Journal of Intelligent & Fuzzy Systems xx (20xx) x–xx
DOI:10.3233/JIFS-162297
IOS Press

Impact of fuzziness categorization on divide and conquer strategy for instance selection

Rana Aamir Raza Ashfaq[a,b] and Xi-Zhao Wang[a,*]
[a]*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China*
[b]*Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan*

Abstract. Fuzziness based divide and conquer (D&C) is a recently proposed strategy for promoting the classifiers (i.e., fuzzy classifiers) performance, where the amount of fuzziness quantity associated with each data point (i.e., both labeled and unlabeled) is considered as an important avenue to the empire for instance selection problem. This technique is regarded as a semi-supervised learning (SSL) technique, where different categories of instances are obtained by using fuzziness measure, and then the instances having less amount of fuzziness are incorporated into training set for improving the generalization ability of a classifier. This study proposes some effective methods and presents a novel algorithm for categorizing the instances into three groups that can effectively integrate with D&C strategy. It is observed by the experimental validation that considering the splitting criteria for instances categorization can lead the classifier to perform better on withheld set. Results on different classification data sets prove the effectiveness of proposed algorithm.

Keywords: Instance selection, fuzziness, divide and conquer, generalization, fuzziness categorization, splitting methods

1. Brief introduction of divide-and-conquer strategy

Fuzziness based divide and conquer (D&C) strategy is proposed by Wang et al. [28] for improving the generalization capability of a classifier $\mathscr{F}$ (i.e., the classifiers whose output is a membership or fuzzy vector v). In [28], fuzziness is considered as an important criterion for dividing the instances x_i into three groups, and then a group having low fuzziness FG_{low} is incorporated into training set L. $\mathscr{F}$ is retrained on the new training set and obtained results are simulated by the experimental verification. Their strategy is regarded as an approach to semi-supervised learning (SSL), where the unlabeled instances having certain magnitude of fuzziness participate in the learning process and provide better predictive ability on withheld set [5].

In their study, they conducted two different experiments to observe the relationship among the fuzziness F of a classifier and the classification accuracy of X. Specifically, in *first experiment*, the relationship between correctly classified instances x_{ci} and their fuzziness f_{ci} is observed. For a set of labeled instances $D = (x_i, y_i)_{i=1}^{n}$, where training set L and testing set T are obtained by randomly splitting the D with the proportion of 70% & 30%. A classifier $\mathscr{F}$ whose output is analogous to v is chosen to perform the following steps as listed below.

1. $\mathscr{F} = \mathscr{F}(L)$ (i.e., Train a classifier).
2. Obtained a membership matrix of L by using $\mathscr{F} : U_L = (\mu_{ij})_{C\times L}$.
3. Obtained a membership matrix of $T : U_T = (\mu_{ij})_{C\times T}$.

*Corresponding author. Xi-Zhao Wang, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China. Tel./Fax: +86 13700326811; E-mail: xzwang@szu.edu.cn.

4. Fuzziness of every instance f_i in $F(U_L)$ and $F(U_T)$ is computed.
5. Sorting is performed both in $F(U_L)$ and $F(U_T)$ based on $(f_i)_{i=1}^{n}$.
6. Three groups i.e,. FG_{Low}, FG_{mid} and FG_{High} are extracted based on (sorted) fuzziness values of the instances belonging to L and T.
7. Obtained the accuracies on FG_{Low}, FG_{mid} and FG_{High} that are extracted by using L and T.

Authors in [28] designed this experiment to observe the correct rate of classification for three groups, where they experimentally showed that the instances belong to FG_{low} have higher classification rate than the FG_{mid} and FG_{high}. To further analysis and support this fact, they conducted a *second experiment* where some useful information i.e., *how fuzziness intacts with the classified and misclassified instances x_{mi}?* is extracted. In their second experiment, above mentioned steps (i to iv) were common, but instead of the dividing the instances into three groups, they performed the following steps on L and T.

1. Separate the correctly classified instances x_{ci} along with their fuzziness values f_{ci} both in L and T.
2. Compute Average fuzziness of all correctly classified instances $(f_i)_{i=1}^{c}$.
3. Compute Average fuzziness of misclassified instances $(f_i)_{i=1}^{m}$.

It is also found by the experimental verification in their studies that the instances having higher fuzziness values (i.e., that are grouped in FG_{high}) have greater chance of misclassification, because those instances are near to the *classification boundary*, while the instances having low fuzziness are *far* from the boundary. In both cases they repeated their experiments multiple times to check the effectiveness of randomization.

The steps (i.e., i to vi) in above mentioned experiment were the basic assumptions toward the fuzziness based D&C strategy, but the second experiment is conducted to figure out and prove the assumption that fuzziness has an essential impact on the classifier's generalization ability (i.e., specially for the misclassified samples). It is also proved in their study that the risk of misclassification becomes higher as the fuzziness of instances increases in L.

Therefore, for the most classification problem, it is difficult to obtain the correct rate of classification for those instances that are having high fuzziness as compared to the low fuzziness instances. In other words, it is difficult the handle the boundary points compared to the points that are far from the boundary (i.e., *inner boundary*).

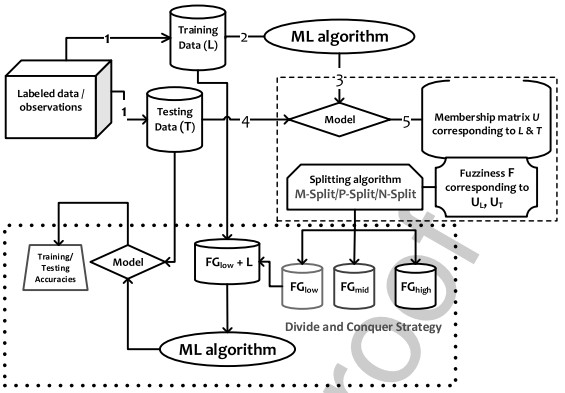

Fig. 1. Flow chart of Experiment-1 reflecting fuzziness based categorization.

The assumption behind D&C strategy is to *separately handling the instances* that are included in FG_{low}, FG_{mid} and FG_{high} categories and to *incorporate the group with highest accuracy* into L, which is an effective way to promote the classifier performance. The flow chart of *D&C* strategy proposed by Wang et al. [28] is depicted in Fig. 1

One can see the Fig. 1, where the FG_{low} is incorporated into L and retraining is performed with new training set to achieve the better classification accuracy on withheld dataset T. From the literatures [4, 30], one can also find the studies related to the generalization capability (i.e., prediction) of a classifier that rely on the fuzziness. An important question arises: *how to effectively divide the training or unlabeled instances into FG_{low}, FG_{mid} and FG_{high} that can guarantee the distribution of useful instances for promoting the D&C strategy?*. We present some states of the art mechanisms and provide a novel algorithm to divide the instances into three different categories based on their fuzziness magnitude. We also compare the impact of these categorization on the generalization ability by using a classifier called neural network with random weight (NNR_w), and observe their effectiveness after integrating it with D&C strategy.

