1007/s12293-010-0054-3

REGULAR RESEARCH PAPER

WRAO and OWA learning using Levenberg–Marquardt and genetic algorithms

B. S. U. Mendis · T. D. Gedeon

Received: 6 July 2009 / Accepted: 10 November 2010 / Published online: 22 December 2010

Abstract The generalized Weighted Relevance Aggregation Operator (WRAO) is a non-additive aggregation function. The Ordered Weighted Aggregation Operator (OWA) (or its generalized form: Generalized Ordered Weighted Aggregation Operator (GOWA)) is more restricted with the additivity constraint in its weights. In addition, it has an extra weights reordering step making it hard to learn automatically from data. Our intension here is to compare the efficiency (or effectiveness) of learning these two types of aggregation functions from empirical data. We employed two methods to learn WRAO and GOWA: Levenberg–Marquardt (LM) and a Genetic Algorithm (GA) based method. We use UCI (University of California Irvine) benchmark data to compare the aggregation performance of non-additive WRAO and additive GOWA. We found that the non-constrained aggregation function WRAO was learnt well automatically and produced consistent results, while GOWA was learnt less well and quite inconsistently.

Keywords WRAO · OWA · GOWA · The Levenberg–Marquardt · Genetic algorithms

1 Introduction

Mendis et al. [27] proposed a non-additive aggregation function called Generalized Weighted Relevance Aggregation Operator (WRAO). The idea behind WRAO is that the weights for each input are the observations of the relevance of that input to the output. Thus, WRAO introduces extra knowledge but is not constrained by additivity in its modeling of data. WRAO has shown success in medical diagnosis [25,26,34], personnel management [25,26,34], eye gaze path analysis [37], and document analysis [22], when used as an aggregation function for the hierarchical structure of fuzzy signatures. In this paper, however, we compare non-hierarchical WRAO (with no support from fuzzy signatures) and GOWA for 3 different benchmark UCI data sets, namely Iris, Wine, and Abalone [1].

The Ordered Weighted Averaging (OWA) aggregation was introduced in 1988 [35]. Following are some recent applications of OWA. Kacprzyk et al. [19], use an OWA operator based approach for the linguistic summarization of time series data. An OWA based linguistic bi-objective genetic algorithm has been used for personal management [17]. Bordogna and Pasi [5] use OWA for indexing heterogeneous structured documents and for retrieving semi-structured documents. As many others in the past, Ben-Arieh [4] used OWA in multi-criteria decision making (MCDM). Torra and Godo [33] analyzed the extended class of OWA called weighted OWA (WOWA) in the defuzzification process.

The OWA operator provides a class of mean type aggregation selections based on weight parameters. OWA has an extra weights reordering step compared to other mean type aggregations. This allows OWA to be the only aggregation function that follows generalized commutativity [35]. However, it is a crucial question to answer whether this reordering gives an extra benefit to successful applicability of the OWA or it is an extra burden in the training phase of OWA weights. Further, OWA is an aggregation function that is constrained by the additivity property. Later, Yager introduced Generalized Ordered Weighted Aggregation Operator (GOWA) [36], which generalized the weights and aggregation in OWA.

B. S. U. Mendis (✉) · T. D. Gedeon
School of Computer Science,
The College of Engineering and Computer Science,
The Australian National University, Acton, ACT, Australia
e-mail: sumudu@cs.anu.edu.au

T. D. Gedeon
e-mail: tom@cs.anu.edu.au

Therefore, we concentrate on the generalized OWA (GOWA) in the rest of the paper.

The major drawback with the steepest gradient descent method is that if there is no line search method combined with it, there is no guarantee of convergence. The Levenberg–Marquardt (LM) algorithm [21,23] is a widely used advanced, second order derivative, optimization algorithm that outperforms simple first order gradient descent methods and other gradient methods when applied in a wide variety of problems. It also converges faster than the steepest gradient decent method [6]. Therefore, we use the Levenberg–Marquardt method as a fast converging local optimization learning algorithm for learning weights and aggregation parameters in WRAO and GOWA.

As the Levenberg–Marquardt method is a local optimization technique it may only find a suboptimal solution. Further, the search for a good solution can be biased based on the starting position of the algorithm. Genetic algorithms (GA) are well known global search techniques used in optimization [15]. Genetic algorithms are computational abstractions of the genetic processes of biological organisms. A Genetic algorithm starts with a random population of individuals each representing a possible solution to the problem. These individuals form the population and are assessed using a fitness function. High fitness individuals are given the opportunity to survive and reproduce a new generation by crossover with other individuals in the population and undergo mutation. This process will continue until a good solution to the problem is found [15].

2 Generalized weighted relevance aggregation operator (WRAO)

We [27] proposed a monotonic aggregation function called Generalized Weighted Relevance Aggregation Operator (WRAO). The weights in WRAO are monotonic and thus they are not necessarily additive. Additionally, unlike in other weighted aggregation functions [9,36], in WRAO the weights are only used as observations of the relevance of that input to the knowledge it represents. Thus, WRAO introduces extra knowledge but is not constrained by the additivity in its modeling of data. We recall the definition of WRAO in [27].

Definition 1 The generalized Weighted Relevance Aggregation Operator (WRAO) of n inputs, $a_1, a_{i2}, \ldots, a_n \in [0, 1]$, and weighted relevancies, $w_1, w_2, \ldots, w_n \in [0, 1]$, is a function $g : [0, 1]^{2n} \to [0, 1]$ such that,

$$a = \left[\frac{1}{n} \sum_{j=1}^{n} (a_j \; w_j)^p \right]^{\frac{1}{p}} \tag{1}$$

The WRAO must satisfy the following three properties,

(i) $w_j \in [0, 1]$
(ii) $\bigvee_{j=1}^{n} w_j \leq 1$ [1]
(iii) $p \neq 0$

In [27], we prove the following two properties for WRAO.

