IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 2, 2018, accepted August 3, 2018, date of current version September 5, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2865135

Analysis of Data Sets With Learning Conflicts for Machine Learning

SERGIO LEDESMA, MARIO-ALBERTO IBARRA-MANZANO, EDUARDO CABAL-YEPEZ, (Member, IEEE), DORA-LUZ ALMANZA-OJEDA, AND JUAN-GABRIEL AVINA-CERVANTES

Department of Electrical and Computer Engineering, School of Engineering, University of Guanajuato, Salamanca 36885, Mexico

Corresponding author: Sergio Ledesma (selo@ugto.mx)

This work was supported in part by DAIP and in part by the University of Guanajuato.

ABSTRACT In supervised learning, a machine learning system requires a data set. In occasions, however, the data set may have learning conflicts that may drastically affect the performance of the learning system. This paper presents a method to analyze the learning conflicts in a data set. Several computer simulations to test and validate our method are performed. Two common functions in the field of optimization are used to create clean data sets. The data sets are, then, contaminated with random data, and the total learning conflict level for each case is computed. The proposed algorithm is used to identify the learning conflicts that are intentionally inserted. Next, an artificial neural network is trained and evaluated using the contaminated data set. The algorithm proposed in this paper is used in a real-world application to detect problems in a data set for a refrigeration system. It is concluded that the algorithm can be used to improve the performance of machine learning systems.

INDEX TERMS Data set, conflict level, conflict removal, machine learning, target value.

I. INTRODUCTION

Machine learning allows computational systems to adapt, and therefore, improve their performance with experience accumulated from observed data [1]. Frequently, machine learning is used to build predictive models by extracting patterns from large data sets. These models are used in predictive data analytics applications including: price prediction, risk assessment, predicting customer behavior, mechanism design, traffic surveillance and document classification, see [2], [3]. On the other hand, machine learning may be used in computing tasks where the design or the programming of the required algorithms is difficult or impossible [4]. In other cases, machine learning must be able to update while the training set is changing, see [5]. Despite its practical and commercial successes, machine learning remains a young field with many under explored research opportunities [6]. Machine learning is divided in three main categories: supervised learning, unsupervised learning and reinforcement learning [7]–[9]. The goal of supervised learning is to model a conditional distribution which for many simple regression problems is chosen to be Gaussian. However, practical machine learning problems can often have significantly non-Gaussian distributions [10]. Upon completion of the supervised learning process, the hope is that the artificial and real outputs will be close enough to be useful for all sets of inputs likely to be encountered in practice [11]. In unsupervised learning, algorithms "experience" a data set containing many features, then learn useful properties of the structure of this data set [12]. In fact, unsupervised learning is often performed as part of an exploratory data analysis [13]. Reinforcement learning agents have achieved some successes in a variety of domains [14]. However, their applicability has previously been limited to domains in which useful features can be handcrafted, or to domains with fully observed low-dimensional state spaces [15]. Despite the number of works in the field of machine learning, just a very scanty number of works are related, somehow, to the subject of finding learning conflicts in data sets.

Definition 1: Let $\mathbb{R}$ be the set of all real numbers. We denote the Cartesian product, $\mathbb{R} \times \mathbb{R} \times \cdots \mathbb{R}$ by $\mathbb{R}^M$. Thus, a data set for supervised machine learning, represented as $[\mathbf{X}|\mathbf{t}]$, is a set of N examples used for learning where

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1M} \\ x_{21} & x_{22} & \cdots & x_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{NM} \end{bmatrix} \tag{1}$$

is the data set input, and

$$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}, \tag{2}$$

is the data set target. Thus, example i in the data set is composed of an input $\mathbf{x}_i \in \mathbb{R}^M$ and a target value $t_i \in \mathbb{R}$.

Definition 2: Let $\mathbb{R}$ be the set of all real numbers. Thus, if $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$, a machine learning system f from $\mathbb{R}^M$ to $\mathbb{R}$ is a subset of $\mathbb{R}^M \times \mathbb{R}$ such that for every $\mathbf{x} \in \mathbb{R}^M$ there is one and only one $y \in \mathbb{R}$ (i.e., there is a unique $y \in \mathbb{R}$) such that $(\mathbf{x}, y) \in f$, see [16]. Additionally, if there is a data set, such that for each example in a data set, the input value $\mathbf{x} \in \mathbb{R}^M$ must produce a target value $t \in \mathbb{R}$. Then, the problem is to find a machine learning system such that, for every $\mathbf{x} \in \mathbb{R}^M$, the output of the system y is as close as possible to the desired output t, see [17].

Definition 3: Let $[\mathbf{X}|\mathbf{t}]$ be a data set for supervised machine learning where each example has an input value $\mathbf{x} \in \mathbb{R}^M$ and a target value $t \in \mathbb{R}$. Thus, there is a learning conflict when a machine learning system from $\mathbb{R}^M$ into $\mathbb{R}$ is not able to produce an output $y \in \mathbb{R}$ that is close to the desired output t, for every $\mathbf{x} \in \mathbb{R}^M$.

II. NORMALIZATION

Because the range of values in a data set may broadly vary, objective functions used in machine learning will typically not work well without some sort of normalization, [18], [19]. In the same sense, algorithms to detect learning conflicts may not perform properly without normalization. In this section, one method to normalize the input and the target of the data set is discussed. Note that this method is commonly used for most machine learning techniques.