The rest of the paper is organized as follows. Section 2 discusses some IS methods that rely on the uncertainties. Section 3 presents the basic concept of fuzziness and also illustrates how fuzziness can be computed for a single layer feed forwarded network(SLFN) called neural network with random

weight (NNR_w), whose output can be obtained like a fuzzy or membership vector. Section 4 presents some states of the art approaches for splitting the instances into three groups, and propose an algorithm for instance categorization. Experimental verification and results are presented in Section 5. Finally, Section 6 concludes this paper.

2. Instance selection process

We consider instance selection as an essential process in the data reduction phase of knowledge discovery and data mining (KDD), whose aim is to reduce the amount of training instances from original data. In this process noisy instances may also be removed. Therefore, this process reduces the size of training set either by retaining the predictive ability or by improving the learning capability. According to [23], several instances participate in learning or training process, but some instances are irrelevant or not useful for classifying, hence, to achieve an acceptable learning performance, it is possible to ignore these non-useful instances, however, this process is considered as *instance selection*. In general, IS methods, that focus on improving the predictive performance of the classifier (apply after instance selection) are called *edition techniques*. Methods that contribute to the reduction of storage requirements are known as *condensation algorithms*. Some IS methods achieve both goals (i.e., generalization capability and storage reduction) simultaneously, they are called *hybrid methods*. In [6], authors described that the expectation to achieve the accuracy either equal or better than the original training set is not usually achieved, and still a certain *loss of accuracy* is inevitable. For the better selection of instances, many other methods have been proposed in the recent literatures. Authors in [13] proposed a novel instance selection mechanism called LAMIS which employs the hyperplane with a large symmetric margin. In their approach, the core of instance selection process is based on keeping the hyperplane that separates the two-class data to provide the large margin separation. LAMIS selects the most informative instances, satisfying both objectives i.e., high accuracy and reduction rates. In [33], an instance reduction method is proposed to speed up the instance selection process for the various instance selection-based multiple-instance learning algorithms. Their method is based on pairwise similarity between instances in a training bag, where the performance can be enhanced by improving the similarity between the instances that are necessary for learning. A detail review of instance selection methods is presented in [23].

Many instance selection methods that rely on uncertainties have been proposed in the scientific literatures, e.g., author in [3] proposed the first instance selection mechanism that queries an instance into the uncertainty area. The method proposed by [3] learns a concept by reducing the volume of uncertainty area using the required instance. This method evades to query those instances that exist in learned area, but it queries only those instances which reside in uncertainty area. This process increases the learning rate because it avoids to acquire the ineffective instances. Another method called *Query by Committee* (QBC) is proposed by Seung et al. [27] in 1992, which acquires the instances or examples according to the principle of *maximal disagreement* of the committee. This method is based on the observation that an instance with maximum disagreement is harder to classify. Therefore, this type of disagreement is considered as a type of uncertainty. Authors in [18] and [19] also proposed the uncertainty based instance selection strategy. Their strategies build a single classifier that could predict and provide the class label to an instance, and also a measurement of certainty. The uncertainty is associated with *posterior probability* using Bayes rule. It only selects the instance which is considered to be misclassified, because the class of the instance is unknown before asking to the domain experts. Wang et al. [29] introduced a new instance selection mechanism called *maximum ambiguity based sample selection in fuzzy decision tree induction*, where the instances are selected based on the principle of *maximal classification ambiguity* to select the instances with maximal evaluated ambiguity in *fuzzy decision tree induction*. It only selects the instance with maximal evaluation ambiguity when it has similarity with fuzzy decision tree.

3. Neural network with random weight (NNR_w) and fuzziness

3.1. Neural network with random weight (NNR_w)

Schmidt et al. [26] are the first one, who earlier studied and investigated the significance of randomization on the generalization performance of SLFN. Authors in [26] experimentally demonstrated that SLFN can gain a better predictive performance by selecting the random weights that connect the input

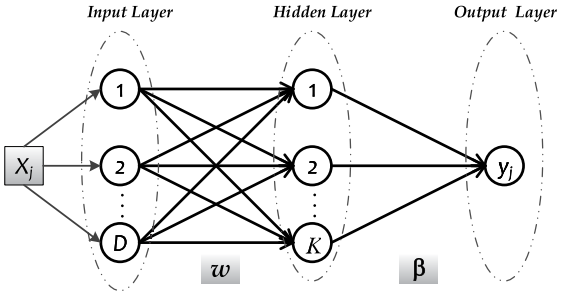

Fig. 2. A schematic overview of NNR_w model.

layer and hidden layer nodes, and by analytically computing the weights of output layer nodes. This was the first study regarding the non-iterative training of neural network (NN) using randomization concept. The researchers also concluded that, in SLFN, the weights of the output layer nodes are significantly most important than the weights found in the hidden layer nodes. Authors in [26] did not propose the name of their SLFN, therefore, in order to recognize their work, we use the neural network with random weights (NNR_w). An overview of the structure of NNR_w is shown in Fig. 2.

The idea of randomization of hidden layer in NN has been proposed several times. Pao et al. [24] proposed a very similar model called random vector functional-link network (RVFLN) and its generalization performance was investigated in [15]. Authors in [31] used this approach for initializing the weights of NN before training it with back-propagation (BP). From the literature [2, 15, 25], one can study that several ideas have been proposed for randomization that incorporate random hidden-layer weights and biases, and the direct connection between the input layer and the output layer. In NNR_w, the parameters of hidden layer nodes (i.e, input weights and hidden layer biases) can be chosen randomly and the output weights between hidden and output layer can be analytically determined with *Moore-Penrose generalized inverse*. Similarly, authors in [7, 14, 36, 37], introduced NN with a randomly initialized hidden layer and trained using pseudo-inverse. NNR_w provides better training speed, because unlike BP (i.e., gradient-descent based algorithm), NNR_w does not require the iterative tunning process for parameters at hidden layer nodes, that overcomes the drawback of local minimal as in conventional gradient based algorithms. Conventional neural networks have great approximation capability but the behavior of those networks during the training process heavily depends on the training set. The classification boundaries generated by the NNs are often unpredictable in the presence of less amount of data. Many extensions related to NNs have been proposed in scientific literatures such as discrete-time stochastic neural network [17], polygonal fuzzy NN used to handle the polygonal fuzzy data [20] and weight networks [16].

