Neurocomputing 225 (2017) 205–213

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Monotonic classification extreme learning machine

Hong Zhu[a], Eric C.C. Tsang[a,*], Xi-Zhao Wang[b], Rana Aamir Raza Ashfaq[b]

[a] Faculty of Information Technology, Macau University of Science and Technology, Macau, China
[b] College of Computer Science & Software Engineering, Shenzhen University, Shenzhen 518060, China

ARTICLE INFO

Communicated by H. Yu

Keywords:
Monotonic classification
Extreme learning machine
Constrained extreme learning machine
Monotonicity
Quadratic programming

ABSTRACT

Monotonic classification problems mean that both feature values and class labels are ordered and monotonicity relationships exist between some features and the decision label. Extreme Learning Machine (ELM) is a single-hidden layer feedforward neural network with fast training rate and good generalization capability, but due to the existence of training error, ELM cannot be directly used to handle monotonic classification problems. This work proposes a generalization of ELM for processing the monotonic classification, named as Monotonic Classification Extreme Learning Machine (MCELM) in which the monotonicity constraints are imposed to the original ELM model. Mathematically, MCELM is a quadratic programming problem in which the monotonicity relationships are considered as constraints and the training error is the objective to be minimized. The mathematical model of MCELM not only can make the generated classifier monotonic but also can minimize the classification error. MCELM does not need to tune parameters iteratively, and therefore, keeps the advantage of extremely fast training which is the essential characteristic of ELM. MCELM does not require that the monotonic relationships existing between features and the output are consistent, which essentially relaxes the assumption of consistent monotonicity used in most existing approaches to handling monotonic classification problems. In comparison with exiting approaches to handling monotonic classification, MCELM can indeed generate a monotonicity-reserving classifier which experimentally shows a much better generalization capability on both artificial and real world datasets.

1. Introduction

As a fundamental task of supervised learning, classification is to get a classifier by training a number of labeled samples, and then to predict the class label of an unseen sample based on the trained classifier. From references we can find many different algorithms proposed to solve classification problems, such as decision tree induction [1], Bayesian classifier [2], kernel methods [3], support vector machine [4], artificial neuron networks [5], etc. They can be used to handle different kinds of data, such as uncertain data [6,7] and ordinal data.

Ordinal classification is a generalization of the traditional classification. In traditional classification problems the class labels are nominal while in the ordinal classification the class labels are ordered, in other words, there is an order relationship among the class labels in ordinal classifications. For example, the damage degree after a typhoon hit can be classified into three levels: slight, moderate and serious. It is clear that there is an order relationship among these class labels. In comparison with the traditional classification, the ordinal classification sufficiently employs the information of order among the labels.

The monotonic classification is essentially an ordinal classification with monotonicity constraints, in which both feature values and class labels are ordered and monotonic relationships exist between them. The monotonicity can be either increasing or decreasing. If the decision value increases (decreases) with the increasing (decreasing) of a particular feature value, then the monotonicity between the decision attribute and the feature is increasing; otherwise it is decreasing. For example, the grade of scholarship is monotonically increasing with respect to the student's academic performance and the satisfaction degree of a car is monotonically decreasing with respect to the maintenance cost. Monotonic classification problems widely exist in many real application fields such as disease diagnoses, bankruptcy risk assessments and employee selections.

In recent decades, the monotonic classification has attracted lots of attention from researchers. The following is an incomplete survey on monotonic classifications.

In 1989, Ben David et al. introduced the first algorithm OLM [8] for ordinal classification with monotonic constraints in the machine learning community. This method consists of two components. It first chooses a subset of training objects and then classifies the samples of the subset by using a function. Unfortunately this approach may

* Corresponding author.
E-mail addresses: xszhuhong@163.com (H. Zhu), cctsang@must.edu.mo (E.C.C. Tsang), xizhaowang@ieee.org (X.-Z. Wang), aamir@szu.edu.cn (R. Aamir Raza Ashfaq).

http://dx.doi.org/10.1016/j.neucom.2016.11.021
Received 21 July 2016; Received in revised form 21 October 2016; Accepted 4 November 2016
Available online 20 November 2016
0925-2312/ © 2016 Published by Elsevier B.V.

produce non-monotonic results. Greco et al. proposed a dominance-based rough set approach [9] to deal with the multi-criteria sorting, which is an extension of the classical rough set approach [10]. The approach handles inconsistencies coming from violation of the monotonicity constraints by substituting the indiscernibility relation with the dominance relation. It is the first approach using the comprehensive theory for monotonic classification tasks in the domain of knowledge discovery. Further in [11,12], the model of dominance rough set was used to extract rules for monotonic classification. Similar pieces of work can be found from many other researchers' studies, such as [13–18]. Among the existing approaches to learning with monotonicity constraints, the nonparametric approach is the most general one. In [19], the authors introduced a probabilistic model for ordinal classifications with monotonicity constraints based on the concept of stochastic dominance and investigated the possible loss functions. Its main contribution is considered as the analysis on nonparametric approach from a statistical point of view. The analysis suggested that convex losses were suitable for monotonic classification, and provided a necessary foundation for nonparametric methods.

Decision tree algorithms such as the CART [20] and the C4.5 [21] are widely used in the machine learning community to solve classification problems. Many variants of these algorithms have been developed and segment based decision tree [22] is an instance. To solve ordinal classifications with monotonicity constraints, lots of decision tree studies on monotonic classifications have been done. The following is a brief summary about them.

Ben-David [23] introduced a tree induction algorithm called Monotone Induction of Decision trees (MID) to this context. His main work was to modify the conditional entropy by adding a term called order-ambiguity-score, which made the splitting strategy monotonic. In 1999, Makino et al. provided two algorithms which were named as Positive Decision Trees (PDT) and Quasi-Positive Decision Trees (Q-PDT) [24] based on the former decision tree algorithms. However, both of these algorithms can only be applied to solve binary-class monotonic classification problems. In [25–29], those two algorithms were extended to handle K-class monotonic classification problems. In 2000, R. Potharst and J. C. Bioch developed an order-preserving tree-generation algorithm and provided a technique for repairing non-monotonic decision trees for multi-attribute classification problems with several linearly ordered classes [30]. In 2003, A. J. Feelders and Martijn Pardoel proposed a collection of pruning techniques to make a non-monotonic tree monotonic [31]. In the same year, a tree construction algorithm was proposed by Cao-Van and De Baets to avoid violating the monotonicity of data [32,33]. In 2008, Xia, et al. extended the Gini impurity used in CATR to ordinal classifications, and called it ranking impurity [34]. In 2009, Kamp, Feelders and Barile proposed a new algorithm for learning isotonic classification trees [35].

The algorithms mentioned above have enhanced the capability of extracting ordinal information. However, they have a common disadvantage, that is, the algorithms cannot guarantee a monotonic tree is generated if the training data are monotonically increasing or decreasing.

To generate a monotonic classifier for monotonic classification problems, in 2011 Qinghua Hu et al. introduced a new measure of feature quality, which was called rank mutual information [36]. It not only can inherit the advantage of robustness from Shannon's entropy, but also can extract ordinal structures from monotonic datasets. Based on the rank mutual information, the authors designed a decision tree algorithm named REMT which can get consistently monotonic decision tree. However, the REMT algorithm does not take the classification error into account, which cannot guarantee the classification accuracy of the generated classifier. Moreover, the classifier is obtained under the restriction that all the monotonic relationships between features and the decision attribute are consistent; otherwise, the data have to be preprocessed to meet this restriction, which may cause information loss. Besides, in many other existing methods, such as those proposed in [19,36], all the monotonic relationships are required to be consistent, too.