Theorem 1 *WRAO in Definition (1) holds the following properties.*

(i) *Idempotent w.r.t a_j, when all w_j are fixed and vice versa,*
(ii) *Commutative, and*
(iii) *Monotonic w.r.t a_j when all w_j are fixed and vice versa.*

The above properties are adequate to satisfy the requirement to be an aggregation function [3], as weights, $w_1, w_2, \ldots, w_n$, in WRAO are fixed for any instance of the decision making phase, and both weights and aggregation operators vary simultaneously only in the learning phase.

Theorem 2 *The WRAO in Definition (1) holds the following characteristics.*

(a) $p \to 0$ *then WRAO* $\to$ *geometric mean*
(b) $\lim_{p \to +\infty} g(a_1, \ldots a_{in}; w_{i1}, \ldots, w_{in})$

$$= max(a_{i1} w_{i1}, \ldots a_{in} w_{in})$$

(c) $\lim_{p \to -\infty} g(a_1, \ldots a_n; w_{i1}, \ldots, w_{in})$

$$= min(a_{i1} w_{i1}, \ldots a_{in} w_{in})$$

(d) $p = 1$ *then WRAO*$\to$ *arithmetic mean*
(e) $p = -1$ *then WRAO*$\to$ *harmonic mean*

3 Generalized ordered weighted aggregation operator (GOWA)

In this section we discuss the basic concepts of Ordered Weighted Averaging (OWA) and generalized Ordered Weighted Averaging (GOWA) [36].

3.1 Ordered weighted average (OWA) operators

Definition 2 The OWA operator of n input arguments $a_1, a_2, \ldots, a_n$ is a mapping $f : [0, 1]^n \to [0, 1]$ that has an associated weighting vector $W = [w_1, w_2, \ldots, w_n]^T$ such that:

[1] The notation $\bigvee$ represents the *maximum* operation [8].

(i) $w_i \in [0, 1]$
(ii) $\sum_{i=1}^{n} w_i = 1$
and where

$$f\,(a_1, a_2, \ldots, a_n) = \sum_{j=1}^{n} w_j b_j \tag{2}$$

where b_j is the jth largest element of the n input argument vector.

OWA operators include *min*, *max* and *mean*. Yager [35] has shown that OWA satisfies the basic properties of monotonicity, idempotence, and generalized commutativity.

In [35] Yager further discusses that a desire to satisfy the property *generalized commutativity* is what forced the use of ordered weighted average rather than weighted average. When the weighting vector takes specific combinations of values then the OWA represents the following classes of aggregation functions [12,13,35].

(i) $min(a_1, a_2, \ldots, a_n)$ if $W = [0, 0, \ldots, 1]$.
(ii) $max(a_1, a_2, \ldots, a_n)$ if $W = [1, 0, \ldots, 0]$.
(iii) any order statistic b_k if $w_k = 1$ and $w_i = 0\ \forall i \neq k$
(iv) arithmetic mean of $(a_1, a_2, \ldots, a_n)$ if $W = \left[\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right]$

Therefore, we observe that ordered weighting in OWA is a process that supports OWA to achieve some desired classes of aggregation functions, such as *min*, *max*, *mean*.

3.2 Generalized ordered weighted average (GOWA) operator

Definition 3 A mapping $M : I^n \to I$ is called a generalized ordered weighted aggregation (GOWA) operator of n input arguments if

$$M\,(a_1, a_2, \ldots, a_n) = \left(\sum_{j=1}^{n} w_j b_j^p\right)^{\frac{1}{p}} \tag{3}$$

where $p \in [-\infty, \infty]$, b_j is the jth largest input, and w_j are weights satisfying

(I) $w_j \in [0, 1]$
(II) $\sum_{j=1}^{n} w_j = 1$

It is clear that the GOWA operators include the above 4 special classes of aggregation functions with respect to the same weighting vectors as the OWA regardless of any value of the parameter p in Eq. (3). Hence, the parameter p in Eq. (3) also controls the meaning of the aggregation of GOWA and additionally gives same special classes of aggregation functions for different values of p [2,36]. In this case GOWA will generate large numbers of aggregation classes and some may give the same results with respect to inputs.

We can conclude that the weights in the context of OWA (GOWA) are essentially focused to be supporting factors to achieve the desired classes of aggregation functions. Unlike OWA type aggregation, WRAO uses the weights to modify the meaning before aggregating them to the next level, and achieves the different classes of aggregation functions discussed above using the aggregation factor p in Eq. 1. Fundamentally, OWA uses an additive measure as weights parameters while WRAO is not constrained to an additive measure.

4 Learning of WRAO and OWA from real world data

Methods of learning the weights in OWA and GOWA from data using Levenberg–Marquardt and Genetic Algorithm learning are discussed in this section.

4.1 Levenberg–Marquardt optimization method

The LM algorithm is a widely used advanced optimization algorithm that outperforms simple gradient descent and other gradient methods when applied to a wide variety of problems. The major drawback with the steepest gradient descent method is that if there is no line search method combined with it, there is no guarantee of convergence. LM is a pseudo-second order method in which the Hessian matrix is estimated using the gradients [11]. The LM algorithm is a Least Squared Error (LSSE) based minimization method that is the function to be minimized is of the following special form as in Eq. 4 [23].