NNR_w algorithm. The key idea of NNR_w is the random initialization of the hidden layer weights and the subsequent training consists of computing the least-squares solution to the linear system defined by the outputs of hidden layer and targets.

Consider a set of l distinct instances (x_i, y_i) with $x_i \in \mathbb{R}^l$, and $y_i \in \mathbb{R}$. The output of SLFN with hidden layer nodes $\tilde{\text{K}}$ can be represented as

$$f(\mathbf{x}) = \sum_{i=1}^{\tilde{\text{K}}} \tilde{\beta}_i g(\mathbf{w}_i, b_i, \mathbf{x}), \mathbf{x} \in \mathbb{R}^l \qquad (1)$$

In Equation (1), $\mathbf{w}_i \in \mathbb{R}^l$ and $b_i \in \mathbb{R}$ represent the random parameters (i.e., input weights and biases respectively) at hidden layer nodes, $\tilde{\beta}_i \in \mathbb{R}^m$ is the output weight and $\tilde{\beta}_i g(\mathbf{w}_i, b_i, \mathbf{x})$ is the output of *ith* hidden node w.r.t input $\mathbf{x}$.

Therefore, for a given dataset $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l} \subset \mathbb{R}^l \times \mathbb{R}^m$, where $\mathbf{x}_i$ is an input vector, and $\mathbf{y}_i$ is the corresponding observed vector. SLFN with $\tilde{\text{K}}$ hidden nodes approximating these n training instances with zero error means that their exist $\tilde{\beta}_i$, $\mathbf{w}_i$, and b_i where $i = 1, \cdots, \text{L}$ such that

$$\sum_{i=1}^{\tilde{\text{K}}} \tilde{\beta}_i g(\mathbf{w}_i, b_i, \mathbf{x}_j) = \mathbf{y}_j, \; j = 1, \cdots, l \qquad (2)$$

The above Equation (2) can be compactly written as

$$\mathbf{H}\tilde{\beta} = \mathbf{Y} \qquad (3)$$

where

$$\mathbf{H}_{l \times \tilde{\text{K}}} = \begin{pmatrix} g(\mathbf{w}_1, b_1, \mathbf{x}_1) & g(\mathbf{w}_2, b_2, \mathbf{x}_1) & \cdots & g(\mathbf{w}_{\tilde{\text{K}}}, b_{\tilde{\text{K}}}, \mathbf{x}_1) \\ g(\mathbf{w}_1, b_1, \mathbf{x}_2) & g(\mathbf{w}_2, b_2, \mathbf{x}_2) & \cdots & g(\mathbf{w}_{\tilde{\text{K}}}, b_{\tilde{\text{K}}}, \mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ g(\mathbf{w}_1, b_1, \mathbf{x}_l) & g(\mathbf{w}_2, b_2, \mathbf{x}_l) & \cdots & g(\mathbf{w}_{\tilde{\text{K}}}, b_{\tilde{\text{K}}}, \mathbf{x}_l) \end{pmatrix},$$

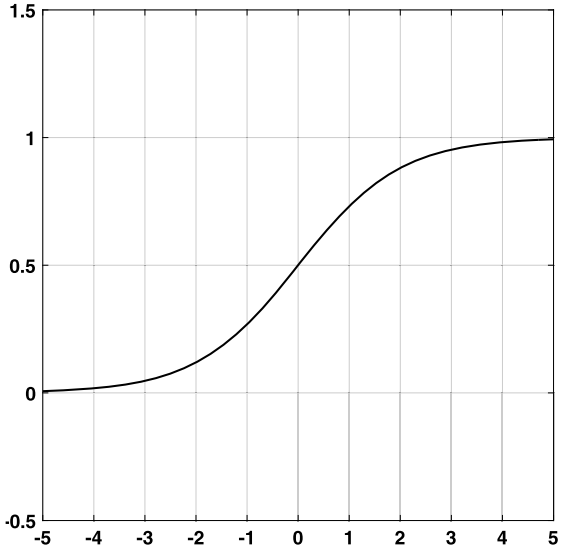

Fig. 3. Sigmoid activation function.

$$\tilde{\beta}_{\tilde{\mathrm{K}}\times m} = \begin{pmatrix} \tilde{\beta}_1 \\ \tilde{\beta}_2 \\ \vdots \\ \tilde{\beta}_{\tilde{\mathrm{K}}} \end{pmatrix}^T$$

and

$$\mathrm{T}_{l\times m} = \begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \vdots \\ \mathbf{t}_l \end{pmatrix}^T$$

$\mathbf{H}$ is a hidden layer output matrix with respect to the input vectors x_i, where $i = 1, \cdots, l$, and $g(z)$ is a *sigmoid activation function*. A *sigmoid activation* or *cost function* uses the sigmoid function to determine its activation and it can be defined as

$$g(z) = \frac{1}{1 + e^{-z}} \tag{4}$$

A typical sigmoid function is presented in Fig. 3, which has the curve in two direction and resemblances to the *English letter* "S". This function transforms an input value to an output ranging from 0 to 1. It is worth noting that this function only returns the positive values. If one needs the NN to return the negative values then this function will be unsuitable.

The above mentioned (3) becomes a system of linear equations, which in most cases can be transferred to a regular system of linear equations.

$$\mathbf{H}^T\mathbf{H}\tilde{\beta} = \mathbf{H}^T\mathrm{Y} \tag{5}$$

Suppose that $\mathbf{H}^T\mathbf{H}$ is non-singular, the solution of system according to Equation (5) can be expressed as

$$\tilde{\beta} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathrm{Y} = \mathbf{H}^{\dagger}\mathrm{Y}. \tag{6}$$

In Equation (6), $H^{\dagger}$ denotes the pseudo-inverse. The (NNR$_w$) algorithm is summarized in Algorithm 1.

Algorithm 1 Illustration of NNR$_w$ algorithm

Require:

$L = \{(\mathbf{x}_i, \mathbf{y}_i) | \mathbf{x}_i \in \mathbb{R}^l, \mathbf{y}_i \in \mathbb{R}^m, i = 1, \cdots, l\}$
Hidden node output function $g(\mathbf{w}, b, \mathbf{x})$
Number of hidden nodes $\tilde{\mathrm{K}}$

Ensure:

Weight Matrix $\tilde{\beta}$
Basic steps:
Randomly select the input parameters $\mathbf{w}_i$ and b_i where $i = 1, \cdots, \tilde{\mathrm{K}}$
Compute the hidden layer output matrix $\mathbf{H}$.
By using Eq. (6), calculate the output weight $\tilde{\beta}$

The proposed solution to the equation $\mathbf{H}\tilde{\beta} = Y$ in the NNR$_w$ algorithm, as $\tilde{\beta} = \mathbf{H}^{\dagger}Y$ has following characteristics making it an attractive solution.