Conventional artificial neural networks have great approximation capability and almost always generate good separation in training datasets, but the behavior of the neural networks training process depends heavily on the training set. For a given training sample the classification boundary generated by a neural network is relatively unpredictable, especially in the case of small sample size. There are many extendings of artificial neural network. Such as the polygonal fuzzy neural network which is proposed to handle polygonal fuzzy data in [37], weighted networks [38,39] and discrete-time stochastic neural networks [40]. In addition, to make a breakthrough, researchers have proposed a few methods which are based on artificial neural networks to handle monotonic classification problems. In order to solve the two-group classification problems with monotonicity constraints, Norman P. Archer [41] developed a modification of the existing back propagation neural network algorithm by preprocessing the training samples with a linear classification function. In [42], C. Li et al. studied the monotonic type-2 fuzzy neural network (T2FNN). Under the monotonicity constraints, a hybrid algorithm was provided to optimize the parameters of the monotonic T2FNN. In [43], C. Li et al. studied the incorporation of monotonicity into interval type-2 fuzzy logic systems. They showed that type-2 fuzzy logic systems were monotonic if the antecedent and consequents parts of their fuzzy rules were arranged according to the proposed monotonicity conditions. These approaches tend to be much more efficient than standard back propagation learning algorithms. However, all the parameters in these algorithms still need to be tuned iteratively by using the gradient descent method, which is time consuming and easily gets stuck in local minima.

In the past two decades, support vector machine (SVM) [44] and its variants [45–48] have been extensively used in classification applications due to their surprising classification capability. But the optimization constraints are strict and the generalization ability is not very good. The ELM method proposed in [49] has overcome the shortcomings of SVM. ELM [49] is a single-hidden layer feed-forward neural network with fast training rate and good generalization capability, but due to the existence of training error, ELM cannot be directly used to handle monotonic classification problems. Motivated by extending ELM for handling monotonic classifications and by overcoming the above-mentioned shortcomings of those approaches handling monotonic classifications, this paper proposes a generalization of ELM, which is named as Monotonic Classification Extreme Learning Machine (MCELM) in which the monotonicity constraints are imposed to the original ELM model. The mathematical model of MCELM is a quadratic programming problem in which both the classification error and monotonicity are taken into account. The model can perform well with very little classification error, and more importantly, can ensure that the generated classifier is monotonic. Moreover, different from other existing approaches, MCELM does not require that all the monotonic relationships between features and the decision attribute are consistent, which indicates that the related data preprocessing is unnecessary and furthermore some information loss can be avoided. Similar to ELM, in MCELM the gradient descent method is not adopted and all the parameters do not need to be tuned iteratively, which keeps the essential advantage of extremely fast training of the original ELM.

The rest of the paper is organized as follows. Section 2 introduces the basic knowledge of monotonic classification and ELM. Section 3 develops our MCELM model and describes the framework of MCELM. Experimental results are presented in Section 4. Finally, Section 5 provides the concluding remarks.

2. Basic knowledge of monotonic classification and ELM

2.1. Monotonic classification

Suppose that $U = \{x_1, \ldots x_n\}$ is the set of objects; A is the set of features describing the objects; D is the decision attribute of the samples, which represents the class label in classification problems. The value of sample x_i in terms of attribute $a \in A$ or D is denoted by $v(x_i, a)$ or $v(x_i, D)$ respectively. The ordinal relationship between samples in terms of attribute a or D is denoted by '$\leq$' or '$\geq$'. We say x_j is not worse than x_i in term of a or D if $v(x_i, a) \leq v(x_j, a)$ or $v(x_i, D) \leq v(x_j, D)$ denoted by $x_i \leq_a x_j$ and $x_i \leq_D x_j$ respectively. Correspondingly, we can also define $x_i \geq_a x_j$ and $x_i \geq_D x_j$. Given $B \subseteq A$, we say $v(x_i, B) = v(x_j, B)$ if for $\forall\, a \in B$, we have $v(x_i, a) = v(x_j, a)$.

Definition 1. Given a feature a, let $B = A - \{a\}$. For $\forall\, x_i, x_j \in U$, under the restriction of $v(x_i, B) = v(x_j, B)$, if $v(x_i, a) \geq v(x_j, a)$ then $v(x_i, D) \geq v(x_j, D)$ or if $v(x_i, a) \leq v(x_j, a)$ then $v(x_i, D) \leq v(x_j, D)$, we say decision attribute D is monotonically increasing with respect to feature a; otherwise if $v(x_i, a) \geq v(x_j, a)$ then $v(x_i, D) \leq v(x_j, D)$ or if $v(x_i, a) \leq v(x_j, a)$ then $v(x_i, D) \geq v(x_j, D)$, we say decision attribute D is monotonically decreasing with respect to feature a.

In monotonic classification problems, some of the monotonicity relationships are increasing and some are decreasing. This phenomenon widely exists in our real life. For instance, in the problem of car evaluation, the car acceptability increases when the size of luggage boot varies from small to big and it decreases when the maintenance cost varies from low to high.

2.2. Extreme Learning Machine (ELM)

In conventional feedforward neural networks, all parameters are tuned iteratively by using gradient descent technique. Therefore the learning speed of feedforward neural networks is in general far slower than required, which has been a major bottleneck in applications of the gradient descent-based neural networks for past decades. Moreover, conventional neural networks may easily converge to local minima. To overcome these shortcomings, Guang Bin Huang et al. proposed the Extreme Learning Machine (ELM) in [49]. The ELM algorithm can learn thousands of times faster than the BP-based algorithm in conventional neural networks. Furthermore ELM can produce good generalization performance in most cases and has been successfully applied to many classification and regression problems [50–53]. In this section, we will introduce the basic framework of ELM briefly.

In fact ELM is a single-hidden layer feedforward neural network (SLFN). There are three layers in the framework of ELM: the input layer, the hidden layer and the output layer. The input weights link the input layer to the hidden layer and the output weights connect the hidden layer to the output layer.

In the framework of ELM, suppose there are n neurons in the input layer, $\widetilde{N}$ neurons in the hidden layer and m neurons in th output layer. For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in R^m$ $(1 \leq i \leq N$, R is a set of real numbers), ELM with activation functions $g_i(x) (i = 1, \ldots, \widetilde{N})$ is mathematically modeled as

$$\sum_{i=1}^{\widetilde{N}} \boldsymbol{\beta}_i g_i(\mathbf{x}_j) = \sum_{i=1}^{\widetilde{N}} \boldsymbol{\beta}_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j, \quad j = 1, \ldots, N. \tag{1}$$

In Eq. (1), $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is a weight vector connecting the ith hidden node and the input nodes, $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is a weight vector connecting the ith hidden node and the output nodes, and b_i is the threshold of the ith hidden node $(i = 1, \ldots, \widetilde{N})$. $\mathbf{w}_i \cdot \mathbf{x}_j$ denotes the inner product of $\mathbf{w}_i$ and $\mathbf{x}_j$.