There are many applications which use least square estimation to achieve good results. There exist special optimization algorithms, such as the Levenberg–Marquardt method, to optimize the sum of squared error for practical problems. The objective function of sum of m squared terms is of the following form:

$$e(x)_k = \frac{1}{2}\sum_{i=1}^{m}(\theta_i - x_i)^2 = \frac{1}{2}\|r(x)\|_2^2 = r^T r \tag{4}$$

where θ_i is the target output and x_i is the predicted output for the ith component. The $\|r(x)\|$ is called the residual at x (the $\frac{1}{2}$ has been introduced in Eq. (4) to avoid the appearance of factor of 2 in the derivative of Eq. (4)) [7,11,14,31]. Further, k denote the iteration number for the calculated error $e(x)$.

Now, the next update of the LM can be written as:

$$\underline{u}_k = \underline{par}_k - \underline{par}_{k-1} \tag{5}$$

The vector $\underline{par}_k$ contains all learning parameters, i.e. all aggregation factors and weights factor, of WRAO in Eq. (1)

and GOWA in Eq. (3) for the kth iteration. The LM defines the next update $\underline{u}_k$ in following manner:

$$(J_k^T J_k + \alpha I)\underline{u}_k = -J_k^T \underline{r}_k \tag{6}$$

The J is the Jacobian matrix of the Eq. (4), I is the identity matrix of J, and α is a regularization parameter, which controls both search direction and magnitude of the next update $\underline{u}_k$.

The LM algorithm uses the restricted step size method to find best quasi-optimal solution. The following algorithm has been given in [11] for calculating the next update $\underline{\alpha}_k$ and using a trust region h_k, which is given in Eq. (7), to achieve good convergence:

I. Given $\underline{par}_k$ and $\underline{\alpha}_k$, use (6) to find $\underline{u}_k$
II. Calculate $\underline{h}_k$ use (7)
If $\underline{h}_k < 0.25$ set $\underline{\alpha}_{k+1} = 4\underline{\alpha}_k$
If $\underline{h}_k > 0.75$ set $\underline{\alpha}_{k+1} = \frac{\underline{\alpha}_k}{2}$
Otherwise $\underline{\alpha}_{k+1} = \underline{\alpha}_k$
III. If $\underline{h}_k \leq 0$ set $\underline{par}_{k+1} = \underline{par}_k$
Else $\underline{par}_{k+1} = \underline{par}_k + \underline{u}_k$
IV. Find new error $\underline{e}_{k+1}$, use Eq. (4)
If $\underline{e}_{k+1} >$ *threshold* then go to I.
Else stop learning.

where k is the current iteration number. The trust region $\underline{h}_k$ calculation for above algorithm is given by:

$$\underline{h}_k = \frac{f\left(\underline{par}_{k-1}\right) - f\left(\underline{par}_k\right)}{f\left(\underline{par}_{k-1}\right) - q\left(\underline{par}_k\right)} \tag{7}$$

where approximation of the error, $q\left(\underline{par}_k\right) = \left\{J_k^T \underline{u}_k + \underline{e}_k\right\}$. Initially, the algorithm starts by choosing an arbitrary $\underline{\alpha}_k > 0$ and arbitrary values for $\underline{par}_k$.

4.1.1 Aggregation and weights learning of WRAO

In this subsection we explain the method of learning WRAO from real world data briefly, with more detailed explanations to be found in [27]. First, to avoid the first 2 constraints on the weighted relevance factor w_i in Definition 1, we replaced it by the following sigmoid function,

$$w_i = \frac{1}{1 + e^{-\lambda_i}} \tag{8}$$

where $\lambda_i \in \mathbb{R}$. Now, the Eq. (1) can be modified as follows,

$$a = \left[\frac{1}{n}\sum_{j=1}^{n}\left(a_i \left[\frac{1}{1 + e^{-\lambda_i}}\right]\right)^p\right]^{\frac{1}{p}} \tag{9}$$

This form of WRAO (9) can be readily used for gradient based learning. The parameters we need to learn are the aggregation factor p and weighted relevance factors λ_i for WRAO. First we can obtain the partial derivatives of the Eq. (9) w.r.t. p,

$$\frac{\partial a}{\partial p} = \left[\frac{a^{1-p}}{np^2}\right]\left\{\sum_{j=1}^{n} t\ln(t) - nt'\ln\left(t'\right)\right\} \tag{10}$$

where $t = (a_i w_i)^p$ and $t' = a^p$. Similarly, we obtain the partial derivatives of the Eq. (9) w.r.t. λ_{ik}

$$\frac{\partial a}{\partial \lambda_{ik}} = \left[\frac{1}{n}\sum_{j=1}^{n}(a_i\ w_i)^p\right]^{\frac{1}{p}-1} T \tag{11}$$

where $w_i = \frac{1}{1+e^{-\lambda_i}}$ and $T = \left\{\frac{d([a_{ik}\ w_{ik}]^p)}{d\lambda_{ik}}\right\}$.