1. Minimum training error can be achieved due to provision of the least-squares solutions.
2. It is considered as the solution with the smallest norm among the least-squares solutions
3. The smallest norm solution among the least-squares solutions is unique for a $\mathbf{H}$, and represented as $\tilde{\beta} = \mathbf{H}^{\dagger}Y$.

The strength of the NNR$_w$ is the fact that there is no need to iteratively tune of the randomly initialized weights as compared to BP. Due to this advantage (i.e., non-iterative) this process makes its learning speed extremely fast.

3.2. Theory of fuzzy set

The theory of fuzzy set is introduces by Zadeh [34] and it relates to classes of objects with un-sharp or unclear boundaries where the membership degree is a significant matter of interest. Conventional crisp sets contain values that only satisfy precise or compact characteristics required for membership.

Suppose a set S consists of values from 1 to 5 is a representation of a crisp set then it can be expressed as

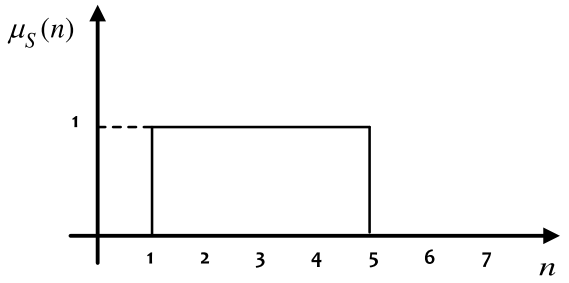

Fig. 4. Membership function of crisp set S.

$$S = \{n \in \Re\}$$

where $\Re$ represents a set of real numbers. Hence S can be modeled by its membership function $F_S(n) : \Re$. A membership function is also called *characteristic function* or *indicator function*

$$F_S(n) = \begin{cases} 1 & (1 \leq n \leq 5 \\ 0 & (else) \end{cases} \tag{7}$$

The above membership function will produce a value 1, if and only if $S \in n$. Otherwise, it will produce a value 0. A graph of membership function for crisp set S is depicted in Fig. 4 One can visualize the above Fig. 4, that shows a clear distinction between the elements either these belong to the set or not. It is worth noted that there is a problem in defining the numbers between 6 to 10, which are also crisp. A similar situation comes to our daily life. For example, in deciding, the person is either tall or short. Regarding the conventional phenomenon or logic, there must be need to define a height threshold that differentiates a tall person to the smaller one. For example, A person is taller, if its height is greater than the threshold (i.e., 6 feet) otherwise that person is shorter. However, this mechanism obviously can not reflect the actual judgment based on human thinking. So, it can be better described as soft switching mechanism rather than threshold. This may lead us to often add some *modifiers* to the word taller or shorter (i.e., very, more, not, not very, somehow etc.) in order to express the degree of height rather than absolute value i.e., either *True* or *False*.

A *fuzzy set*, enables us to make decision according to human thinking. Therefore, in a fuzzy set, "height of a person", degree related the height is defined that provides us a *continous* transition rather than a sharp transition (i.e., from true to false).

3.3. Fuzziness of a fuzzy set

The term fuzziness refers to the unclear boundary between two linguistic variables and firstly proposed by Zadeh [34] with the proposed concept of *fuzzy set*. According to Dubois and Prade [11], A fuzzy set is a set that contains the elements with varying the membership quantity or values. The elements of a fuzzy set are mapped to a universe of membership values by using a function (i.e, membership function), and it maps the elements of objects of a fuzzy set into the *real* values in the interval of [0, 1] [1]. Fuzzy set is different from classical crisp set where the elements in a set have full membership which means that membership value must be equal to 1.

Zadeh in [35] also generalized the probability measure of an event to a fuzzy event and suggested using entropy in *information theory* to interpret the uncertainty associated with the fuzzy event [30]. Authors in [9], considered fuzziness to be a type of uncertainty and defined fuzzy entropy which is based on Shannon's entropy function, and also introduced the set of properties for which a fuzziness should satisfy. These properties of fuzziness have widely accepted and become a criteria for defining the fuzziness [1]. Those properties also depict that a fuzziness degree attains its*maximum* when the membership degree of every element is equal and *minimum* when every elements either belongs to fuzzy set or absolutely not.

We consider fuzziness as a type of *cognitive uncertainty*, coming from the transition of uncertainty from one linguistic variable to another. where a linguistic variable is a fuzzy set and defined in a certain universe of discourse. Let $S = \{\mu_1, \mu_2, \cdots, \mu_n\}$ be a fuzzy set then the fuzziness of S can be defined as

$$F(S) = -K \sum_{i=1}^{n} (\mu_i \log \mu_i + (1 - \mu_i) \log(1 - \mu_i)) \tag{8}$$

In Equation (8), μ_i represents the membership function and K is a constant and equal to $(1/n)$.

The fuzziness of fuzzy set defined by Equation (8) attains its maximum when the membership degree of every element is $\mu_i = 0.5$ for every $i = 1, 2, \cdots, n$ and minimum when every element belongs to the fuzzy set or absolutely not for every $\mu_i = 0$ or $\mu_i = 1$, $i(1 \leq i \leq n)$.

3.4. Fuzziness of training/testing instances

Now we associate fuzziness with the output of a classifier. It is well found that many classifiers have

the output in the form of fuzzy vector, where each component of vector corresponds to the membership degree of the testing instances belonging to a class. These types of classifiers include artificial neural network (ANN) [12, 22], support vector machines (SVMs) [8, 10, 32], fuzzy decision tree [29] and etc. It is important to mention here that some classifiers, such as NN, *a simple transformation can transfer the initial output to a form a fuzzy vector if the components of the initial output are not in the interval of* [0, 1].