As named by Huang et al. in [54,55], $\mathbf{H}$ is called the hidden layer output matrix of the neural network. The expression of $\mathbf{H}$ is shown as follows:

$$\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_{\widetilde{N}}, b_1, \ldots, b_{\widetilde{N}}, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\widetilde{N}} \cdot \mathbf{x}_1 + b_{\widetilde{N}}) \\ \cdots & \cdots & \cdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_{\widetilde{N}} \cdot \mathbf{x}_N + b_{\widetilde{N}}) \end{bmatrix}_{N \times \widetilde{N}}. \tag{2}$$

Traditionally, in order to train a single-hidden layer feedforward neural network, we wish to find specific $\widehat{\mathbf{w}}_i, \widehat{b}_i \ (i = 1, \ldots, \widetilde{N})$ and $\widehat{\boldsymbol{\beta}}$ such that

$$\|\mathbf{H}(\widehat{\mathbf{w}}_1, \ldots, \widehat{\mathbf{w}}_{\widetilde{N}}, \widehat{b}_1, \ldots, \widehat{b}_{\widetilde{N}})\widehat{\boldsymbol{\beta}} - \mathbf{T}\| = \min_{\mathbf{w}_i, b_i, \boldsymbol{\beta}} \|\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_{\widetilde{N}}, b_1, \ldots, b_{\widetilde{N}})\boldsymbol{\beta} - \mathbf{T} \\ \|, \tag{3}$$

where

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \cdots \\ \boldsymbol{\beta}_{\widetilde{N}}^T \end{bmatrix}_{\widetilde{N} \times m}, \quad \widehat{\boldsymbol{\beta}} = \begin{bmatrix} \widehat{\boldsymbol{\beta}}_1^T \\ \cdots \\ \widehat{\boldsymbol{\beta}}_{\widetilde{N}}^T \end{bmatrix}_{\widetilde{N} \times m}, \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \cdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m}, \tag{4}$$

which is equivalent to minimizing the cost function

$$E = \sum_{j=1}^{N} \left(\sum_{i=1}^{\widetilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) - t_j \right)^2. \tag{5}$$

In the ELM model, the input weights $\mathbf{w}_i$ and hidden layer biases b_i are randomly assigned. For fixed $\mathbf{w}_i$ and b_i, where $1 \leq i \leq \widetilde{N}$, to train a single-hidden layer feedforward neural network is simply equivalent to finding a least-squares solution $\widehat{\boldsymbol{\beta}}$ of the linear system $\boldsymbol{H\beta} = \mathbf{T}$, i.e.,

$$\|\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_{\widetilde{N}}, b_1, \ldots, b_{\widetilde{N}})\widehat{\boldsymbol{\beta}} - \mathbf{T}\| = \min_{\boldsymbol{\beta}} \|\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_{\widetilde{N}}, b_1, \ldots, b_{\widetilde{N}})\boldsymbol{\beta} - \mathbf{T}\|. \tag{6}$$

According to[49], the smallest norm least-squares solution of the above system is

$$\widehat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{7}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$ [56,57].

The learning speed of ELM is much faster than those of the traditional gradient-based learning algorithms because activation functions are infinitely differentiable in ELM, which makes input weights and hidden layer biases can be randomly assigned. In ELM, once input weights and hidden layer bases are fixed, output weights can be analytically determined by Eq. (7). All parameters do not need to be tuned iteratively in ELM.

ELM can infinitely approximate any function when the number of hidden neurons is infinitely close to the number of training samples. But limited by some factors, the number of hidden neurons or training samples may not be enough, which can weaken the approximation capability of ELM. Therefore, when ELM is used to solve monotonic classification problems, the generated classifier cannot be guaranteed being monotonic, which will be illustrated by the following example.

Example 1. We use the following function to generate a dataset:

$$f(x_1, x_2, x_3) = x_1 - x_2^2 + \sin x_3, \tag{8}$$

where x_1, x_2 and x_3 are three random variables which represent the values of three features a_1, a_2 and a_3 ($a_1, a_2, a_3 \in A$ is the set of features that describe samples) of each sample. x_1, x_2 and x_3 are independently drawn from the uniform distribution over the interval $[0, \pi]$.

In order to generate ordered class labels, the resulting numeric values are discretized into eight intervals $[-10, -8], (-8, -6], \ldots, (4, 6]$, corresponding to which the class labels are marked as $1, 2, \ldots, 8$. The samples of which the function values belonging to the same interval share the same rank label. Then we form an eight-class dataset in which the class label is monotonically increasing with respect to the feature a_1 and monotonically decreasing with respect to the feature a_2.

According to the above method, we generate a dataset with 100 samples, based on which a classifier is trained by ELM. Then we test the performance of the classifier and find that the predicted class label of sample (1.3418, 1.5335, 0.8736) is 5 while the predicted class label of sample (0.7872, 1.5335, 0.8736) is 6, which goes against of the increasing monotonicity relationship between feature a_1 and the decision attribute.

The example shows that ELM cannot guarantee the generated classifier is monotonic when the training data are monotone. In order to solve monotonic classification problems, we impose monotonicity constraints to the framework of the original ELM model in this work.

3. Monotonic classification extreme learning machine (MCELM)

3.1. The framework of MCELM

Similar to ELM, there are three layers in the framework of MCELM. The first layer is called the input layer, suppose where there are n neurons, and each of them represents a real input value of a feature. The number of input neurons equals to that of the features which are used to describe samples. The second layer is the hidden layer which contains $\widetilde{N}$ neurons. The last layer is the output layer. In MCELM, there is only one neuron which represents the classification result in the output layer.

Suppose $S = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ is a training set and $\mathbf{x}_j = [x_1, x_2, \dots, x_n]$ $(1 \le j \le N)$ is a sample which belongs to S, where N is the number of training samples; n is the total number of features. x_k $(1 \le k \le n)$ is the kth feature that describes the samples. $t_j \in \mathbf{R}$ is the decision value of sample x_j.

MCELM with $\widetilde{N}$ hidden nodes and activation function $g(x)$ is mathematically modeled as

$$\sum_{i=1}^{\widetilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = o_j, \quad j = 1, \dots, N, \tag{9}$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T$ is the weight vector connecting the ith hidden node and the input nodes; β_i is the output weight connecting the ith hidden node and the output node; b_i is the threshold of the ith hidden node, where $1 \le i \le \widetilde{N}$.

When solving monotonic classification problems based on MCELM, if increasing monotonicity is desired for a particular feature, then the partial derivative of the decision attribute with respect to the feature is constrained to be positive. Similarly, partial derivative with respect to a feature where decreasing monotonicity is required is considered to be negative. It is not necessary to require all the monotonic relationships between features and the decision attribute to be consistent, in other words, some of the monotonic relationships can be increasing and some can be decreasing.

By constraining the signs of the partial derivatives, we can get r monotonicity constraints in monotonic classification problems, where (r is the total number of features that are monotonically increasing or decreasing with the decision attribute and $1 \le r \le n$). We incorporate these constraints into the framework of ELM.

3.2. The mathematical model of MCELM

To train a monotonic classifier by MCELM means to minimize the cost function

$$E = \sum_{j=1}^{N} \left(\sum_{i=1}^{\widetilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) - t_j \right)^2 \tag{10}$$

under those monotonicity constraints

$$\frac{\partial o}{\partial x_k} > 0 \quad \text{or} \quad \frac{\partial o}{\partial x_k} < 0, \tag{11}$$

where

$$o = \sum_{i=1}^{\widetilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x} + b_i). \tag{12}$$

$\widetilde{N}$ is the number of neurons in the hidden layer; k $(1 \le k \le n)$ is the label of the feature which has monotonic relationship with the decision attribute.