4.1.2 Aggregation and weights learning of GOWA

Beliakov [2] has shown a way of learning weights in GOWA. We use our LM based method for learning [27,28] of both weights and the parameter p in GOWA operators. First, to avoid the 2 constraints on the weights w_i in Definition 3, we replaced it by the following function [10],

$$w_i = \frac{e^{\lambda_i}}{\sum_{k=1}^{n} e^{\lambda_k}} \tag{12}$$

where $\lambda_i \in [0, 1]$ and $k \in [1, n]$. Now, the Eq. (3) can be modified as follows,

$$a = \left(\sum_{i=1}^{n}\left[\frac{e^{\lambda_i}}{\sum_{k=1}^{n} e^{\lambda_{ik}}}\right] b_i^p\right)^{\frac{1}{p}} \tag{13}$$

Now, the parameters need to be learnt are the p and λ_i for GOWA. First we obtain the partial derivatives of the Eq. (13) w.r.t. p,

$$\frac{\partial a}{\partial p} = \left[\frac{a^{1-p}}{p^2}\right]\left\{\sum_{i=1}^{n} t - \left[t'\ln\left(t'\right)\right]\right\} \tag{14}$$

where $t = [w_i b_i \ln(b_i)]$ and $t' = \sum_{k=1}^{n} w_k b_k$. Similarly, we obtain the partial derivatives of the Eq. (13) w.r.t. λ_i

$$\frac{\partial a}{\partial \lambda_i} = \frac{t^{\left(\frac{1}{p}-1\right)}}{p}\left[w_i\left(b_i^p - t\right)\right] \tag{15}$$

where $t = \sum_{i=1}^{n} w_i b_i^p$.

4.2 Genetic algorithm method

Holland first raised the idea of genetics algorithm, which were inspired by the mechanisms of natural evolution of genetics, in his seminal work [18]. David Goldberg, popularized and showed a way to solve real world problems

using Genetic Algorithms [15]. He solved a difficult problem involving the control of gas-pipeline transmission [16].

The Genetic Algorithm is a stochastic optimization method. It starts with parallel random starting points called the initial population. Each individual in the population represents a possible solution to the problem. Genetic Algorithm find a good approximations by applying the survival of the fittest procedure to the population of solutions. This will allow the later population to be adapted more, compared to the earlier population. Through the evolution, the crossover and mutation help the individuals to survive. A typical Genetic algorithm contain the following steps:

1. Initialize population;
2. Evaluate Population;
3. While good individual found
   - i Select good individuals;
   - ii Crossover and Mutation;
   - iii Evaluate Population;
4. End

4.3 Encoding and initial population

In genetic algorithms individuals are called chromosomes and mostly they encoded as bit strings. An alphabet is used to map the values in the chromosomes (genotypes) to real variable (phenotype) domain. Here we use integer binary bit representation as the alphabet. For WRAO and GOWA, according to Eqs. (1) and (3), the parameters need to be optimized are the aggregation p and weights $w_1, \ldots, w_n$. Figure 1 shows an example of a chromosome we used.

A large number of optimization problems have real-valued variables. The bit representation of the chromosome is the most commonly used encoding method in genetic algorithms [30,32]. Bit representation is simpler to create and manipulate, many types of information can be easily encoded, and the genetic operators are easily applied [30]. Our genetic algorithm starts with a random population of individuals each encoded as in Fig. 1 and representing a possible solution to the problem. One important issue which needs to be considered is the diversity of the initial population generated: if the diversity is too low, the genetic algorithm may converge to a sub-optimal solution.

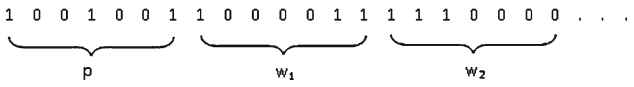

Fig. 1 WRAO/GOWA chromosome

4.4 Evaluation and fitness function

Only phenotypic values contain the real meaning of the solution, but the genotype (chromosome) yields no information about the problem. The search process will operate on the these genotypic values rather than the real values. These individuals form the population and are assessed using a fitness function.

The population is evaluated against a fitness function during each iteration and ranked based on the outcome. The fitness function is a critical component to find a good solution and thus problem dependent [20]. Here we use mean squared error (MSE) as the fitness function (see Eq. 16). The mean squared error between target θ and predicted output x for n record can be written as:

$$MSE(x) = \frac{1}{n}\sum_{i=1}^{n}(\theta_i - x_i)^2 \tag{16}$$

High fitness ranked individuals are given the opportunity to survive and reproduce a new generation by crossover with other individuals in the population and undergo mutation [15].

4.5 Selection

During the selection, the individual chromosomes are selected according to their fitness, which is evaluated using an objective function shown in Eq. (16). Once the evaluation done, the chromosome with a higher fitness value will have a higher probability of contributing one or more chromosomes to the next generation. In the literature [15,30,32], there are many existing methods the selection can be done. The most common method is using a weighted roulette wheel with slots sized according to fitness [30]. Thus, similar to the natural phenomenon of survival, on the roulette wheel the highest fitness individual will have a better chance to survive compared to the other individuals in the population [30].

4.6 Crossover

During first step of the crossover each of two selected parent chromosomes will be divided into two halves: left and right sub-chromosomes. Now they swap parts to generate two new offspring. The simplest method is to do the single point crossover. A randomly chosen single point in between the length of the chromosome will be used for this method. Therefore, each child gets one part of each parent. Further advancement of this method is to use 2 point crossover. In general, studies have shown that two point crossover works better compared to single point crossover [30].

Crossover helps to increase the diversity in the population and advance through good direction of optimization. However, crossover is not always effective. Therefore, only a portion of the population will go through the crossover to generate new individuals and the crossover rate will define the numbers of chromosomes crossed over.

4.7 Mutation

Once the crossover is completed, the mutation can produce new offspring that are in new areas of the search space. In general, during mutation when a parent chromosome is chosen, a randomly selected part of the chromosome will be modified at random. Mutation of a bit is done by changing its value from 1 to 0 or vice versa. The mutation rate gives a probability that a bit will be flipped [29,30]. The great advantage of mutation is that it will help avoid local minima which exist in the search space.