Therefore, for a given set of training instances $x_i{}_{i=1}^{n}$, a fuzzy partition of these examples assigns the membership degrees of every example to the class C. Hence, the fuzzy partition can be described by a membership matrix as shown in below Equation (9)

$$U = (\mu_{ij})_{C\times n} \tag{9}$$

In Equation (9), (μ_{ij}) represents the *jth* instances of x belonging to *ith* class. The elements in the membership matrix have to obey the following properties

$$\sum_{i=1}^{C} \mu_{ij} = 1$$

where

$$0 < \sum_{j=1}^{n} \mu_{ij} < n$$

and

$$\mu_{ij} \in [0, 1]$$

Therefore, once the training process (of a classifier) completes, U upon n can be obtained. For the *jth* instance of x, the output of a trained classifier is represented as a fuzzy set i.e.,

$$\mu_j = \{\mu_{1j}, \mu_{2j}, \mu_{3j}, \cdots, \mu_{Cj}\}$$

Based on the Equation (8), the fuzziness of a training classifier on x_j can be depicted as Equation (10)

$$F(U) = -\frac{1}{C}\sum_{i=1}^{C}(\mu_{ij}\log\mu_{ij} + (1-\mu_{ij})\log(1-\mu_{ij})) \tag{10}$$

3.5. Fuzziness of a classifier

Once the training process is completed, we can easily obtain the fuzziness of learned classifier. Let the membership vector U of a classifier on n training samples with C classes be $U = (\mu_{ij})_{C\times n}$, the fuzziness associated with classifier is given in Equation (11)

$$F(U) = -\frac{1}{Cn}\sum_{i=1}^{C}\sum_{j=1}^{n} (\mu_{ij}\log\mu_{ij} + (1-\mu_{ij})\log(1-\mu_{ij})) \tag{11}$$

Above Equation (11) illustrates the fuzziness of a trained classifier that has output analogous to a fuzzy vector. (11) also plays an essential role for investigating the *classifier's generalization* based on fuzziness. This equation actually represents the average fuzziness of classifier's output on all training instances or we can say that it is the training fuzziness of the classifier.

The most appropriate representation of classifier fuzziness must be the average fuzziness of entire instance space that includes both the training and testing instances. However the fuzziness of testing samples is generally unknown and for any supervised learning, there is a well acknowledged assumption, that is, the training samples have a distribution identical to the distribution of samples in the entire space. It indicates the reason-ability that we use Equation (11) for the classifier's fuzziness.

4. Splitting approaches for group formation

How to effectively categorize the instances into low, mid and high fuzziness group is a major concern of this research studies. The formation of groups (or categories) in-fact depends on the situation of a specific problem, where different approaches can be utilized about how and what threshold has to set for dividing the instance into 3 categories.

We illustrate two state of the art techniques and present an algorithm that can be used to categorize the instances into 3 categories based on the fuzziness quantity. We illustrate them briefly.

4.1. Percentage split (P-split)

In this technique, a vector $V = \{v_1, v_2, \cdots, v_n\}$, that represents the sorted fuzziness values of respective instances $X = \{x_1, x_2, ...x_n\}$ in ascending order. where n represents the total number of samples and v_i is the value (i.e., fuzziness) corresponds to the i^{th} instance. This is a very simple mechanism, which actually focus the proportion of percentage of those instances that are having lower fuzziness values and higher fuzziness values. For example, if there are 5000 instances in a data set and after obtaining a V, we assign a lp value and hp value for the FG_{low} and FG_{high}. if V is a sorted list and we assign 30%

instance for low category groups and 20% for high category groups than it calculates the total instances in each category as (12).

$$n_{low} = n \times \frac{lp}{100} \tag{12}$$

Hence the FG_{low} will contain the instances $\{x_1, x_2, \cdots, x_{(n_{low})}\}$. Similarly, the total amount of elements in FG_{high} can be computed as (13).

$$n_{high} = n \times \frac{hp}{100} \tag{13}$$

Based on the total number of elements in high groups we obtain the instances that belong to FG_{high} as $FG_{high} = \{x_{(n-n_{high}+1)}, .., x_n\}$. All remaining elements will be included in FG_{mid} category.

4.2. Natural split (N-split)

This is also a simple mechanism, that distributes the entire set into 3-parts. For example, if there are n instances. It will assign the equal distribution of instance to every group by using (14).

$$d = \frac{n}{3} \tag{14}$$

Hence by using (14), the instances for each group can be extracted as (16)

$$\begin{aligned} FG_{low} &= x_1, x_2, \cdots, x_d \\ FG_{mid} &= x_{d+1}, \cdots, x_{2\times d} \\ FG_{high} &= x_{(2\times d)+1}, x_{(2\times d)+2}, \cdots, x_n \end{aligned} \tag{15}$$

The instances must be sorted based on their fuzziness quantity in ascending order before applying the N-split criteria.

4.3. Proposed splitting method

In this technique we present an algorithm that will *auto* extract the instances for FG_{low}, FG_{mid} and FG_{high} categories based on their fuzziness values. We will use the measure i.e., *median* to illustrate this algorithm called M-split. *Median* is the middle value in the set of elements. To compute the median value, the vector V must be arranged in the ascending order first, there are two ways to compute the median depending on the amount of elements in a set. If the total number of elements n in a set V are *even* then the median M_v can be computed as (16).

$$M_v = \frac{\left(\frac{n}{2}\right)^{th} term + \left(\frac{n}{2}+1\right)^{th} term}{2} \tag{16}$$

If the total number of elements n in a set V are *odd* then the median M_v can be obtained by using (17).

$$M_v = \left(\frac{n+1}{2}\right)^{th} term \tag{17}$$

The illustration of M-split is depicted in Algorithm 2, the algorithm first finds the *median* of all values $[1, 2, 3, \cdots, n]$ in a set and place the obtained value into $q1$. Now we have 2 ranges of values (i.e., 2 sets); one is from first element to q_1th element and the other is $q_1th + 1$ to nth element. Again we find the median values of both sets and keep this value in $q1$ and $q2$ respectively. At this stage we create 3 groups i.e., FG_{low}, FG_{mid} *and* FG_{high} and place the elements in these groups as mentioned in (19). We can also use *mean* instead of *median*, but in this study, we are only categorizing the instance based on median.

$$\begin{aligned} FG_{low} &= 1, 2, \cdots, q1 \\ FG_{mid} &= q1+1, q1+2, \cdots, q2 \\ FG_{high} &= q2+1, q2+2, \cdots, n \end{aligned} \tag{18}$$

5. Experimental validation

The splitting techniques are applied on 10 benchmark data sets, that are taken from *UCI machine learning repository* [21] to experimentally acquire the statistical relation between *instances obtained by proposed fuzziness based categorization mechanism* and the their *correct rate of classification*. The data sets which are selected (during the experiments) belong to a wide variety of classification problems, where the number of instances, their classes and types of features differ. The detail of these data sets is summarized in Table 1.