The activation functions in MCELM are restricted to be infinitely differentiable, such as the sigmoidal function, the radial basis, sine, cosine, exponential, and many non-regular functions as shown in [54], which can make input weights $\mathbf{w}_i$ and hidden layer biases b_i be assigned randomly. In this paper, we choose sigmoidal function as the active function. Once the values of $\mathbf{w}_i$ and b_i are fixed, the cost function is quadratic and the constraints are linear in terms of β_i. Therefore, the mathematical model of MCELM is actually a quadratic programming problem in terms of β_i, where $1 \le i \le \widetilde{N}$. The partial derivative of o with respect to feature x_k can be represented as follows:

$$\begin{aligned}
\frac{\partial o}{\partial x_k} &= \frac{\partial(\sum_{i=1}^{\widetilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x} + b_i))}{\partial x_k} \\
&= \frac{\partial(\beta_1 g(w_{11}x_1 + \dots + w_{1n}x_n + b_1) + \dots + \beta_{\widetilde{N}} g(w_{\widetilde{N}1}x_1 + \dots + w_{\widetilde{N}n}x_n + b_{\widetilde{N}}))}{\partial x_k} \\
&= \beta_1 \frac{\partial g(w_{11}x_1 + \dots + w_{1n}x_n + b_1)}{\partial x_k} + \dots + \beta_{\widetilde{N}} \frac{\partial g(w_{\widetilde{N}1}x_1 + \dots + w_{\widetilde{N}n}x_n + b_{\widetilde{N}})}{\partial x_k} \\
&= \beta_1 w_{1k} g'(w_{11}x_1 + \dots + w_{1n}x_n + b_1) \qquad + \dots + \beta_{\widetilde{N}} w_{Nk} g'(w_{\widetilde{N}1}x_1 + \dots \\
&\quad + w_{\widetilde{N}n}x_n + b_{\widetilde{N}}) \quad = \sum_{i=1}^{\widetilde{N}} \beta_i w_{ik} g'(\mathbf{w}_i \cdot \mathbf{x} + b_i) \\
&= \sum_{i=1}^{\widetilde{N}} \beta_i w_{ik} g(\mathbf{w}_i \cdot \mathbf{x} + b_i)(1 - g(\mathbf{w}_i \cdot \mathbf{x} + b_i)).
\end{aligned} \tag{13}$$

If the decision attribute is monotonically increasing with a feature x_k, the derivative of the decision attribute with respect to x_k is positive, i.e.,

$$\sum_{i=1}^{\widetilde{N}} \beta_i w_{ik} g(\mathbf{w}_i \cdot \mathbf{x} + b_i)(1 - g(\mathbf{w}_i \cdot \mathbf{x} + b_i)) > 0. \tag{14}$$

Similarly if the decision attribute is monotonically decreasing with feature x_k, the derivative of the decision attribute with respect to x_k is negative, i.e.,

$$\sum_{i=1}^{\widetilde{N}} \beta_i w_{ik} g(\mathbf{w}_i \cdot \mathbf{x} + b_i)(1 - g(\mathbf{w}_i \cdot \mathbf{x} + b_i)) < 0, \tag{15}$$

where $\widetilde{N}$ is the number of neurons in the hidden layer; n is the total number of features.

In equations (14) and (15) are considered as the monotonicity constraints in our MCELM model.

For the monotonically increasing situation, it is well noted that $\beta_i w_{ik} g(\mathbf{w}_i \cdot \mathbf{x} + b_i)(1 - g(\mathbf{w}_i \cdot \mathbf{x} + b_i)) > 0$ for each $(1 \le i \le \widetilde{N})$ implies In equation (14). Since $g(\mathbf{w}_i \cdot \mathbf{x} + b_i)(1 - g(\mathbf{w}_i \cdot \mathbf{x} + b_i))$ is a positive function, $\beta_i w_{ik} g(\mathbf{w}_i \cdot \mathbf{x} + b_i)(1 - g(\mathbf{w}_i \cdot \mathbf{x} + b_i)) > 0$ is equivalent to $\beta_i w_{ik} > 0$ for each i $(1 \le i \le \widetilde{N})$. In this way, we will have $\widetilde{N}$ constraints which can be expressed as

$$\begin{cases} \beta_1 w_{1k} > 0 \\ \dots \\ \beta_{\widetilde{N}} w_{\widetilde{N}k} > 0. \end{cases} \tag{16}$$

Similarly we can discuss the monotonically decreasing situation and express the constrains as

$$\begin{cases} \beta_1 w_{1k} < 0 \\ \dots \\ \beta_{\widetilde{N}} w_{\widetilde{N}k} < 0. \end{cases} \tag{17}$$

The mathematical model of MCELM can be represented as follows:

Minimize:

$$E = \sum_{j=1}^{N}\left(\sum_{i=1}^{\widetilde{N}}\beta_i g(\mathbf{w}_i\cdot\mathbf{x}_j + b_i) - t_j\right)^2 = \sum_{j=1}^{N}\sum_{i=1}^{\widetilde{N}} g^2(\mathbf{w}_i\cdot\mathbf{x}_j + b_i)\cdot\beta_i^2 + \beta_i\sum_{k=i+1}^{\widetilde{N}}\beta_k\cdot 2g(\mathbf{w}_i\cdot\mathbf{x}_j + b_i)g(\mathbf{w}_k\cdot\mathbf{x}_j + b_k) - 2g(\mathbf{w}_i\cdot\mathbf{x}_j + b_i)t_j\cdot\beta_i + t_j^2 \quad (18)$$

Subject to:

$$\begin{cases}\beta_1 w_{1k} > 0\\ \cdots\\ \beta_{\widetilde{N}} w_{\widetilde{N}k} > 0\end{cases}$$

or

$$\begin{cases}\beta_1 w_{1k} < 0\\ \cdots\\ \beta_{\widetilde{N}} w_{\widetilde{N}k} < 0.\end{cases}$$

The standard form of the quadratic programming model of MCELM is shown as follows:

$$\min_{\boldsymbol{\beta}}\frac{1}{2}\boldsymbol{\beta}^T\mathbf{C}\boldsymbol{\beta} + \mathbf{f}^T\boldsymbol{\beta} \quad (19)$$

$$\mathbf{A}\cdot\boldsymbol{\beta} \le \mathbf{B}, \quad (20)$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_{\widetilde{N}}]^T$. $\mathbf{C}$ is the Hessian matrix of the cost function (10), which is symmetric. The expression of $\mathbf{C}$ is shown as follows:

$$\mathbf{C} = \begin{bmatrix} \frac{\partial^2 E}{\partial\beta_1^2} & \frac{\partial^2 E}{\partial\beta_1\partial\beta_2} & \cdots & \frac{\partial^2 E}{\partial\beta_1\partial\beta_{\widetilde{N}}}\\ \frac{\partial^2 E}{\partial\beta_2\partial\beta_1} & \frac{\partial^2 E}{\partial\beta_2^2} & \cdots & \frac{\partial^2 E}{\partial\beta_2\partial\beta_{\widetilde{N}}}\\ \cdots & \cdots & \cdots & \cdots\\ \frac{\partial^2 E}{\partial\beta_{\widetilde{N}}\partial\beta_1} & \frac{\partial^2 E}{\partial\beta_{\widetilde{N}}\partial\beta_2} & \cdots & \frac{\partial^2 E}{\partial\beta_{\widetilde{N}}^2} \end{bmatrix}. \quad (21)$$