We use simple genetic algorithms to learn WRAO and GOWA as in Eqs. (1) and (3) respectively. We learnt aggregation p and weights $w_1, \ldots, w_n$ for both WRAO and GOWA separately using the genetic algorithm discussed above. However as GOWA has the extra weights reordering and additivity of weights constraint, the genetic algorithm learning algorithm for GOWA is computationally more expensive compare to that for WRAO.

In summary, the genetic algorithms starts with several parallel initial starting points but the Levenberg–Marquardt method uses only one starting point at a time. Further, the mutation operation in the genetic algorithms helps to avoid locally optimized solutions. The genetic algorithm method is a global optimization method. In other words, the genetic algorithm method is capable of searching a wider search space area compare to the Levenberg–Marquardt method. Thus, we here expect that the resulting aggregation function learning using the genetic algorithm optimization to be better or equal to that of the Levenberg–Marquardt method. Therefore, if an aggregation function follows this criteria during the learning; we can conclude that it has no irregular properties or constraints that affect the learning criteria. In the next section, using some experiments with benchmark data sets, we show that WRAO simply follows this general criterion about local and global optimization. However, we further show that GOWA fails to follow this criterion and the results are worse compared to that of WRAO.

5 Experiment: learning WRAO and GOWA from empirical data

In a summary, the two main differences between WRAO and GOWA are: GOWA has an extra weights reordering step and it's weights need to be summed up to 1. Our main scope of this experiment is to check the effect of the these two differences when learning WRAO and GOWA from empirical data. We used Levenberg–Marquardt and genetic algorithm methods for the learning of WRAO and GOWA as described in Sect. 4. Further, we use 3 different UCI [1] benchmark data sets, namely *Iris*, *Wine*, and *Abalone* [1] to demonstrate the experiment.

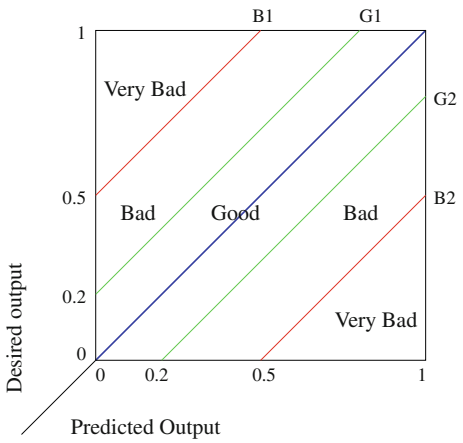

Fig. 2 Fuzzy classification error rules

Next, before we go to the details of the experiment, we first describe the *Fuzzy Classification Error* which has been used in this paper as a measure of error, in addition to the mean squared error.

5.1 Fuzzy classification error

We, first, briefly introduce the concept of Fuzzy Classification Error (FYCLE) that can be used to better illustrate and compare the results of the experiments. The Mean Squared Error (MSE) is good at showing the quantitative error but in real world applications sometimes does not visualize the qualitative grading of the results. Therefore, we introduced Fuzzy Classification Error (FYCLE) [24,26] for better analysis and visualization of qualitative grading of results.

We formulate the Fuzzy Classification Error (FYCLE) in the following way. First, we specify that both *desired* output and *predicted* output of an experiment are in the range [0, 1]. Next we define a set of rules for the classification, these rules are visualized in the following Fig. 2.

According to Fig. 2, there are 3 categories of classifications that can occur, being *Good*, *Bad* and *Very Bad (VB)*. Now we assume the pair of *predicted* and *desired* values of the ith input, respectively, are taken as x and y coordinates of the point P_i on the 2 dimensional classification error space in Fig. 2. The Fuzzy Classification Error of an arbitrary point P_i can be written as,

$$FYCLE(P_i) = \begin{cases} 0 & \text{if } P_i \in Good \\ 0.5 & \text{if } P_i \in Bad \\ 1 & \text{if } P_i \in Very\ Bad \end{cases}$$

Let us consider the 4 straight lines, $B1$, $B2$, $G1$, and $G2$, in Fig. 2. In this experiment, they are equivalent to,

$$B1 \equiv y - x - 0.5$$
$$G1 \equiv y - x - 0.2$$
$$G2 \equiv y - x + 0.2$$
$$B2 \equiv y - x + 0.5$$

Now, The Fuzzy Classification Error of an arbitrary point P_i can be calculated as,

$$FYCLE(P_i) = \begin{cases} 0 & \text{if } G1(P_i) \leq 0\ AND\ G2(P_i) \geq 0 \\ 0.5 & \text{if } (B1(P_i) \leq 0\ AND\ G1(P_i) > 0) \\ & \quad OR \\ & \quad (G2(P_i) < 0\ AND\ B2(P_i) \geq 0) \\ 1 & \text{if } B1(P_i) > 0\ OR\ B2(P_i) < 0 \end{cases}$$

Next, the Sum of Fuzzy Classification Error (SYCLE) for a set of data with m records can be calculated as,

$$SYCLE(P) = \sum_{i=1}^{m} FYCLE(P_i) \quad \text{where } m \in \mathbb{N} \tag{17}$$

Further, we can define the Average Fuzzy Classification Error (AVGFYCLE) as follows:

$$AVGFYCLE(P) = \frac{1}{m} \sum_{i=1}^{m} FYCLE(P_i) \quad \text{where } m \in \mathbb{N} \tag{18}$$

The AVGFYCLE, is more useful when we need to compare the results of the same experiment carried out with two or more different data sets and the number of data points in these data sets are different. In this paper, we always reprise the same data sets for all experiments. Therefore, the Sum of Fuzzy Classification Error (SYCLE) is adequate to visualize and classify the results of the experiments in this paper.