For the purpose of experimental design, we use 10-fold cross validation technique to evaluate the per-

Table 1
List of data sets acquired for experimentation

Data set	Instances	Input features	Classes
Automobiles	159	15	6
Autompg	392	5	3
Cleveland	297	5	5
Ecoli	336	5	8
Glass	214	9	6
Penbased	10992	16	10
Vehicle	846	8	4
Vowel	990	10	11
Wine Quality	4898	11	7
Yeast	1484	8	10

Algorithm 2 Proposed M-split algorithm

Require:

Set of sorted instances $X = (x_i)_{i=1}^{n}$ along with fuzziness values $V = \{v_1, v_2, \cdots, v_n\}$

Ensure:

FG_{low}, FG_{mid}, FG_{high} {i.e., instances in low, mid and high categories}

Basic steps:

initialize $q_1 = q_2 = 0$
$M_1 = M_v(V)$
if $M_1 \neq 0$ **then**
  $q_1 = M_v(V < M_1)$
  if $(q_1 = 0)$ **then**
    $low_M = q_1$
    if $\max(V \neq q_1) > 0$ **then**
      $q_2 = M_v(V > q_1)$
      $high_M = q_2$
    end if
  else
    $q2 = M_v(V > M_1)$
    $V_1 = V < m$ and $V \geq q_1$)
    $low_M > m$ and $V \geq q_2$)
    $high_M = M_v(V_1)$
  end if
end if
if $M_1 = 0$ **then**
  $low_M = M_1$;
  if $\max(V \neq M_1) > 0$ **then**
    $q_2 = M_v(V > M_1)$
    $high_M = q_2$
  end if
  if $\max(V) > q_2$ **then**
    $high_M = M_v(V > q_2)$
  end if
end if
for $i = 1$ to n **do**
  if $V_i \leq low_M$ **then**
    $FG_{low} = x_i$
  else if ($V_i \leq low_M$ and $V_i \leq high_M$) **then**
    $FG_{mid} = x_i$
  else if $V_i \geq high_M$ **then**
    $FG_{high} = x_i$
  end if
end for

formance of all splitting approaches. We performed necessary normalizing between [0, 1]. For the NNR_w, the initial interval for the random parameters is set to [0, 1], and the number hidden nodes are selected as 60 corresponding to each dataset. We applied all three mechanisms for splitting the instances into 3 groups during the experimental process. Table 2 lists the results obtained by using 10-times 10-fold cross validation scheme, where average is taken to demonstrate the effectiveness of sampling method.

5.1. Analysis of results

For further analysis of our experiments, we take a typical case of one data set (i.e., automobile), We vary the number of hidden nodes from 10 to 100, and check the impact of categorization by using 3 splitting approaches. One can see in Fig. 5, where the proposed splitting approach gains more accuracy than the others. For the NNR_w, we also analyze the impact of different initialization intervals on the overall performance of our proposed splitting approaches. The initialization interval is set to $[0, \lambda]$, $1 \leq \lambda] \leq 10$ during the simulation. The input weights $\mathbf{w}_i$ between input layer nodes and hidden layer nodes, and biases b_i at the hidden layer of NNR_w are the random values that follow the *uniform distribution* over $[0, \lambda]]$. The impact of initialization interval is shown in Fig. 6. The results which are shown in Table 2, a smaller interval, i.e., [0, 1] is selected for all the data sets.

One can analyze the Figs. 5 and 6, that all splitting methods are improving the accuracy rate, where FG_{low} is extracted after applying each splitting method and incorporating into original training set L. In-fact, the division of instances into 3 groups depends on the nature of a problem, and the amount of instances that the data sets hold. Suppose in a case, we have limited fuzziness values then M-split criteria may fail to produce the desirable results, hence, we further need to judge the impact of fuzziness values. The other two types of splitting can be effectively utilized in such type of situation. Same is the case with *penbased* and yeast data sets, where the M-split criteria achieves slightly low or equivalent results than *N-split* and *P-split* dividing approaches. M-split criteria is useful in the case, when most of the fuzziness values are similar, for example, in the case of 100 instances, if 50 instances are outputting the same fuzziness values i.e., 0, then the M-split criteria will place all similar values in one category, and other values will be further divide into two groups. The main step in D&C strategy was to incorporate that group with L, that are having high accuracy. All splitting methods are following this phenomenon, we have listed the accuracy obtained by FG_{low}, FG_{mid} and FG_{high} in Table 2, where one can see the accuracies obtained by 3 groups by using different splitting approaches. By using any splitting method, FG_{low} is

Table 2
Experimental Results using different splitting methods for instance categorization