The element in the *i*th row and the *j*th column of matrix $\mathbf{C}$ is

$$\frac{\partial^2 E}{\partial\beta_i\partial\beta_j} = \sum_{l=1}^{N} 2g(\mathbf{w}_i\cdot\mathbf{x}_l + b_i)g(\mathbf{w}_j\cdot\mathbf{x}_l + b_j), \quad (22)$$

where $i = 1, \ldots, \widetilde{N}$ and $j = 1, \ldots, \widetilde{N}$. $\mathbf{f}$ is a column vector consists of monomial coefficients in terms of β_i $(1 \le i \le \widetilde{N})$ in the cost function. Its expression is

$$\mathbf{f} = \left[\sum_{i=1}^{N} 2t_i g(\mathbf{w}_1\cdot\mathbf{x}_i + b_1),\ \sum_{i=1}^{N} 2t_i g(\mathbf{w}_2\cdot\mathbf{x}_i + b_2), \ldots,\ \sum_{i=1}^{N} 2t_i g(\mathbf{w}_{\widetilde{N}}\cdot\mathbf{x}_i + b_{\widetilde{N}})\right]^T. \quad (23)$$

$\mathbf{A}$ is a matrix consists of coefficients in terms of β_i $(1 \le i \le \widetilde{N})$ in the inequality constraints which represent the monotonicity restrictions. The number of columns in $\mathbf{A}$ is $\widetilde{N}$, which equals to the number of hidden nodes. Suppose there are r $(1 \le r \le n)$ features that are monotonically increasing or decreasing with the decision label, and each of these features corresponds to $\widetilde{N}$ constraints. So the number of rows in matrix $\mathbf{A}$ is $r \times \widetilde{N}$. Suppose the *i*th feature x_i of samples is monotonically increasing with the decision label, that is $\frac{\partial o}{\partial x_i} > 0$, where $1 \le i \le n$, then the rows responding to this monotonicity constraint in matrix $\mathbf{A}$ is

$$\begin{bmatrix} -w_{1i}, 0, 0, \ldots, 0\\ 0, -w_{2i}, 0, \ldots, 0\\ \cdots\cdots\\ 0, 0, 0, \ldots, -w_{\widetilde{N}i}\end{bmatrix}. \quad (24)$$

Similarly, if x_i $(1 \le i \le n)$ is monotonically decreasing with the decision label, that is $\frac{\partial o}{\partial x_i} < 0$ $(1 \le i \le n)$ then the rows responding to this monotonicity constraint in matrix $\mathbf{A}$ is

$$\begin{bmatrix} w_{1i}, 0, 0, \ldots, 0\\ 0, w_{2i}, 0, \ldots, 0\\ \cdots\cdots\\ 0, 0, 0, \ldots, w_{\widetilde{N}i}\end{bmatrix}. \quad (25)$$

$\mathbf{B}$ is a matrix of which the elements are constant terms on the right side of the '≤' sign in the inequality constraints (18). In the mathematical model of MCELM, $\mathbf{B}$ is a zero column vector because all the constant terms in the monotonicity constraints are zero.

Mathematically, if the Hessian matrix of the cost function is positive semidefinite, then the quadratic programming is convex quadratic. In this situation, if there is at least one vector which satisfies the constraints and has lower bound in the feasible region, then a global minimum of the quadratic programming exists. If the Hessian matrix is positive definite, then the quadratic programming is strictly convex and the global minimum is unique. If the Hessian matrix is indefinite, then the quadratic programming is nonconvex which is more challenging to be solved because there are a few local minimum points.

Therefore, in the mathematical model of MCELM, if the objective function is strictly convex or convex, and the feasible region exists, then the global optimal solution can be obtained. However, in this work, we have not discussed the existing condition of the feasible region. The solution of $\widehat{\boldsymbol{\beta}}$ in the quadratic programming model is solved by using the tool box of Matlab, which can ensure the generated classifier performs well with less classification error. On the other hand, the classifier is monotonic due to the monotonicity constraints. Besides, during the solving process of the quadratic programming, the gradient descent method is not involved. All parameters of MCELM do not need to be tuned iteratively, so MCELM is much faster than the monotonic classification methods which are based on conventional artificial neural networks. Moreover, MCELM does not require that the monotonic relationships existing between features and the output are consistent, which essentially relaxes the assumption of consistent monotonicity used in most existing approaches to handling monotonic classification problems.

3.3. The algorithm of MCELM

The MCELM learning algorithm described here is implemented by using Matlab where there is a function called quadprog to solve quadratic programming problem. The call format of function quadprog is

$$X = quadprog(\mathbf{C}, \mathbf{f}, \mathbf{A}, \mathbf{B}), \quad (26)$$

where $\mathbf{C}$ is the Hessian matrix of the cost function; $\mathbf{f}$ is a vector consists of monomial coefficients in terms of β_i $(1 \le i \le \widetilde{N})$ in the cost function; $\mathbf{A}$ is a matrix consists of coefficients in terms of β_i $(1 \le i \le \widetilde{N})$ in the inequality constraints which represent the monotonicity restrictions; $\mathbf{B}$ is a matrix of which the elements are constant terms on the right side of the '≤' sign in the inequality constraints. In the mathematical model of MCELM, $\mathbf{B}$ is a zero vector because all the constant terms in the monotonicity constraints are zero. All the values of parameters $\mathbf{C}$, $\mathbf{f}$, $\mathbf{A}$, $\mathbf{B}$ can be obtained based on their expressions which are proposed in the above section.

The MCELM learning algorithm based on Matlab can be summarized as follows:

- **Input:** training set $\aleph = \{(\mathbf{x}_i, t_i) | \mathbf{x}_i \in \mathbf{R}^n, t_i \in \mathbf{R}, i = 1, \ldots, N\}$ ($\mathbf{R}$ is a set of real numbers), activation function $g(x)$, and hidden node number $\widetilde{N}$.
- **Output:** parameters $\mathbf{w}$, $\mathbf{b}$ and $\boldsymbol{\beta}$.
  - (Step-1:) Randomly assign input weight $\mathbf{w}_i$ and bias b_i, $i = 1, \ldots, \widetilde{N}$.
  - (Step-2:) Based on the expressions of $\mathbf{C}$, $\mathbf{f}$, $\mathbf{A}$ and $\mathbf{B}$, obtain the values of parameters $\mathbf{C}$, $\mathbf{f}$, $\mathbf{A}$, $\mathbf{B}$. Solve the quadratic programming problem by using the *quadprog* function in Matlab and get the

solution of $\boldsymbol{\beta}$.

- (Step-3:) Return $\mathbf{w}$, $\mathbf{b}$ and $\boldsymbol{\beta}$.

Remark 1. The activation function $g(x)$ in the MCELM algorithm is sigmoidal function.

Remark 2. There is only one neuron in the output layer of MCELM, and among the multiclass labels, the class label which is closest to the output value is chosen as the predicted class label of the input data.

Remark 3. The algorithm of MCELM based on Matlab is designed under the assumption that the feasible region of the quadratic programming exists. If the feasible region exists, we can get the global optimal solution which can minimize the cost function and preserve the monotonicity between features and the decision label.

4. Performance evaluation

In this section, we will experimentally compare MCELM with CART [20], Rank Tree [34], OLM [8], OSDL [58], REMT [36] and ELM [49] based on both artificial datasets and real world datasets. OLM is an ordinal learning model introduced by Ben-David et al., while OSDL is an ordinal stochastic dominance learner based on associated cumulative distribution [58].

In traditional classification problems, we usually use the misclassification rate (divide the number of misclassified samples by the number of test samples) to measure the performance of a classifier. However, it is not efficient in monotonic classification problems. In monotonic classifications, the class labels are ordered, so the deviation of the predicted output and the actual output should be emphasized, while, the misclassification rate is not feasible from this point of view.