5.2 Wine data set

The data set contains a result of a chemical analysis of wines grown in the same region in Italy [1] but derived from three different cultivars. Based on the quantities of 13 variables three types of wines need to be classified [1]. Table 1 shows the results of the WRAO and GOWA learning.

According to the Table 1, Levenberg–Marquardt learning of MSE decreases by 4.02% when GA learning is applied for WRAO and it show 7.33% of MSE decrease for the of testing phase of WRAO. Further, sum of fuzzy classification error (SYCLE) of test phase is also has reduced amount for the GA results of WRAO. However, GOWA learning has slightly increased MSE when GA is used for training phase and sum of fuzzy classification error remain unchanged for both Levenberg–Marquardt and GA methods. According to Table 1, we can see that WRAO learning is consistent, ie. the global optimizer has reduced error compared to the local optimizer's error, while GOWA show an irregular learning pattern for Wine data. Overall, Genetic algorithm learning of WRAO shown the best performance for Wine data.

Table 1 Wine: Levenberg–Marquardt and GA learning of WRAO & GOWA

		MSE train	SYCLE train	MSE test	SYCLE test
LM	WRAO	0.0323	13	0.0327	13
	GOWA	0.0447	16	0.0450	16.5
GA	WRAO	0.0310	13.5	0.0303	12.5
	GOWA	0.0449	16	0.0447	16.5

For the test phase of the Wine data prediction experiment between WRAO and GOWA the GA method give the best performances. Therefore, in Fig. 3, we visualize the classification error of WRAO and GOWA with GA method in the test phase. According to Fig. 3a and b, we can see that all the predicted results of GOWA lie around a neighborhood of 0.3 while the desired output result varies from 0 to 1. However, WRAO's results are more spread in the fuzzy classification error space and has more results which lie on or closer to the $x = y$ line in the fuzzy classification error space.

5.3 Iris data set

The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant [1]. Table 2 shows the results of the WRAO and GOWA learning.

This time, as the classification in the data set is not very complicated [1], for WRAO, both local and global search converged to the same minimum. As shown by Table 2, Levenberg–Marquardt learning of MSE remains the same when GA learning is applied for GOWA. However, GA training has 28% less sum of fuzzy classification error (SYCLE) compared to that of the Levenberg–Marquardt training. During the testing phase MSE increased by 20% and sum of fuzzy classification error (SYCLE) increased by 55% for GA compared to Levenberg–Marquardt. According to Table 2, we can see that again WRAO learning is consistent, ie. the global optimizer has reduced error compared to the local optimizer's error, while GOWA show an irregular learning pattern for Iris data as well. Overall, so far, WRAO shown the best performance for both Wine and Iris data.

For the test phase of the Iris data classification experiment between WRAO and GOWA, the GA and LM methods give

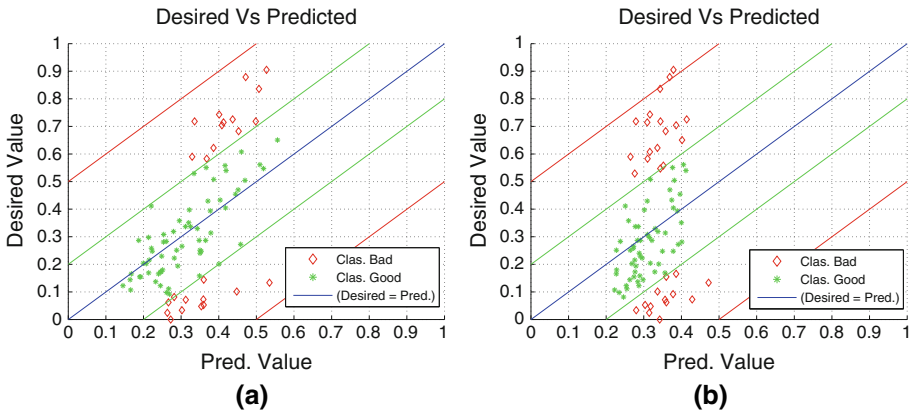

Fig. 3 Wine: SYCLE for WRAO and GOWA. **a** WRAO: GA test; **b** GOWA: GA test

Table 2 Iris: Levenberg–Marquardt and GA learning of WRAO & GOWA

		MSE train	SYCLE train	MSE test	SYCLE test
LM	WRAO	0.0199	6.5	0.0170	4
	GOWA	0.0208	7	0.0177	4.5
GA	WRAO	0.0199	6.5	0.0170	4
	GOWA	0.0208	5	0.0213	7

Table 3 Abalone: Levenberg–Marquardt and GA learning of WRAO & GOWA

		MSE train	SYCLE train	MSE test	SYCLE test
LM	WRAO	0.0093	51	0.0093	54.5
	GOWA	0.0096	53	0.0096	54.5
GA	WRAO	0.0092	50	0.0091	54
	GOWA	0.0099	55	0.0097	58

the best performances respectively. Therefore, in Fig. 4, we visualize the classification error of WRAO with GA method and GOWA with LM in the test phase. According to Fig. 4a and b, we can see both WRAO and GOWA have shown good classification while WRAO shows the best performance for classification and during training. We could suspect that the irregularity of the learning behavior of GOWA may be due the to extra re-ordering of weights step in GOWA.