A. INPUT NORMALIZATION

The simplest method to normalize a data set is rescaling its values to a different range. The following text describes how to scale the data to the range of [0 1]. Suppose that a_i is the minimum value of column i of $\mathbf{X}$ in Equation 1, that is,

$$a_i = \min(x_{1i},\ x_{2i},\ \ldots\ x_{Ni}) \qquad i = 1, 2, 3, \ldots M \tag{3}$$

and suppose that b_i is the maximum value of column i of $\mathbf{X}$,

$$b_i = \max(x_{1i},\ x_{2i},\ \ldots\ x_{Ni}) \qquad i = 1, 2, 3, \cdots M \tag{4}$$

then, the normalized input can be defined as

$$\mathbf{Z} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1M} \\ z_{21} & z_{22} & \cdots & z_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ z_{N1} & z_{N2} & \cdots & z_{NM} \end{bmatrix} \tag{5}$$

where

$$z_{ij} = \frac{x_{ij} - a_j}{b_j - a_j}. \tag{6}$$

B. TARGET NORMALIZATION

In order to be able to provide an analysis that does not depend on the range of the target, it is very important to normalize the target values in the data set. Thus, the normalized target τ_i is defined as

$$\tau_i = \frac{t_i - \min(t_1,\ t_2,\ \ldots, t_N)}{\max(t_1,\ t_2,\ \ldots, t_N) - \min(t_1,\ t_2,\ \ldots, t_N)} \tag{7}$$

where $i = 1, 2, \ldots, N$ and $0 \leq \tau_i \leq 1$.

III. LEARNING CONFLICTS

There is a learning conflict when two (or more) input samples that are very similar have different target values. For some applications, these learning conflicts may not be very significant, however, these conflicts may interfere with the overall performance of the system for other applications. In the next subsection, it is discussed how to measure the similarity between two input samples in a data set.

A. NORMALIZED INPUT DIFFERENCE

One simple method to compute the degree of similarity between two input samples is by using the Euclidean distance of the normalized input $\mathbf{Z}$ of Equations 5 and 6. Thus, suppose that δ_{ij} is the normalized input difference between input sample i and input sample j. Therefore, δ_{ij} must be zero when both input samples are equal, δ_{ij} must be close to zero when both input samples are similar, and δ_{ij} must be close to one when both samples are completely different. Then, δ_{ij} can be defined as

$$\delta_{ij}^2 = \frac{1}{M} \sum_{k=1}^{M} (z_{ik} - z_{jk})^2 \tag{8}$$

where $\delta_{ij} = \delta_{ji}$. δ_{ij} is the normalized Euclidean distance between the two input samples, and therefore, $0 \leq \delta_{ij} \leq 1$. For learning conflict analysis, it is necessary to compute the value of δ_{ij} for each possible combination of sample input pairs; these values can be organized in matrix form as

$$\mathbf{D} = \begin{bmatrix} 0 & \delta_{12} & \cdots & \delta_{1N} \\ \delta_{21} & 0 & \cdots & \delta_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{N1} & \delta_{N2} & \cdots & 0 \end{bmatrix}, \tag{9}$$

observe that $\mathbf{D}$ is a hollow symmetric matrix (a symmetric matrix with zeros in its main diagonal). Observe that matrix $\mathbf{D}$ provides a direct method to locate similar input samples. For instance, a threshold value can be defined to extract similar rows in a data set, in this case any value in $\mathbf{D}$ that is less than a specified threshold value will help identify those sample rows that are similar. Once a metric to express the degree of similarity at the input has been defined, it is necessary to have a corresponding metric for the target difference, this is the subject of the next sub-section. Before moving to the next subsection, note that there may be other methods that can be used to compute the degree of similarity between samples. For instance, López-Soto *et al.* [20] compare the performance

of a pattern-based classification with a classification based on the Euclidean distance.

B. NORMALIZED TARGET DIFFERENCE

The target of the data set includes the values that are desired at the output of a machine learning system. Under normal conditions, similar inputs should have similar target values, however, in some cases this is not true. The normalized target difference expresses the degree of difference in the target values between two samples, and it is defined as

$$T_{ij} = \left|\tau_i - \tau_j\right| \qquad i, j = 1, 2, \ldots N \tag{10}$$

where τ is the normalized target of Equation 7 and $0 \leq T_{ij} \leq 1$. Figure 1 shows the value of T_{ij} as a function of two target values: (τ_i and τ_j). The blue region represents the values of T_{ij} that are close to zero, and therefore, it is produced when both normalized target values are similar. On the contrary, zones in red, magenta or black represents a high target difference, that is they are produced when both target values (τ_i and τ_j) are very different. As learning conflicts are produced only when the target values are different, zones that are not blue may represent a learning conflict depending on the similarity of the input values.

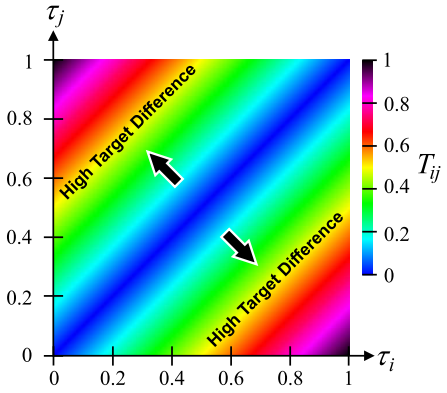

FIGURE 1. Normalized target difference.