In this work, we use the Mean Absolute Error (MAE) to measure the performance of classifiers. The expression of MAE is shown as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|, \tag{27}$$

where N is the number of samples in the test set; $\hat{y}_i$ is the predicted output of the ith sample and y_i $(i = 1, 2, \ldots, N)$ is its actual output[36]. This criterion takes the deviation degree into account and thus it is more suitable than the misclassification rate.

4.1. Artificial case

We use the following function to generate monotonic datasets:

$$f(x_1, x_2) = x_1 + \frac{1}{2}(x_2^2 - x_1^2), \tag{28}$$

where x_1 and x_2 represent inputs which are randomly assigned from the uniform distribution over the unit interval.

We split the unit interval $[0, 1]$ into k sub intervals $[0, 1/k], (1/k, 2/k], \ldots, (k-1)/k, 1]$, where k is the number of categories. If the function value $f(x_1, x_2)$ falls into the ith sub interval, then the class label of sample (x_1, x_2) is set to be i $(1 \le i \le k)$.

According to the above approach, we generate several monotonic datasets with different number of categories. Each dataset contains 1000 samples. Based on these datasets, we compare MCELM with CART, Rank Tree, OLM, OSDL, REMT and ELM by using the measure criterion of MAE.

First we evaluate the sensitivity of these algorithms to the number of categories. We do the experiment based on the 5-fold cross validation technique. The comparison result is represented in Fig. 1.

Generally for training classifiers, the larger the number of categories is, the bigger the classification error will be. Therefore, all the curves in Fig. 1 tend to be ascending when the number of categories continues to increase; however, the MCELM algorithm performs much better than any other methods, which indicates that the MCELM can ensure the generated classifier is a monotonic one with less classification error.

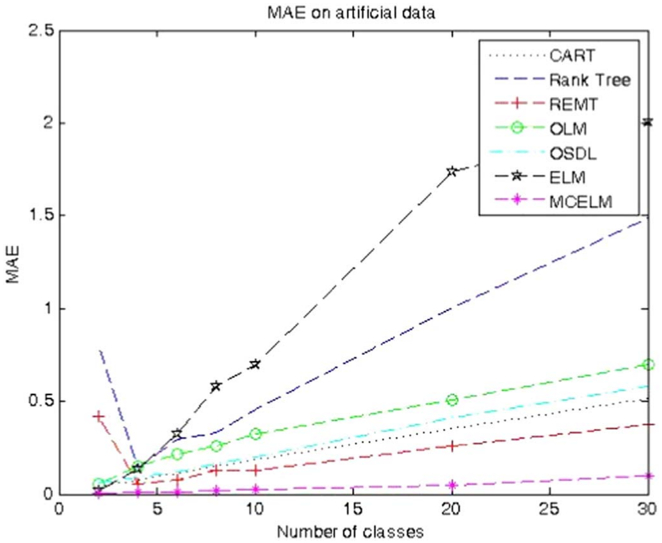

Fig. 1. MAE of CART, Rank Tree, REMT, OLM, OSDL, ELM and MCELM, where the number of classes gradually increases.

Then we estimate the influence of the number of training samples on the performances of the trained models. We first generate a data set with 1000 samples belonging to 4 classes, then extract training samples randomly from each class of the data set. The number of training samples ranges from 4 to 36. The remaining samples are used as testing samples to evaluate the performances of the trained models. The curves of MAE varying with the number of training samples are shown in Fig. 2.

Fig. 2 shows that no matter how many training samples are used, MCELM is much more precise than CART, Rank Tree, REMT and ELM, which is due to the classifier trained by MCELM is a monotonic one with less classification error. The result shows that MCELM is not sensitive to the number of training samples, so when the training set of a monotonic classification task is small in size, we can adopt MCELM to get a more precise classifier.

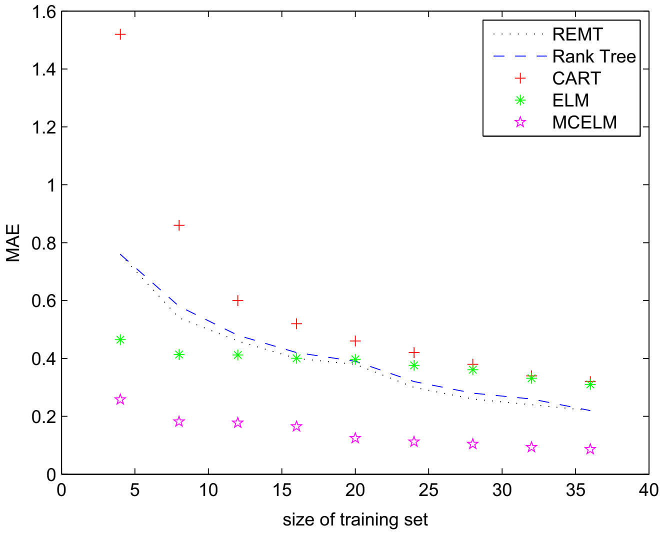

Fig. 2. MAE curves of CART, Rank Tree, REMT, ELM and MCELM, where the size of training samples gradually increases.

Table 1
Real data sets used in the experiment.

Data set	Attributes	Instances	Classes
SWD	10	1000	4
ESL	4	488	9
LEV	4	1000	5
Car	6	1728	4
Breast cancer	9	699	2

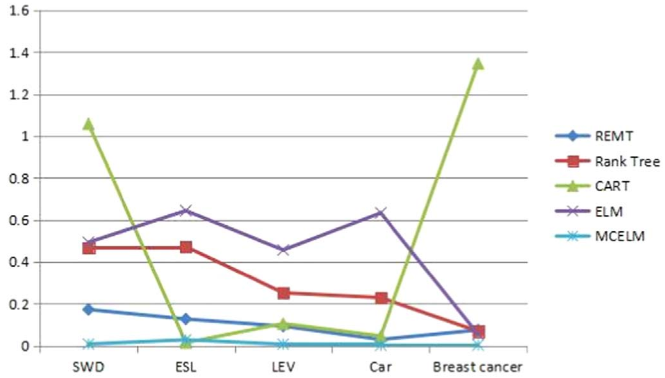

Fig. 3. MAE of REMT, Rank Tree, CART, ELM and MCELM on different real datasets.

4.2. *Real world data sets*

We have verified the high validity of MCELM on the artificial data. Next we will evaluate its performance on five real world datasets with monotonicity constraints. Three of these datasets SWD, ESL and LEV are obtained from the weka homepage (http://www.cs.waikato.ac.nz/ml/weka/) provided by [8]. The other two datasets Car and Breast cancer Wisconsin come from UCI repository [59]. The characteristics of all the datasets are shown in Table 1. In the Breast cancer Wisconsin dataset, there are 16 samples with missing attribute values, by removing which, we process this dataset. We compare the performances of REMT, Rank Tree, CART, ELM and MCELM on the five datasets by using 5-fold cross validation technique. The measure index is MAE. Results are shown in Fig. 3.

From Fig. 3 we know that MCELM performs much better than REMT, Rank Tree, CART and ELM on the collected real datasets, because the classifier generated by MCELM is a monotonic one with less classification error.

From the experimental results presented in Sections 4.1 and 4.2, we can see that MCELM performs much better than any other methods mentioned in this paper for solving monotonic classification problems. This phenomenon appears because in our MCELM model monotonic constraints theoretically guarantee that the learned ELM can keep the monotonicity in the whole domain. The other methodologies mentioned in this paper cannot analytically guarantee the monotonicity in the whole domain. We think of that for building a monotonic classifier the monotonicity condition should be first met, which is the essential point of the learning. This conclusion is confirmed experimentally. Training a regular ELM can be transferred into a non-constrained optimization problem while our model for training an ordered ELM can be transferred into a constrained optimization problem. Viewing our model has an excellent performance, we can think of that the imposed constrains play a critical role.