Data set	Split method	Groups accuracy			(Orignal accuracy)	D&C Accuracy
		FG_{low}	FG_{mid}	FG_{high}		$(L \cup FG_{low})$
		$Train_{acc}$, $Test_{acc}$	$Train_{acc}$, $Test_{acc}$	$Train_{acc}$, $Test_{acc}$	$Train_{acc}$, $Test_{acc}$	$Train_{acc}$, $Test_{acc}$
	M-split	98.8182, 80.1375	93.8207, 62.6682	81.8428, 52.6000		90.7516, 69.9414
Automobile	N-split	99.1458, 83.1538	94.4605, 63.2000	81.1195, 52.6154	91.2747, 66.0861	90.8728, 68.2418
	P-split	99.7241, 87.8750	92.7433, 65.3034	78.3583, 47.3750		91.1446, 67.0458
	M-split	96.8211, 95.2013	93.3508, 86.7625	73.0313, 60.7509		87.4273, 81.2254
Autompg	N-split	96.8811, 95.7807	93.3381, 86.3577	71.4244, 59.9167	87.0946, 80.8026	87.3888, 81.0976
	P-split	97.2375, 96.8442	91.4578, 83.9874	63.8414, 54.2857		87.3036, 80.9244
	M-split	84.0777, 68.4866	73.6655, 55.8810	59.7145, 38.3875		72.1898, 55.9016
Cleveland	N-split	84.9461, 71.3651	73.4587, 55.8000	59.1743, 36.8889	72.3882, 54.6837	72.2759, 55.3746
	P-split	88.6638, 77.4444	72.6884, 53.8333	55.1925, 34.7778		72.3295, 54.7008
	M-split	97.7287, 95.5022	91.0613, 85.5333	84.3161, 73.6883		91.2654, 85.1053
Ecoli	N-split	97.9787, 95.6667	90.9200, 86.1409	84.3977, 72.8000	91.0333, 84.8736	91.1821, 85.0232
	P-split	98.7580, 97.6515	90.8700, 85.3113	83.7884, 70.7121		91.1656, 84.9089
	M-split	92.9371, 78.1214	85.0707, 64.6923	77.5583, 48.7368		84.9568, 66.1966
Glass	N-split	93.6922, 80.3647	85.0273, 65.1484	77.0230, 48.7368	85.2432, 64.5897	85.0629, 64.8355
	P-split	96.5216, 84.2333	84.5175, 65.7500	76.1611, 39.9666		85.1487, 64.5491
	M-split	99.4064, 99.3752	96.7494, 96.5805	88.5855, 88.2326		94.7906, 94.5366
Penbased	N-split	99.8337, 99.8131	96.2213, 96.0770	84.9236, 84.4599	94.6845, 94.5011	94.7557, 94.5693
	P-split	99.5307, 99.5087	96.7504, 96.5688	87.9733, 87.6387		97.7854, 94.5384
	M-split	91.8648, 86.2321	79.8904, 72.6243	69.5294, 62.8333		80.6501, 74.8663
Vehicle	N-split	92.9659, 87.3471	90.0156, 73.4436	68.5645, 61.7837	80.5101, 74.1843	80.5101, 74.7486
	P-split	95.9224, 92.4435	80.6590, 73.7958	64.6470, 57.1399		80.7266, 74.2444
	M-split	96.5716, 90.2376	91.0909, 82.8485	83.6699, 72.8125		90.3772, 83.6818
Vowel	N-split	96.9635, 91.0681	91.1370, 83.3778	83.2443, 71.5292	90.3770, 82.0379	90.3860, 83.3485
	P-split	98.1420, 93.9135	91.0029, 82.4839	80.7112, 68.6154		90.4339, 82.5303
	M-split	61.9334, 59.5643	55.4848, 54.6103	50.4094, 49.7300		56.2356, 55.0391
Wine Q.	N-split	62.4835, 60.0390	55.4844, 54.4016	50.0505,49.4889	56.0007, 54.6395	56.2289, 55.0107
	P-split	63.9780, 61.1239	55.6447, 54.4331	49.0918, 48.7773		56.1774, 54.9376
	M-split	69.9695, 67.7769	59.6117, 56.5937	58.2914, 53.6273		62.9979, 59.8140
Yeast	N-split	70.8206, 68.7880	59.3902, 56.4407	58.4635, 54.0169	62.8497, 59.6581	63.0291, 59.8308
	P-split	73.0670, 70.8039	60.8055, 58.3083	58.7928, 52.5645		63.0335, 59.7451

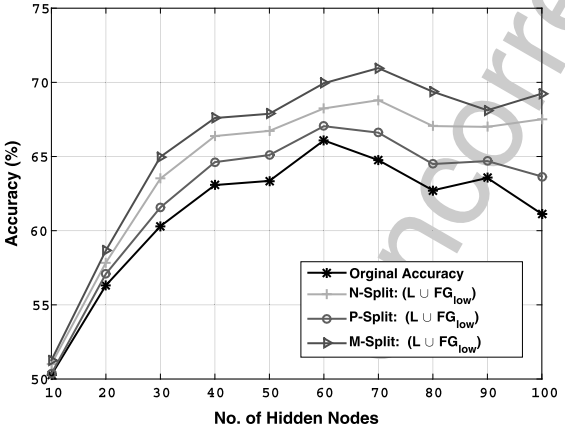

Fig. 5. Comparison of splitting methods (Testing accuracy).

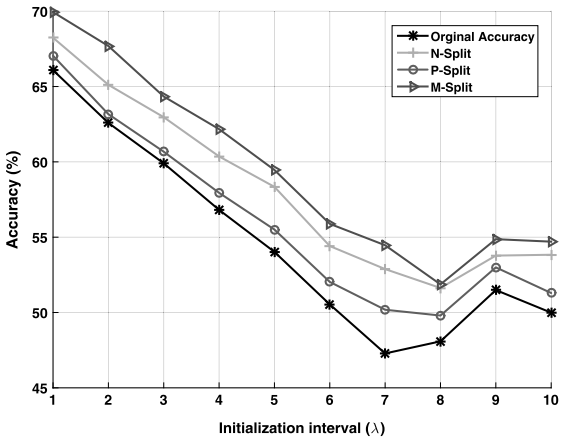

Fig. 6. Impact of different initialization interval (Testing accuracy).

getting higher accuracy than FG_{mid} and FG_{high}, and for many data sets *M-split* criteria is reflecting higher accuracy rate on withheld set than the original testing accuracy. Table 2 and both Figs. 5 and 6 prove the effectiveness of *M-Split* criteria.

For the so called *problem of big-data*, selection of *M-split* criteria may be much useful than other criteria, because N-split or P-split may extract much misclassified instances for FG_{low}. During this study, we have also observed some interesting findings by adding two groups collectively along with L, we will present this part in our next study.

6. Conclusion

Fuzziness based divide and conquer strategy is an effective and useful strategy for promoting the classifier's performance. However, the critical problem is to categorize the instances according to their fuzzy values into 3 groups. We have investigated different states of the art methods for categorization, and proposed an efficient mechanism that can effectively extract the instances for low, mid and high fuzziness categories. We used NNR_w to obtain the membership vector corresponding to each data point by using simple transformation. Other classifiers i.e., Fuzzy k-nearest neighbor, support vector machine (SVM) or BP can also be utilized to compute the fuzziness for data points. It is observed by the experimental simulation that proposed splitting algorithm provides better solution for placement the data points into respective categories, and also promotes the efficacy of D&C strategy. This technique of categorization can be much helpful for the so called *problem of big-data*. Experimental results have shown its effectiveness.

Acknowledgments

This work is supported by the National Natural Science Foundation of China 71371063, the Basic Research Project of Knowledge Innovation Program in Shenzhen (JCYJ20150324140036825).

References

[1] S. Al-sharhan, F. Karray, W. Gueaieb and O. Basir, Fuzzy entropy: A brief survey. In *10th IEEE International Conference on Fuzzy Systems (Cat. No. 01CH37297)*, volume 2, 2001, pp. 1135–1139. IEEE.

[2] M. Alhamdoosh and W. Dianhui, Fast decorrelated neural network ensembles with random weights, *Information Sciences* **264** (2014), 104–117.

[3] D. Angluin, Queries and concept learning, *Machine Learning* **2**(4) (1988), 319–342.