C. NORMALIZED LEARNING CONFLICT LEVEL

As it was previously indicated, there is a learning conflict when the normalized input difference is close to zero, $\delta_{ij} \approx 0$, while the normalized target difference is not zero, $T_{ij} \neq 0$. For most applications, it is convenient and easy to combine the values of δ_{ij} and T_{ij} in one single value that represents the conflict level between two samples. One straightforward method to measure the learning conflict level c_{ij} is

$$c_{ij} = T_{ij} W(\delta_{ij}) \tag{11}$$

where $W(\delta_{ij})$ is a weighting function with

$$0 \leq W(\delta_{ij}) \leq 1.$$

For this type of application, the weighting function $W(\delta_{ij})$ should have a peak at $\delta_{ij} = 0$, and diminish to zero as δ_{ij} increases. A possible choice for the weighting function $W(\delta_{ij})$ is the Gaussian function

$$W(\delta_{ij}) = \alpha \exp\left[-\frac{(\delta_{ij} - \beta)^2}{2\sigma^2}\right] \tag{12}$$

for any arbitrary real constants: α, β and σ. In this case, choosing the Gaussian function has nothing to do with assumptions of normal distributions in the data set. The Gaussian function is simply a well behaved function that can be easily computed. There are, however, several other functions that can be used for weighting purposes. The parameter α in Equation 12 is the maximum amplitude of the function, in this study, this parameter is set to one in order to obtain a learning conflict level that is normalized; remember that T_{ij} is a normalized variable in the range [0 1]. The Gaussian function is centered at its maximum value, β, for this specific case, it should be zero. Therefore, Equation 12 can be written as

$$W(\delta_{ij}) = \exp\left(-\frac{\delta_{ij}^2}{2\sigma^2}\right). \tag{13}$$

The parameter σ in Equation 13 controls the width of the bell. More information about how to estimate an appropriate value for this parameter will be provided in Subsection III-D.

In order to have a direct expression to compute the normalized learning conflict level, Equation 10 can be used in Equation 11 to get

$$c_{ij} = \left|\tau_i - \tau_j\right| W(\delta_{ij}) \tag{14}$$

then, applying Equation 13 in the previous equation, we get

$$c_{ij} = \left|\tau_i - \tau_j\right| \exp\left(-\frac{\delta_{ij}^2}{2\sigma^2}\right) \tag{15}$$

and finally, Equation 8 can be used to obtain a direct expression for the conflict level

$$c_{ij} = \left|\tau_i - \tau_j\right| \exp\left[-\frac{1}{2M\sigma^2}\sum_{k=1}^{M}(z_{ik} - z_{jk})^2\right]. \tag{16}$$

Figure 2 displays the learning conflict value c_{ij} between case i and case j using a color scale when $\tau_i = 0.25$ which is an arbitrary value chosen only for illustrative purposes. Observe that for other values of τ_i, the figure is very similar, the only difference is that the vertical blue zone is shifted to the right or to the left. In this figure, low learning conflicts levels are indicated by a blue color while high learning conflict values are shown in red, magenta and black. The graph is divided in two main parts: the top part and the bottom part. The top part of the graph includes values for δ_{ij} that are bigger than 0.025, in this part both inputs are different, and therefore, the learning conflict c_{ij} is small as it is indicated by the blue color. On the other hand, when δ_{ij} is less than 0.02 in Figure 2, then both inputs are very similar, and consequently, the learning conflict level is strongly influenced by the target difference, $\left|\tau_i - \tau_j\right|$. Figure 3 displays the effect of the target values τ_i and τ_j on the learning conflict level when

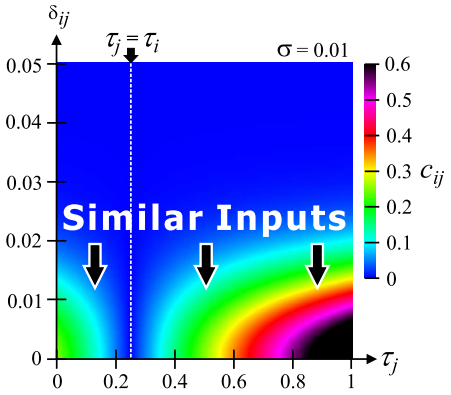

FIGURE 2. Conflict level and input similarity when $\tau_i = 0.25$.

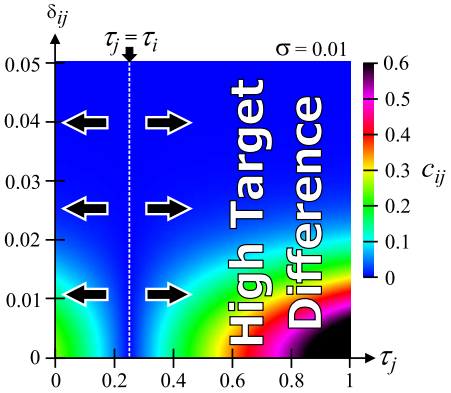

FIGURE 3. Conflict level and target difference when $\tau_i = 0.25$.

$\tau_i = 0.25$, which is the same value used in Figure 2. The area around the vertical dot line describes a zone where the target difference is close to zero, that is, $\tau_j \approx \tau_i$, and consequently, the conflict level is minimum in the vicinity of this line as it is indicated by the blue color in this zone. As the value of τ_j moves away from τ_i, the target difference gets bigger. In fact, the target difference increases when $\tau_j < \tau_i$ as well as when $\tau_j > \tau_i$ making the learning conflict level depend on the input similarity in this region. Figure 4 shows the value of the learning conflict level c_{ij} as a function of the normalized target value τ_j and the normalized input difference δ_{ij} when $\sigma = 0.01$ and $\tau_i = 0.25$. From this figure, it can be seen that high learning conflict levels are only produced when both the normalized input difference δ_{ij} is less than 0.02 and the target difference is bigger than 0.4. That is a high learning conflict level is produced when both inputs are similar while the target values are different.