5.4 Abalone data set

The data set contains 8 classes of 4177 instances each [1]. This problem involves predicting the age of abalone from physical measurements [1]. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope, which is a very time consuming task. Therefore, other measurements, which are easier to obtain, are used to predict the age [1]. Table 2 shows the results of the WRAO and GOWA learning.

The Levenberg–Marquardt learning of MSE decreases by 1% when GA learning is applied for WRAO and it show 2% of MSE decrease for the of testing phase of WRAO. Further, sum of fuzzy classification error (SYCLE) of test phase also has reduced error for the GA results of WRAO. However, according to Table 3, GOWA continues its irregular learning pattern for the Abalone data as well. That is, GOWA has increased MSE and sum of fuzzy classification error (SYCLE) for both testing and training pases during GA optimization compared to the Levenberg–Marquardt optimization.

For the test phase of the Abalone data prediction experiment between WRAO and GOWA the GA and LM methods

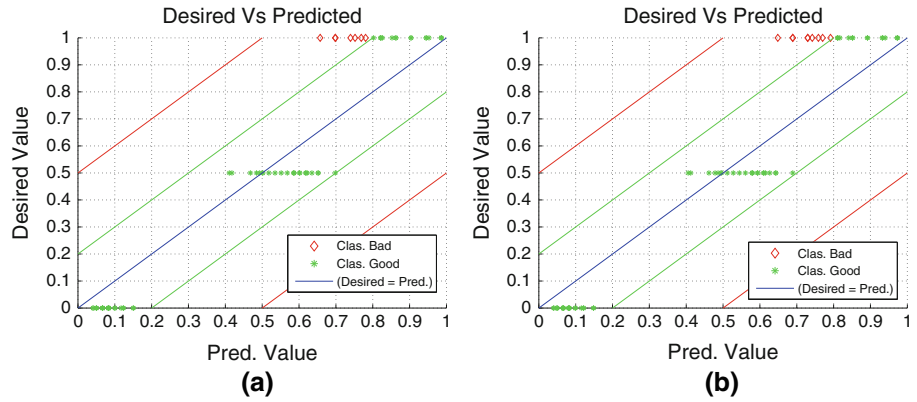

Fig. 4 Iris: SYCLE for WRAO and GOWA. **a** WRAO: GA test; **b** GOWA: Levenberg–Marquardt test

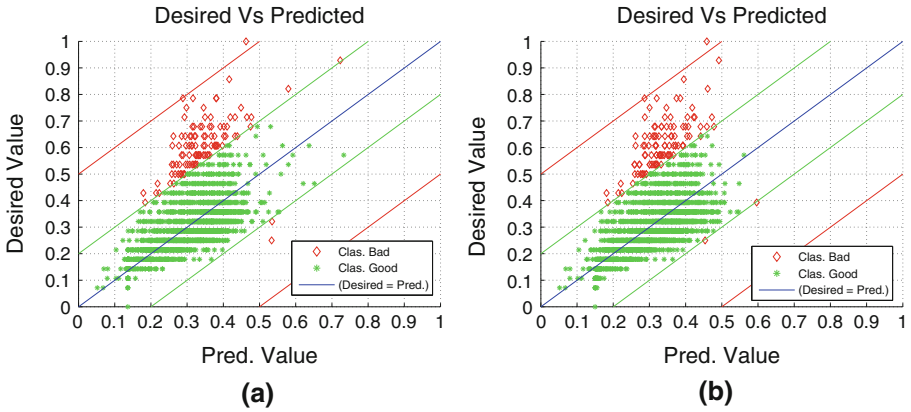

Fig. 5 Abalone: SYCLE for WRAO and GOWA. **a** Abalone: GA test; **b** Abalone: Levenberg–Marquardt test

give the best performances respectively. Therefore, in Fig. 5, we visualize the classification error of WRAO with GA method and GOWA with LM in the test phase. According to Fig. 5a and b, we can see both WRAO and GOWA have shown similar classification output pattern while WRAO shows slightly better performance for classification.

We continued the experiment for other data sets in the UCI data repository but do not detail the results here. For all of them, WRAO learning is consistent, ie. the global optimizer has reduced error compared to the local optimizer's error, while GOWA shows an irregular learning pattern across all data sets. Further, for all experiments, WRAO has shown the best performance during the learning and testing phases.

6 Conclusions

We discussed Levenberg–Marquardt and Genetic algorithm based algorithms for both aggregation and weights extraction in WRAO and GOWA from real data samples. We conducted the experiment and described the results for 3 data sets, namely Wine, Iris, and Abalone, in UCI data repository. For all experiments, WRAO has shown the best performance during the learning and testing phases. During the experiment, for all data sets WRAO learning is consistent, ie. the global optimizer has a reduced error compared to the local optimizer's error, while GOWA shows an irregular learning pattern across all data sets. Mostly, it has less error for the local optimizer compared to the global optimizer. One hypothesis would be that the irregularity of the learning behavior of GOWA may be due to the extra re-ordering of weights. The second could be that, by making the search space harder with the extra additivity property of GOWA, it could restrict the capabilities of heuristic type learning methods. Therefore, it is worth investigating these two issues further in the future.