5. Concluding remarks

Monotonic classification essentially is an ordinal classification with monotonicity constrains. The monotonicity between a feature and the decision attribute may be increasing or decreasing. Several methods have been proposed to solve monotonic classification problems, such as Rank Tree, OLM and REMT. Although these methods enhance the capability of extracting ordinal information, they suffer from many shortcomings. Most of them cannot guarantee the generated classifiers are monotonic ones with good classification performances. Moreover, some of these algorithms get stuck in information loss and some suffer from time consuming. In order to overcome those disadvantages, we propose MCELM on which some comments are listed below.

1. MCELM can ensure the generated classifier is monotonic. Although the existing methods such as Rank Tree and OLM can extract ordinal information, they cannot ensure that a monotonic classifier is obtained when the training data are monotonic. ELM is a single-hidden layer feedforward neural network with fast training rate and good generalization capability, but due to the existence of training error, ELM cannot be directly used to handle monotonic classification problems. However, the classifier generated under the restriction of monotonicity constraints in MCELM is monotonic.
2. The classifier generated by MCELM performs well with very little classification error. In the REMT [36] algorithm, the classification error has not been considered. In comparison with REMT for constructing a monotonic decision tree where the rank mutual information is used as a heuristic, our MCELM model is a quadratic programming problem which can take both monotonicity and the classification error into account. MCELM shows an accuracy improvement.
3. MCELM basically avoids information loss. In some of the monotonic classification methods, all the monotonicity constraints are limited to be consistent, which means that all the monotonicity relationships are either increasing or decreasing; otherwise the data have to be transformed, which may cause information loss. In MCELM, the monotonicity constraints are represented by restricting the signs of the partial derivatives of the decision attribute with respect to the features. It is not necessary to transform the data when using MCELM.
4. MCELM is much faster than conventional artificial neural networks which are based on the gradient-descent technique. Similar to ELM, the input weights and hidden layer biases can be randomly assigned when the activation functions are infinitely differentiable. The output weights can be analytically determined. Different from the conventional artificial neural networks, the gradient descent method is not adopted in MCELM. All the parameters in MCELM do not need to be tuned iteratively.
5. Experimental results have shown that MCELM has much better generalization capability than other approaches when they are applied to handling a monotonic classification problem.

Acknolwdedgement

This work is supported by the Macao Science and Technology Development Funds (100/2013/A3 and 081/2015/A3), China Postdoctoral Science Foundations (2015M572361 and 2016T90799), Basic Research Project of Knowledge Innovation Program in Shenzhen (JCYJ20150324140036825), and National Natural Science Foundations of China (61503252, 61473194, and 71371063).

References

[1] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[2] I. Kononenko, Naive and non-naive Bayesian classifier, Elektroteh. Vestn. 57 (5) (1990) 326–330.
[3] J.S. Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.
[4] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[5] A.K. Jain, J. Mao, K.M. Mohiuddin, Artificial neural networks: a tutorial, Computer

29 (3) (1996) 31–44.

[6] Xizhao Wang. Learning from big data with uncertainty-Editorial. Journal of Intelligent & Fuzzy Systems, 2015, 28(5): 2329-2330

[7] Xizhao Wang, RAR Ashfaq, Aimin Fu. Fuzziness based sample categorization for classifier performance improvement. Journal of Intelligent & Fuzzy Systems, 2015: 1-12.

[8] A. Ben-David, L. Sterling, Y.H. Pao, Learning and classification of monotonic ordinal concepts, Comput. Intell. 5 (1989) 45–49.

[9] S. Greco, B. Matarazzo, R. Slowinski, Rule-based decision support in multicriteria choice and ranking. symbolic and quantitative approaches to reasoning with uncertainty. in: Proceedings of the 6th European Conference, ECSQARU, Lecture Notes in Artificial Intelligence, vol. 2143, 2001, pp 29–47.

[10] Z. Pawlak, Rough sets, Int. J. Comput. Inf. Sci. 11 (5) (1982) 341–356.

[11] J.C. Bioch, V. Popova, Rough sets and ordinal classification, algorithmic learning theory, in: Proceedings of the 11th International Conference, ALT 2000 Sydney, Australia, December 11–13, vol. 1968, 2000, pp. 291–305.

[12] V. Popova, Knowledge discovery and monotonicity. Rotterdam, Nederland: Erasmus Research Institute of Management (ERIM) Rotterdam School of Management/ Rotterdam School of Economics Erasmus University Rotterdam, 2004, pp. 1–157.

[13] S. Greco, B. Matarazzo, R. Slowinski, Rough approximation of a preference relation by dominance relations, Eur. J. Oper. Res. 117 (1) (1999) 63–83.

[14] S. Greco, B. Matarazzo, R. Slowinski, Rough sets methodology for sorting problems in presence of multiple attributes and criteria, Eur. J. Oper. Res. 138 (2) (2002) 247–259.

[15] S. Greco, B. Matarazzo, R. Slowinski, Rough approximation by dominance relations, Int. J. Intell. Syst. 17 (2) (2002) 153–171.

[16] J.W.T. Lee, D.S. Yeung, E.C.C. Tsang, Rough sets and ordinal reducts, Soft Comput. 10 (1) (2005) 27–33.

[17] Q. Hu, D. Yu, M. Guo, Fuzzy preference based rough sets, Inf. Sci. 180 (10) (2010) 2003–2022.

[18] A. Ben-David, L. Sterling, T. Tran, Adding monotonicity to learning algorithms may impair their accuracy, Expert Syst. Appl. 36 (3) (2009) 6627–6634.

[19] W. Kotlowski, R. Slowinski, On nonparametric ordinal classification with monotonicity constraints, IEEE Trans. Knowl. Data Eng. 25 (11) (2013) 2576–2589.

[20] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, Chapman and Hall/CRC, London, the United Kingdom, 1984, pp. 1–368.

[21] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, California, the United States of America, 1993, pp. 1–299.

[22] Ran Wang, Sam Kwong, Xizhao Wang, Qingshan Jiang. Segment based decision tree induction with continuous valued attributes. IEEE Transactions on Cybernetics, 2015, 45(7): 1262-1275

[23] A. Ben-David, Monotonicity maintenance in information-theoretic machine learning algorithms, Mach. Learn. 19 (1) (1995) 29–43.

[24] S. Makino, et al., Cardiomyocytes can be generated from marrow stromal cells in vitro, J. Clin. Investig. 103 (5) (1999) 697–705.

[25] R. Potharst, J.C. Bioch, T.C. Petter, Monotone decision trees. Technical report, Erasmus University Rotterdam, 1997.

[26] R. Potharst, Classification using decision trees and neural networks, Ph. D. dissertation, Erasmus University Rotterdam, 1999.

[27] R. Potharst, J. Bioch, Decision trees for ordinal classification, Intell. Data Anal. 4 (2) (2000) 97–112.

[28] R. Potharst, A.J. Feelders, Classification trees for problems with monotonicity constraints, SIGKDD Explor. 4 (1) (2002) 1–10.

[29] J.C. Bioch, V. Popova, Monotone decision trees and noisy data, Technical report, Erasmus University Rotterdam, 2002.

[30] R. Potharst, J.C. Bioch, Decision trees for ordinal classification, Intell. Data Anal. 4 (2000) 97–111.