In practice, computing and storing the values for c_{ij} using Equation 16 for all values of $i = 0, 1, \ldots N$ and $j = 0, 1, \ldots N$ can be very difficult, or in some cases, impossible when hardware resources are limited. However, for most applications it is necessary to store only the greatest values of c_{ij}. To do this, it is possible to use a multi-set structure such as the one included in the C++ Standard Template library (STL). A **multiset** in the STL library is a container that store elements following a specific order; a **multiset** is typically

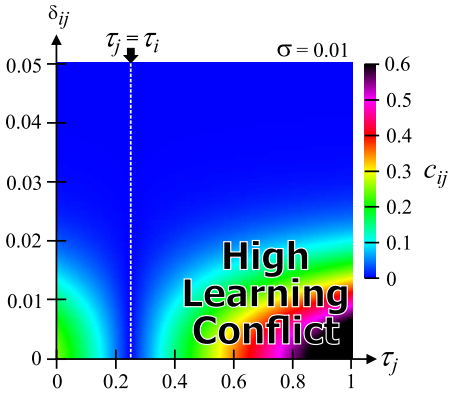

FIGURE 4. Conflict level c_{ij} for $\sigma = 0.01$ and $\tau_i = 0.25$.

implemented as binary search tree. When a value of c_{ij} is computed, it is inserted in the **multiset**; if the number of elements in the container exceeds some pre-established value, an element is deleted from the beginning of the **multiset**. As elements are sorted, the deleted element is always the learning conflict with the smallest value at the moment. This approach, of course, has some disadvantages but it provides good results in most cases while keeping memory requirements low.

D. WEIGHTING FUNCTION WIDTH

The width of the bell of the weighting function of Equation 12 depends on the value of σ. Figure 5 displays the shape of the weighting function for four values of σ. From this figure, it can be seen that a small σ produces a narrow weighting function. For most applications, the value of σ should be small enough so that the weighting function quickly falls down, however, more research is needed to explore other methods to compute the optimum value for σ. Thus, the main feature of the weighting function is to reduce the conflict level when two input samples are not very similar. If the bell of the weighting function is too width, many samples in the data set will have a high conflict level even though these samples present no conflict at all. An easy method to estimate the width of the bell can be estimated by solving for σ in Equation 13

$$\sigma = \frac{\delta_{ij}}{\sqrt{-2\ln(W_{ij})}} \tag{17}$$

where $0 \leq \delta_{ij} \leq 1$ and $0 < W_{ij} < 1$. If there is a maximum value in the weighting function $W_{\max}$ that it is acceptable at a specified normalized input distance $\delta_{\max}$, then the optimum value of σ can be estimated using

$$\sigma_{optimum} = \frac{\delta_{\max}}{\sqrt{-2\ln(W_{\max})}}. \tag{18}$$

In practice, a typical value for $\delta_{\max}$ is 0.05 when $W_{\max} = 0.05$; which means that when two input samples (sample i and sample j) have a normalized input difference of 0.05, then the weighting function attenuates the target conflict by 95 %. Figure 6 displays the behavior of the learning conflict level

when the normalized input difference δ_{ij} and the normalized target value τ_j are changed with $\sigma = 0.02$. By comparing Figure 4 with Figure 6, it can be observed that the value of σ affects the learning conflict level when $\tau_j \neq \tau_i$; that is, σ controls the learning conflict sensitivity when there is a problem in the target values. It is recommended to set $\sigma < 0.02$ in order to avoid high learning conflict levels for normal variations in the target values.

E. TOTAL LEARNING CONFLICT LEVEL

The first step in the process to solve learning conflicts in a data set is to create a list of those cases with the highest conflict levels. This list may include the index values i and j for the two conflicting cases and the conflict level c_{ij}. Once two conflicting cases have been identified from this list, there are several approaches that can be used to solve this specific conflict. In this study, we propose some methods to solve this learning conflicts. One possible method is to delete from the data set the two conflicting cases. Other possible approach is to replace the two conflicting cases with the average of them. One final method is to delete only one of the conflicting cases. In all these cases, knowing how the data set was created may provide important information that can be used to solve a learning conflict.

Another more convenient method to solve a learning conflict is by computing the total learning conflict level

$$C_i = \frac{1}{N-1} \sum_{\substack{j=1 \\ i \neq j}}^{N} c_{ij}. \tag{19}$$

Note that $c_{ij} = 0$ when $i = j$, and therefore, the total learning conflict is averaged over $N - 1$ samples (instead of N samples), and $0 \leq C_i \leq 1$ for $i = 1, 2, \ldots N$. However, in practice many values of c_{ij} are very small and, therefore, the total learning conflict level C_i does not reach a value of one. In fact, C_i is closer to zero than to one. The main advantage of using C_i instead of the individual conflict values $c_{i1}, c_{i2}, \ldots, c_{iN}$ to solve a learning conflict is that C_i includes the effects of all conflicts produced by this specific sample data. However, it is important to notice that the total learning conflict, C_i, does not contain the same information stored

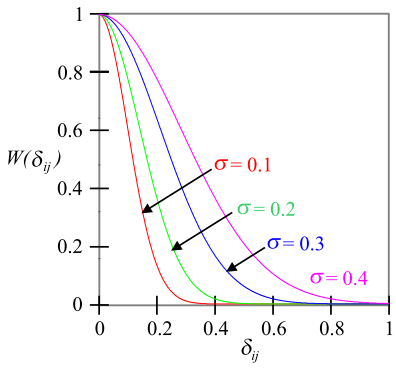

FIGURE 5. Weighting function.