References

1. Asuncion A, Newman D (2007) UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html
2. Beliakov G (2005) Learning weights in the generalized owa operators. Fuzzy Optim Decision Making 4(2):119–130
3. Beliakov G, Warren J (2001) Appropriate choice of aggregation operators in fuzzy decisionsupport systems. Fuzzy Syst IEEE Trans 9(6):773–784
4. Ben-Arieh D (2005) Sensitivity of multi-criteria decision making to linguistic quantifiers and aggregation means. Comput Indus Eng 48(2):289–309
5. Bordogna G, Pasi G (2005) Personalised indexing and retrieval of heterogeneous structured documents. Inform Retriev 8(2):301–318
6. Botzheim J, Kóczy LT, Ruano AE (2002) Extension of the levenberg–marquardt algorithm for the extractionof trapezoidal and general piecewise linear fuzzy rules. In: Fuzzy systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE international conference on, vol. 1
7. Dennis J Jr, Gay D, Walsh R (1981) An adaptive nonlinear least-squares algorithm. ACM Trans Math Softw 7(3):348–368
8. Dubois D, Prade H (1986) Weighted minimum and maximum operations in fuzzy set theory. Info Sci 39(2):205–210
9. Dyckhoff H, Pedrycz W (1984) Generalized means as model of compensative connectives. Fuzzy Sets Syst 14(2):143–154
10. Filev D, Yager R (1998) On the issue of obtaining owa operator weights. Fuzzy Sets Syst 94(2):157–169
11. Fletcher R (1987) Practical methods of optimization. Wiley, New York
12. Fodor J, Marichal J, Roubens M (1994) Characterization of some aggregation functions arising from mcdm problems. Proc IPMU 94:1026–1031
13. Fodor J, Marichal J, Roubens M (1995) Characterization of the ordered weighted averaging operators. Fuzzy Syst IEEE Trans 3(2):236–240
14. Gill P, Murray W, Wright M (1981) Practical optimization. Academic Press, London
15. Goldberg D (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley Longman Publishing Co., Reading
16. Haupt R, Haupt S (2004) Practical genetic algorithms. Wiley, London
17. Herrera F, López E, Mendaña C, Rodríguez M (2001) A linguistic decision model for personnel management solved with a linguistic biobjective genetic algorithm. Fuzzy Sets Syst 118(1):47–64

18. Holland J (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor
19. Kacprzyk J, Wilbik A, Zadrozny S (2007) Linguistic summaries of time series via an OWA operator based aggregation of partial trends. In: IEEE international fuzzy systems conference. FUZZ-IEEE 2007, pp 1–6
20. Koza J (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge
21. Levenberg K (1944) A method for the solution of certain nonlinear problems in least squares. Q Appl Math 2:164–168
22. Manna S, Mendis BSU, Gedeon TD (AUG 2009) Hierarchical document signature: a specialized application of fuzzy signature for document computing. FUZZ-IEEE 2009 international conference on Fuzzy systems, pp 1–6
23. Marquardt D (1963) An algorithm for least-squares estimation of nonlinear parameters. J Soc Indus Appl Math 11(2):431–441
24. Mendis BSU (2008) Fuzzy signatures: Hierarchical fuzzy systems and alpplications. Ph.D. thesis, College of Engineering and Computer Science, The Australian National University, Australia
25. Mendis BSU, Gedeon TD (2008) Aggregation selection for hierarchical fuzzy signatures: a comparison of hierarchical owa and wrao. In: International conference of information processing and management of uncertainty in knowledge based systems (IPMU), pp 1–8
26. Mendis BSU, Gedeon TD (2008) A comparison: Fuzzy signatures and choquet integral. In: IEEE world congress on computational intelligence, WCCI, Hong Kong, pp 1464–1471
27. Mendis BSU, Gedeon TD, Botzheim J, Kóczy LT (2006) Generalised weighted relevance aggregation operators for hierarchical fuzzy signatures. In: International conference on computational inteligence for modelling control and automation and international conference on intelligent agents web technologies and international commerce (CIMCA'06). Sydney, Australia, pp 198–203
28. Mendis BSU, Gedeon TD, Kóczy LT (2006) Learning generalized weighted relevance aggregation operators using levenberg–marquardt method. In: Sixth international conference on hybrid intelligent systems (HIS'06) and 4th conference on neuro-computing and evolving intelligence (NCEI 06'). New Zealand, pp 1–6
29. Meuth R, Lim M, Ong Y, Wunsch D (2009) A proposition on memes and meta-memes in computing for higher-order learning. Memetic Comput 1(2):85–100
30. Miller J, Potter W, Gandham R, Lapena C (1993) An evaluation of local improvement operators for genetic algorithms. IEEE Trans Syst Man Cybernet SMC 23:1340–1340
31. Mor J (1977) The levenberg–marquardt algorithm: implementation and theory. Lecture Notes Math 630:105–116
32. Srinivas M, Patnaik L (1994) Genetic algorithms: a survey. Computer 27(6):17–26
33. Torra V, Godo L (2002) Continuous WOWA operators with application to defuzzification. Aggregation operators: new trends and applications, p 159
34. Wong K, Gedeon T, Kóczy L (2004) Construction of fuzzy signature from data: an example of sars pre-clinical diagnosis system. Fuzzy systems, 2004. In: Proceedings of 2004 IEEE international conference on, vol 3
35. Yager R (1988) On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Trans Syst Man Cybernet 18(1):183–190
36. Yager R (2004) Generalized owa aggregation operators. Fuzzy Optim Decision Making 3(1):93–107
37. Zhu D, Mendis BSU, Gedeon TD, Asthana A, Goecke R (2008) A hybrid fuzzy approach for human eye gaze pattern recognition. In: 15th international conference on neural information processing (ICONIP08), vols. 1–8. Auckland, New Zealand