[4] R.A.R. Ashfaq, Y.-L. He and D.-G. Chen, Toward an efficient fuzziness based instance selection methodology for intrusion detection system, *International Journal of Machine Learning and Cybernetics* (2016). DOI:10.1007/s13042-016-0557-4

[5] R.A.R. Ashfaq, X.-Z. Wang, J.Z. Huang, H. Abbas and H. Yu-Lin, Fuzziness based semisupervised learning approach for intrusion detection system, *Information Sciences* **378** (2017), 484–497.

[6] J. Calvo-Zaragoza, J.J. Valero-Mas and J.R. Rico-Juan, Improving kNN multi-label classification in Prototype Selection scenarios using class proposals, *Pattern Recognition* **48**(5) (2015), 1608–1622.

[7] F. Cao, H. Ye and W. Dianhui, A probabilistic learning algorithm for robust modeling using neural networks with random weights, *Information Sciences* **313** (2015), 62–78.

[8] C. Cortes and V. Vapnik, Support-vector networks, *Machine Learning* **20**(3) (1995), 273–297.

[9] A. De Luca and S. Termini, A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory, *Information and Control* **20**(4) (1972), 301–312.

[10] S. Ding, X. Zhang and J. Yu, Twin support vector machines based on fruit fly optimization algorithm, *International Journal of Machine Learning and Cybernetics* **7**(2) (2016), 193–203.

[11] D. Dubois and H.M. Prade, Fuzzy sets and systems: Theory and applications, Academic Press, 1980.

[12] R.B. Gnana Jothi and S.M. Meena Rani, Hybrid neural network for classification of graph structured data, *International Journal of Machine Learning and Cybernetics* **6**(3) (2015), 465–474.

[13] J. Hamidzadeh, R. Monsefi and H.S. Yazdi, Large symmetric margin instance selection algorithm, *International Journal of Machine Learning and Cybernetics* **7**(1) (2016), 25–45.

[14] Y.-L. He, X.-Z. Wang and J.Z. Huang, Fuzzy nonlinear regression analysis using a random weight network, *Information Sciences* **364** (2016), 222–240.

[15] B. Igelnik and P. Yoh-Han, Stochastic choice of basis functions in adaptive function approximation and the functionallink net, *IEEE Transactions on Neural Networks* **6**(6) (1995), 1320–1329.

[16] X. Jiang and W. Zhang, Structure learning for weighted networks based on Bayesian nonparametric models, *International Journal of Machine Learning and Cybernetics* **7**(3) (2016), 479–489.

[17] W. Kang, S. Zhong and J. Cheng, Relaxed passivity conditions for discrete-time stochastic delayed neural networks, *International Journal of Machine Learning and Cybernetics* **7**(2) (2016), 205–216.

[18] D.D. Lewis and J. Catlett, Heterogeneous uncertainty sampling for supervised learning, In *Proc 11th International Conference on Machine Learning*, Morgan Kaufmann, 1994, pp. 148–156.

[19] D.D. Lewis and W.A. Gale, A Sequential Algorithm for Training Text Classifiers, Springer-Verlag, 1994, pp. 3–12.

[20] X. Li and D. Li, The structure and realization of a polygonal fuzzy neural network, *International Journal of Machine Learning and Cybernetics* **7**(3) (2016), 375–389.

[21] M. Lichman, {UCI} Machine Learning Repository, 2013.

[22] D. Liu and Y. Du, New results of stability analysis for a class of neutral-type neural network with mixed time delays, *International Journal of Machine Learning and Cybernetics* **6**(4) (2015), 555–566.

[23] J. Arturo Olvera-López, J. Ariel Carrasco-Ochoa, J. Francisco Martínez-Trinidad and J. Kittler, A review of instance selection methods, *Artificial Intelligence Review* **34**(2) (2010), 133–143.

[24] Y.-H. Pao, G.-H. Park and D.J. Sobajic, Learning and generalization characteristics of the random vector functional-link net, *Neurocomputing* **6**(2) (1994), 163–180.

[25] S. Scardapane, D. Wang, M. Panella and A. Uncini, Distributed learning for Random Vector Functional-Link networks, *Information Sciences* **301** (2015), 271–284.

[26] W.F. Schmidt, M.A. Kraaijveld and R.P.W. Duin, Feedforward neural networks with random weights, In *Proceedings, 11th IAPR International Conference on Pattern Recognition Conference B: Pattern Recognition Methodology and Systems*, Vol. II, pp. 1–4. IEEE Comput Soc Press.

[27] H.S. Seung, M. Opper and H. Sompolinsky, Query by committee, In *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*, New York, New York, USA, 1992, pp. 287–294. ACM Press.

[28] X.-Z. Wang, R.A.R. Ashfaq and A.-M. Fu, Fuzziness based sample categorization for classifier performance improvement, *Journal of Intelligent & Fuzzy Systems* **29**(3) (2015), 1185–1196.

[29] X.-Z. Wang, L.-C. Dong and J.-H. Yan, Maximum ambiguity-based sample selection in fuzzy decision tree induction, *IEEE Transactions on Knowledge and Data Engineering* **24**(8) (2012), 1491–1505.

[30] X.-Z. Wang, H.-J. Xing, Y. Li, Q. Hua, C.-R. Dong and W. Pedrycz, A study on relationship between generalization abilities and fuzziness of base classifiers in ensemble learning, *IEEE Transactions on Fuzzy Systems* **23**(5) (2015), 1638–1654.

[31] Y.F. Yam, W.S. Tommy and C.T. Chow, Leung, A new method in determining initial weights of feedforward neural networks for training enhancement, *Neurocomputing* **16**(1) (1997), 23–32.

[32] Z.-M. Yang, H.-J. Wu, C.-N. Li and Y.-H. Shao, Least squares recursive projection twin support vector machine for multi-class classification, *International Journal of Machine Learning and Cybernetics* **7**(3) (2016), 411–426.

[33] L. Yuan, J. Liu, X. Tang, D. Shi and L. Zhao, Pairwise-similarity-based instance reduction for efficient instance selection in multiple-instance learning, *International Journal of Machine Learning and Cybernetics* **6**(1) (2015), 83–93.

[34] L.A. Zadeh, Fuzzy sets, *Information and Control* **8**(3) (1965), 338–353.

[35] L.A. Zadeh, Probability measures of Fuzzy events, *Journal of Mathematical Analysis and Applications* **23**(2) (1968), 421–427.

[36] J. Zhai, X. Wang and X. Pang, Voting-based instance selection from large data sets with Map Reduce and random weight networks, *Information Sciences* **367-368** (2016), 1066–1077.

[37] J. Zhao, Z. Wang, F. Cao and W. Dianhui, A local learning algorithm for random weights networks, *Knowledge-Based Systems* **74** (2015), 159–166.