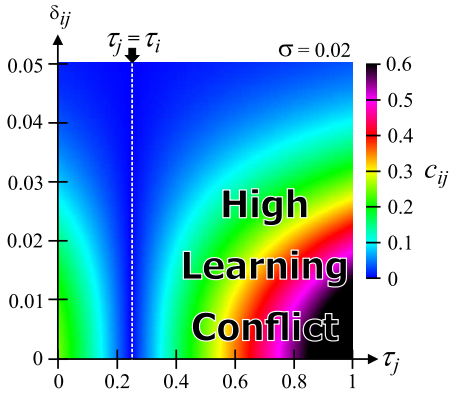

FIGURE 6. Conflict level c_{ij} for $\sigma = 0.02$.

in the individual conflict values $c_{i1}, c_{i2}, \cdots, c_{iN}$. In fact, in order to solve a conflict, it is better to analyze the values of $c_{i1}, c_{i2}, \ldots, c_{iN}$ individually. Consider for instance a data set where a single sample has conflicts with many cases; this learning conflict can be easily solved by removing this specific case.

In the computer simulations of the next section, learning conflicts are solved by deleting those cases that have the highest total conflict level C_i. Note, however, that more research in needed to choose individually the best method to solve each learning conflict.

IV. COMPUTER SIMULATIONS

This section presents some computer simulations to evaluate the performance of the total learning conflict level C_i of Equation 19. Two common functions used in the field of optimization will be used: Booth's function and McCormick function.

A. BOOTH'S FUNCTION

Booth's function is a continuous, differentiable, non-separable, non-scalable, and unimodal function. This function has been used in the area of engineering design and optimization, see [21] and [22]. Booth's function is defined as

$$f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2. \tag{20}$$

The experiment, in this case, began by building a data set with 2500 cases. The data set has two input variables (x_1 and x_2) and one output, $f(x_1, x_2)$. The values of x_1 and x_2 are randomly generated using a uniform probability distribution with values in the range from -10 to 10; the output value is computed using Equation 20. The data set, then, is contaminated by inserting 10 cases with random values in the same range as the original data set. Next, Equation 19 is used to compute the values of $C_1, C_2, \cdots, C_{2510}$, that is, the learning conflict for each case is computed. In order to identify the learning conflicts, a threshold value is set; any case in the data set with a total learning conflict C_i bigger than the threshold is classified as a learning conflict. Because most of the conflict

values fall in two different ranges, it is possible to set the threshold value just by inspecting the conflict level values. However, more research is needed to estimate the optimum value of the learning conflict threshold. Table 1 shows the resulting confusion matrix for the experiment; as it can be seen, the proposed algorithm is able to correctly classify the 10 learning conflicts. Table 2 shows the confusion matrix when the data set is contaminated with 20 random cases; from the 20 learning conflicts, the algorithm correctly classified 17 of the learning conflicts. Table 3 shows the confusion matrix when the number of random samples is 30, in this cases 6 normal cases are incorrectly identified as learning conflicts and only one real learning conflict is not identified.

TABLE 1. Confusion matrix for Booth's function with 10 conflicts.

	Booth's function	Random Values
Booth's function	2500	0
Random Values	0	10

TABLE 2. Confusion matrix for Booth's function with 20 conflicts.

	Booth's function	Random Values
Booth's function	2500	0
Random Values	3	17

TABLE 3. Confusion matrix for Booth's function with 30 conflicts.

	Booth's function	Random Values
Booth's function	2494	6
Random Values	1	29

In order to validate the method proposed in this paper, an artificial neural network, ANN, with two inputs, one output and one hidden layer is implemented. The activation function used in the neural network was the hyperbolic tangent. The number of neurons in the hidden layer of the network is adjusted to avoid over fitting. The multi-layer network with 8 neurons in the hidden layer was trained using a hybrid method. This training method consisted of simulated annealing with 100 temperatures, 100 iterations per temperature with a linear cooling schedule. The training was, then, improved using the conjugate gradient method with 1000 epochs. The contaminated data set is randomly splitted to create the training set and the validation set; 80% of the cases are used for training and the remaining 20% are used for validation. Table 4 shows the resulting root-mean-square error (RMSE) for training and for validation. The simulation began by training the ANN using the contaminated training set, then, the RMSE for training and the RMSE for validation are computed using the data set with the learning conflicts. Next, the algorithm proposed in this paper is used to identify possible learning conflicts in the contaminated data set; any identified learning conflict is removed from the contaminated data set to produce a clean data set. Finally, another ANN is trained and validated using the clean data set. The RMSE for training and the RMSE for validation are, then, computed. Table 4 shows the results for this computer simulation and

TABLE 4. RMSE before and after removing the learning conflicts for Booth's function.

	Training		Validation	
# Conflicts	With conflicts	Without conflicts	With conflicts	Without conflicts
10	96.0	0.971	66.0	0.978
20	95.8	4.59	56.5	1.57
30	141	3.73	112	1.26

for different number of learning conflicts: 10, 20 and 30. Consider the first row in Table 4, in this case the RMSE for training using the contaminated data set is 96.0; after removing the identified learning conflicts, the RMSE for training is reduced to 0.971. A drastic reduction in the RMSE is also observed during the validation, in fact, a reduction from 66.0 to 0.978 is observed. A similar behavior is noticed when the number of learning conflicts in the data set is 20 and 30.