[31] A.J. Feelders, MartijnPardoel. Pruning for monotone classification trees, Lecture Notes in Computer Science, 2811, 2003, pp. 1–12.

[32] K. Cao-Van, B.De. Baets, Growing decision trees in an ordinal setting, Int. J. Intell. Syst. 18 (2003) 733–750.

[33] C.-V. Kim, Supervised ranking from semantics to algorithms. (Ph.D. thesis), Ghent University, Belgium, 2003.

[34] F. Xia, W. Zhang, F. Li, Y. Yang, Ranking with decision tree, Knowl. Inf. Syst. 17 (3) (2008) 381–395.

[35] R.V. Kamp, A.J. Feelders, N. Barile, Isotonic classification trees, in: N. Adams (Ed.), Proceedings of IDA, Springer, 2009, pp. 405–416.

[36] Q. Hu, X. Che, L. Zhang, et al., Rank entropy based decision trees for monotonic classification, IEEE Trans. Knowl. Data Eng. 24 (11) (2012) 2052–2064.

[37] Xiaoping Li, Dan Li. The structure and realization of a polygonal fuzzy neural network. International Journal of Machine Learning and Cybernetics, 2016, 7(3): 375-389

[38] Xiaojuan Jiang, Wensheng Zhang. Structure learning for weighted networks based on Bayesian nonparametric models. International Journal of Machine Learning and Cybernetics, 2016, 7(3): 479-489

[39] Junhai Zhai, Xizhao Wang, Xiaohe Pang. Voting-based Instance Selection from Large Data Sets with MapReduce and Random Weight Networks, Information Sciences 367¨C368 (2016) 1066¨C1077

[40] Wei Kang, Shouming Zhong, Jun Cheng. Relaxed passivity conditions for discrete-time stochastic delayed neural networks. International Journal of Machine Learning and Cybernetics, 2016, 7(2): 205-216

[41] N.P. Archer, S. Wang, Application of the back propagation neural network algorithm with monotonicity constraints for Two-Group classification problem, Decis. Sci. 24 (1) (2007) 60–75.

[42] C. Li, J. Yi, M. Wang, G. Zhang, Monotonic type-2 fuzzy neural network and its application to thermal comfort prediction, Neural Comput. Appl. 23 (2013) 1987–1998.

[43] C. Li, J. Yi, G. Zhang, On the monotonicity of interval type-2 fuzzy logic systems, IEEE Trans. Fuzzy Syst. 22 (5) (2014) 1197–1212.

[44] C. Cortes, V. Vapnik. Support vector networks. Mach. Learn., 1995, 20(3): 273-297

[45] Shigeo Abe. Fusing sequential minimal optimization and Newton¡¯s method for support vector training. International Journal of Machine Learning and Cybernetics, 2016, 7(3): 345-364

[46] Zhi-Min Yang, He-Ji Wu, Chun-Na Li, Yuan-Hai Shao. Least squares recursive projection twin support vector machine for multi-class classification. International Journal of Machine Learning and Cybernetics, 2016, 7(3): 411-426

[47] Xinjun Peng, Dongjing Chen, Lingyan Kong, Dong Xu. Interval twin support vector regression algorithm for interval input-output data. International Journal of Machine Learning and Cybernetics, 2015, 6(5): 719-732

[48] Shifei Ding, Xiekai Zhang, Junzhao Yu. Twin support vector machines based on fruit fly optimization algorithm. International Journal of Machine Learning and Cybernetics, 2016, 7(2): 193-203

[49] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.

[50] N. Liang, P. Saratchandran, G. Huang, N. Sundararajan, Classification of mental tasks from EEG signals using extreme learning machine, Int. J. Neural Syst. 16 (1) (2006) 29–38.

[51] H. Rong, G. Huang, Y. Ong, Extreme learning machine for multicategories classification applications, Neural network, 2008, in: Proceedings of the IEEE International Joint Conference on, IJCNN 2008, (IEEE World Congress on Computational Intelligence) 1–8, 2008, Hong Kong, pp. 1709–1713.

[52] G. Huang, L. Chen, C. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Netw. 17 (4) (2006) 879–892.

[53] G. Huang, Q. Zhu, C. Siew, Real-time learning capability of neural networks, IEEE Trans. Neural Netw. 17 (4) (2006) 863–878.

[54] G. Huang, H. Babri, Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions, IEEE Trans. Neural Netw. 9 (1) (1998) 224–229.

[55] G. Huang, Learning capability and storage capacity of two-hidden-layer feedforward networks, IEEE Trans. Neural Netw. 14 (2) (2003) 274–281.

[56] D. Serre, Matrices: Theory and Applications, Springer, New York, 2002.

[57] C.R. Rao, S.K. Mitra, Generalized Inverse of Matrices and its Applications, Wiley, New York, 1971.

[58] K. Cao-Van, Supervised Ranking from Semantics to Algorithms [D], 2003.

[59] A. Asuncion, D. Newman. ?UCI Machine Learning Repository,? ⟨http://www.ics.uci.edu/mlearn/MLRepository.html⟩, 2007.

Hong Zhu received the B.Sc. and M.Sc. degrees from Hebei University, Baoding, China, in 2012 and 2015, respectively. She is currently a Ph.D. candidate of the Faculty of Information Technology, Macao University of Science and Technology. Her main research interests include decision tree and neural networks.

Eric C.C. Tsang received his B.Sc. degree in Computer Studies from the City University of Hong Kong in 1990 and Ph.D. degree in computing at the Hong Kong Polytechnic University in 1996. He is an associate professor of the Faculty of Information Technology, Macao University of Science and Technology. His main research interests include fuzzy systems, rough sets, fuzzy rough sets and multiple classifier systems.

Xi-Zhao Wang (M′03-SM'04-F′12) received the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1998. He is currently a Professor with the College of Computer Science and Software Engineering, Shenzhen University, Guangdong, China. From September 1998 to September 2001, he was a Research Fellow with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His current research interests include learning from examples with fuzzy representation, fuzzy measures and integrals, neuro fuzzy systems and genetic algorithms, feature extraction, multiclassifier fusion, and applications of machine learning. Dr. Wang has been the PI/Co-PI for 16 research projects supported partially by the National Natural Science Foundation of China and the Research Grant Committee of Hong Kong Government. He was the BoG Member of the IEEE Systems, Man, and Cybernetics Society (IEEE SMCS) in 2005, 2007–2009, and 2012–2014; the Chair of the IEEESMC Technical Committee on Computational Intelligence, an Associate Editor of IEEE TRANSACTIONS ON CYBERNETICS; an Associate Editor of Pattern Recognition and Artificial Intelligence; a Member of the Editorial Board of Information Sciences; and an Executive Member of the Chinese Association of Artificial Intelligence. He was the recipient of the IEEE SMCS Outstanding Contribution Award in 2004 and the recipient of IEEE SMCS Best Associate Editor Award in 2006. He is the general Co-Chair of the 2002–2014 International Conferences on Machine Learning and Cybernetics, cosponsored by IEEE SMCS. He is a Distinguished Lecturer of the IEEE SMCS.

Rana Aamir Raza Ashfaq received his Master degree in Computer Science from Blekinge Tekniska Hogskola (BTH), Sweden. He also received his Bachelor and Master Degrees in Computer Science from Bahauddin Zakariya University, Multan, Pakistan. Since 2010 he is working as Assistant Professor in Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan. He is currently a Ph.D. candidate in College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, Guangdong, China. His main research interests include machine learning and data mining.