TABLE 5. Confusion matrix for Booth's function with 20 conflicts and 5% of noise.

	Booth's function	Random Values
Booth's function	2498	2
Random Values	3	17

TABLE 6. Confusion matrix for Booth's function with 20 conflicts and 10% of noise.

	Booth's function	Random Values
Booth's function	2488	12
Random Values	3	17

B. BOOTH'S FUNCTION WITH NOISE

Table 5 shows the confusion matrix for Booth's function with 20 conflicts, but in this case, the original data was contaminated with 5% of noise. The noise was generated using a uniformly distributed random generator. By comparing the results from Table 2 with the results in Table 5, it can be seen when the data set has noise, the proposed method performs a total of five misclassifications. In fact, from the 2500 normal cases, the method incorrectly classified two of these cases as learning conflicts. Table 6 shows the confusion matrix for Booth's function but in this case the data was contaminated with 10% of noise. From this table, it can be observed that when the noise level increases the number of misclassifications also increases. However, it can be seen that the presence of noise affects mainly the classification of normal data samples. That is, under the presence of noise, some regular samples are now classified as learning conflicts.

C. MCCORMICK FUNCTION

The McCormick's function is a convex, multimodal and differentiable function, [21]. This function is defined as

$$f(x_1, x_2) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1. \tag{21}$$

The computer simulations began by building a data set with 2500 cases; the data set had two input variables (x_1 and x_2) and one output computed using the McCormick's function of Equation 21. The data set is built generating uniformly distributed random values for x_1 and x_2 in the range from -3 to 3. Then, the data set is contaminated by inserting 10 cases with random values, and Equation 19 is used to compute the learning conflict values: C_1, C_2, $\cdots$, C_{2510}. Table 7 shows the confusion matrix for this computer simulation; in this case zero classification errors are made. Table 8 and 9 show the classification results when the data set is contaminated with 20 and 30 random cases respectively. From inspecting Table 8, it can be observed that the algorithm incorrectly classified 1 case out of the 20 conflicts when there are 20 learning conflicts in the data set. Similarly, the algorithm makes 3 misclassifications when the data set has 30 learning conflicts, however, in this case one regular case is incorrectly classified as a learning conflict.

TABLE 7. Confusion matrix for McCormick's function with 10 conflicts.

	McCormick's function	Random Values
McCormick's function	2500	0
Random Values	0	10

TABLE 8. Confusion matrix for McCormick's function with 20 conflicts.

	McCormick's function	Random Values
McCormick's function	2500	0
Random Values	1	19

TABLE 9. Confusion matrix for McCormick's function with 30 conflicts.

	McCormick's function	Random Values
McCormick's function	2499	1
Random Values	2	28

In order to test the validity of our method, an ANN is trained and validated using the contaminated data set. Later, the algorithm proposed in this paper is used to identify possible learning conflicts, and then, remove these conflicts. Another ANN is trained and validated using the data set without the learning conflicts. Table 10 presents the values of the RMSE for training and for validation before and after removing the learning conflicts. From this table, it can be observed that the proposed algorithm reduces the RMSE for both training and validation. For instance, consider a data set with 10 learning conflicts, that is the first row in Table 10, the RMSE for validations is reduced from 1.16 to 0.0530 after removing the conflicts.

D. CONFLICTS AND THE NUMBER OF INPUTS

This section illustrates the effectiveness of the method to find learning conflicts when the number of input M changes. In this case, the simulations are based on Equation 22 which is a sine function in multi-dimensions. The simulation began by building a data set using random values for $x_1, x_2, \ldots x_M$ in the range from -4 to 4 and computing the target value using

TABLE 10. RMSE before and after removing the learning conflicts for McCormick's function.

	Training		Validation	
# Conflicts	With conflicts	Without conflicts	With conflicts	Without conflicts
10	1.32	0.0437	1.16	0.0530
20	1.84	0.0558	1.59	0.0527
30	2.46	0.106	1.18	0.0881

Equation 22. The number of cases N is computed assuming that 30 points are necessary for each dimension. Note that this number of points is enough to approximately represent a sine wave. After the data set is created, the data set is contaminated with random values at the input and at the target. The results of the simulation are displayed in Table 11. The first row at this table displays the simulation results for a data set with two inputs $M = 2$. In this case, two learning conflicts are introduced in the original data set, and the method proposed in this paper is able to detect both learning conflicts. The second row in Table 11 shows the results for a data set with three inputs $M = 3$; this data set is contaminated introducing 68 learning conflicts and the proposed method detects 58 of these conflicts resulting in an effectiveness of 85%. The last row in Table 11 includes the simulation results for a data set with four inputs; $M = 4$; from the 2025 learning conflicts that are introduced, 1725 are detected resulting in an effectiveness of 85%.

$$f(x_1, x_2, \ldots x_M) = \frac{1}{2}\sum_{i=1}^{M}\left\{[1-(\cos\pi)^i]\cos x_i + [1-(\cos\pi)^{i-1}]\sin x_i\right\}. \tag{22}$$

Figure 7 shows the effectiveness of the proposed method to find learning conflicts in the multivariable function of Equation 22. The simulation was performed using three inputs ($M = 3$) and $N = 30^3$ cases. As it can be seen from Figure 7, the proposed method has an effectiveness of 85% when the data set has approximately 60 learning conflicts. From the same figure, it can also be observed that when the number of learning conflicts is bigger than 100 the effectiveness of the method is approximately 70%.

E. REAL-WORLD APPLICATION

The algorithm proposed in this paper is used to find (and then remove) learning conflicts in a data set for a refrigeration system an considerably improve the performance of the model, see [23]. Three artificial neural networks were designed to model individually three typical energy parameters:

TABLE 11. Learning conflicts found when *M* changes.

#inputs M	#cases N	Introduced learning conflicts	Learning conflicts found	Effectiveness
2	30^2	2	2	100%
3	30^3	68	58	85%
4	30^4	2025	1725	85%

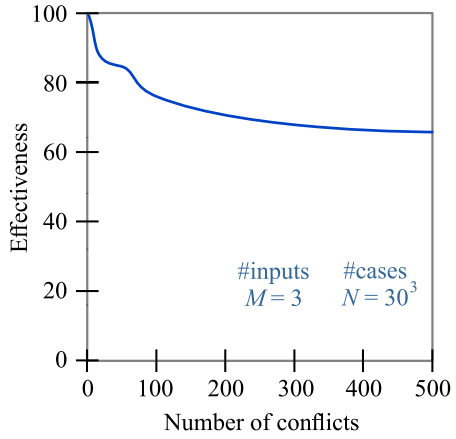

FIGURE 7. Effectiveness of the proposed method.

the cooling capacity, the power consumption and the coefficient of performance. The training of the multi-layer neural networks was based on the temperature of the evaporator and the condenser. From the validation results of the model, it can be concluded that our algorithm may also be used in real-world applications to detect and correct learning conflicts. Note, however, that the proposed algorithm has some limitations. For instance, the algorithm may not work properly in functions with discontinuities or when the data set has very few samples for some specific input values.

F. NORMALIZATION TO ZERO-MEAN AND UNIT-VARIANCE

In all previous computer simulations, the input values and the target values have been scaled to the range of [0, 1]. However, it is well known that this method of normalization is very sensitive to outliers, and therefore, other normalizing methods can be considered. In this subsection, the normalization to zero-mean and unit variance is considered.

TABLE 12. Confusion matrix for Booth's function with 5% of noise and normalization to zero-mean and unit-variance.

	Booth's function	Random Values
Booth's function	2500	0
Random Values	3	17

Table 12 shows the confusion matrix for Booth's function, defined in Equation 20. For this simulation, the data set has 20 learning conflicts and 5% of random noise, however, the normalization method used for this table is zero-mean and unit-variance. By comparing the confusion matrix of Table 5 with the confusion matrix of Table 12, it can be seen that the normalization to zero-mean and unit-variance made 3 misclassification, while the normalization to [0 1] made 5 misclassifications. From these two tables, it can also be observed that the reduction in misclassifications occurs in samples that do not have learning conflicts. Table 13 shows the confusion matrix also for Booth's function, but in this case the data set has 10% of noise. From this table, it can be seen that the normalization to zero-mean and unit-variance

TABLE 13. Confusion matrix for Booth's function with 20% of noise and normalization to zero-mean and unit-variance.

	Booth's function	Random Values
Booth's function	2494	6
Random Values	2	18

produces a total of 8 misclassifications, while the normalization to [0 1] produces a total of 15, see Table 6. Again, it can be seen that the reduction in misclassification was mainly on those samples that do not have a learning conflict.

This paper presents a method to compute the learning conflict for each case in a data set. The algorithm can be used to detect possible learning conflicts between two cases in the data set, or to compute the learning conflict for each case individually. The algorithm can be used to reduce or to eliminate these learning conflicts and improve the performance of systems based on machine learning techniques. This includes deep learning techniques which allow computational models to learn representations of data with multiple levels of abstraction [24], [25].

V. CONCLUSIONS

This paper proposes a method that can be used to find learning conflicts in data sets used for machine learning. In supervised learning, the data set has two parts: the input and the target. A learning conflict is produced when two or more similar inputs produce different target values. In order to detect possible learning conflicts in a data set, the first step is to normalize the input and the target values. Second, the normalized input difference and the normalized target difference must be computed; these two values are then combined using a weighting function to estimate the learning conflict between two specific cases in the data set. In order to solve learning conflicts, a method to estimate the total learning conflict for each case in the data set is proposed. For instance, to improve the performance of a machine learning system, those cases with the highest learning conflict level can be removed. To test the validity of our method, some data sets are build using common functions in the field of optimization. These data sets are, then, contaminated by inserting cases with random data. The proposed method is used, thus, to identify these cases and the respective confusion matrix is computed for validation purposes. Additionally, an artificial neural network is trained using the contaminated data set and the data set obtained after removing the learning conflicts; it is concluded that the performance of the network improved considerably when the the learning conflicts are removed. Finally, the proposed algorithm is used to improve the performance of the model of a refrigeration system in a real-world application.

REFERENCES

[1] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*. New York, NY, USA: AML, 2012.

[2] J. D. Kelleher, B. M. Namee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. Cambridge, MA, USA: MIT Press, 2015.

[3] W. Liu, M, Zhang, Z. Luo, and Y. Cai, "An ensemble deep learning method for vehicle type classification on visual traffic surveillance sensors," *IEEE Access*, vol. 5, pp. 24417–24425, 2017.
[4] D. Koller, and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques* (Adaptive Computation and Machine Learning Series). London, U.K.: MIT Press, 2009.
[5] B. Jin, Z. Jing, and H. Zhao, "Incremental and decremental extreme learning machine based on generalized inverse," *IEEE Access*, vol. 5, pp. 20852–20865, 2017.
[6] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.
[7] S. Marsland, *Machine Learning: An Algorithmic Perspective* (Chapman, and Hall/CRC Machine Learning, and Pattern Recognition), 2nd ed. Boca Raton, FL, USA: CRC Press, 2015.
[8] K. P. Murphy, *Machine Learning* (A Probabilistic Perspective). Cambridge, MA, USA: MIT Press, 2012.
[9] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. London, U.K.: Pearson, 2016.
[10] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
[11] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Data Mining, Inference, and Prediction). New York, NY, USA: Springer-Verlag, 2009.
[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning Series). London, U.K.: MIT Press, 2016.
[13] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R* (Springer Texts in Statistics). New York, NY, USA: Springer-Verlag, 2013.
[14] E. Alpaydin, *Machine Learning: The New AI* (The MIT Press Essential Knowledge Series). London, U.K.: MIT Press, 2016.
[15] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.
[16] A. N. Michel, and C. J. Herget, *Applied Algebra and Functional Analysis*. New York, NY, USA: Dover, 1981.
[17] T. Masters, *Deep Belief Nets in C++ and CUDA, Restricted Boltzmann Machines*. Lexington, KY, USA: Masters, 2015.
[18] L. Bo, L. Wang, and L. Jiao, "Feature scaling for kernel fisher discriminant analysis using leave-one-out cross validation," *Neural Comput.*, vol. 18, no. 4, pp. 961–978, 2006.
[19] S. Theodoridis, and K. Koutroumbas, *Pattern Recognition*, 4h ed. New York, NY, USA: Academic, 2009.
[20] D. López-Soto, S. Yacout, and F. Angel-Bello, "Root cause analysis of familiarity biases in classification of inventory items based on logical patterns recognition," *Comput., Ind. Eng.*, vol. 93, pp. 121–130, Mar. 2016.
[21] J. Momin and Y. Xin-She, "A literature survey of benchmark functions for global optimization problems," *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013.
[22] R. V. Rao and V. J. Savsani, *Mechanical Design Optimization Using Advanced Optimization Techniques*. London, U.K.: Springer-Verlag, 2012.
[23] J. M. Belman-Flores, A. Mota-Babiloni, S. Ledesma, and P. Makhnatch, "Using ANNs to approach to the energy performance for a small refrigeration system working with R134a and two alternative lower GWP mixtures," *Appl. Thermal Eng.*, vol. 127, pp. 996–1004, Dec. 2017.
[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[25] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

SERGIO LEDESMA received the M.S. degree from the University of Guanajuato, Mexico, while he was involved in the setup of Internet in Mexico, and the Ph.D. degree from the Stevens Institute of Technology, Hoboken, NJ, USA, in 2001. After graduating, he was with Barclays Bank as part of the ITHR Group. He was a Software Engineer for several years. He is the Creator of the Software Neural Lab, Wintempla, and TexLab Neural Lab. He is currently a Research Professor with the University of Guanajuato. His areas of interests are artificial intelligence and software engineering.

MARIO-ALBERTO IBARRA-MANZANO received the B.Eng. degree in communication and electronic engineering and the M.Eng. degree in electric from the University of Guanajuato, Salamanca, Mexico, in 2003 and 2006, respectively, and the Ph.D. degree (Hons.) from the Institut National des Sciences Appliquées, Toulouse, France. He is currently an Assistant Professor with the Electronics Engineering Department, University of Guanajuato. He is interested in digital design on FPGA for image processing applied on autonomous robots and real-time systems.

EDUARDO CABAL-YEPEZ (M'08) received the M.Eng. degree from the Faculty of Electrical and Electronic Mechanical Engineering, University of Guanajuato, Mexico, in 2001, and the Ph.D. degree from the University of Sussex, U.K., in 2007. In 2008, he joined the Engineering Division, Campus Irapuato-Salamanca, University of Guanajuato, where he is currently a Leading Professor and the Dean of the Multidisciplinary Studies Department. He has authored/co-authored of over 31 research papers published in journals, registered in the journal citation report. His current research interests are digital instrumentation, digital image and signal processing, artificial intelligence, robotics, smart sensors, real-time processing, mechatronics, FPGAs, power quality, and embedded systems, among others. He is an active member of the IEEE Instrumentation and Measurement Society, the IEEE Industrial Electronics Society, and the IEEE Signal Processing Society. He serves as an Area Editor for the international journal *Computers and Electrical Engineering* (Elsevier) and an Associate Editor for the international journal *Digital Signal Processing* (Elsevier) and the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT.

DORA-LUZ ALMANZA-OJEDA received the B.S. degree in electronics engineering and the M.S. degree in electrical engineering from the University of Guanajuato, Salamanca, Mexico, in 2003 and 2005, respectively, and the Ph.D. degree from the University of Toulouse III: Paul Sabatier, Toulouse, France, in 2011. She is currently an Assistant Professor with the Electronics Engineering Department, University of Guanajuato. She is interested in embedded vision for controlling autonomous robots and real-time systems.

JUAN-GABRIEL AVINA-CERVANTES received the B.Eng. degree in electronics and communications engineering and the M.Eng. degree in electrical engineering (instrumentation and digital systems) from the University of Guanajuato in 1998 and 1999, respectively, and the Ph.D. degree in informatics and telecommunications from the National Polytechnic Institute of Toulouse and LAASCNRS, France, in 2005. He is currently a Researcher and a full-time Professor with the University of Guanajuato. His research interests include artificial vision for outdoor mobile robotics, pattern recognition, optimal control systems, and image processing.