DE+RBFNs based classification: A special attention to removal of inconsistency and irrelevant features

Ch. Sanjeev Kumar Dash, Aditya Prakash Dash, Satchidananda Dehuri, Sung-Bae Cho, Gi-Nam Wang

ARTICLE INFO

Article history:
Received 2 January 2013
Received in revised form
7 May 2013
Accepted 23 August 2013
Available online 21 September 2013

Keywords:
Classification
Radial basis function neural networks
Differential evolution
Feature selection
Dataset consistency

ABSTRACT

A novel approach for the classification of both balanced and imbalanced dataset is developed in this paper by integrating the best attributes of radial basis function networks and differential evolution. In addition, a special attention is given to handle the problem of inconsistency and removal of irrelevant features. Removing data inconsistency and inputting optimal and relevant set of features to a radial basis function network may greatly enhance the network efficiency (in terms of accuracy), at the same time compact its size. We use Bayesian statistics for making the dataset consistent, information gain theory (a kind of filter approach) for reducing the features, and differential evolution for tuning center, spread and bias of radial basis function networks. The proposed approach is validated with a few benchmarked highly skewed and balanced dataset retrieved from University of California, Irvine (UCI) repository. Our experimental result demonstrates promising classification accuracy, when data inconsistency and feature selection are considered to design this classifier.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

Classification is one of the fundamental tasks in data mining (Dehuri and Ghosh, 2013) and pattern recognition (Nanda and Panda, 2013). Over the years many models have been proposed. (Huang and Wang, 2006; Chatterjee and Bhattacharyee, 2011) However, it is a consensus that the accuracy of the discovered model (i.e., neural networks (NNs) (Haykin, 1994; Yaghini et al., 2013), rules (Das et al., 2011), and decision tree (Carvalho and Freitas, 2004)) strongly depends on the quality of the data being mined. Hence inconsistency removal and feature selection brings lots of attention of many researchers (Battiti, 1994; Yan et al., 2008; Keunia, 2012; Ebrahimzadeh and Ghazalian, 2011; Liu et al., 2010). If the inconsistent data are simply deleted or classified as a new category then inevitably some useful information will be lost. The method used in this paper for making the dataset consistent is based on the Bayesian statistical method (Wu, 2007). Here the inconsistent data are classified as the most probable one and the redundant data records are deleted as well. So the loss of information due to simple deletion or random classification of inconsistent data is reduced and the size of the dataset is also reduced.

Feature selection is the process of selecting a subset of available features to use in empirical modeling. Like feature selection, instance selection (Liu and Motoda, 2002) is to choose a subset sample to achieve the original purpose of a classification task, as if the whole dataset is used. Many variants of evolutionary and non-evolutionary based approaches are discussed in Derrac et al. (2010). The ideal outcome of instance selection is a model independent, minimum sample of data that can accomplish tasks with little or no performance deterioration. Unlike feature selection and instance selection, feature extraction at feature level fusion recently attracts data mining/machine learning researchers (Liu et al., 2011) to give special focus while designing a classifier. However, in this work, we restrict ourselves with feature selection and inconsistency removal only.

Feature selection can be broadly classified into two categories: (i) filter approach (it depends on generic statistical measurement), and (ii) wrapper approach (based on the accuracy of a specific classifier) (Aruna et al., 2012). In this work, the feature selection is performed based on information gain theory (entropy) measure with a goal to select a subset of features that preserve the relevant information found in the entire set of features as much as possible. After the selection of the relevant set of features the fine tuned
radial basis function network is modeled using differential evolution for the classification of both balanced and unbalanced datasets. In imbalance classification problems the number of instances of each class that occur can be very different (Perez-Godoy et al., 2010).

Over the decade radial basis function (RBF) networks have attracted a lot of interest in various domains of interest (Haykin, 1994; Novakovic, 2011; Naveen et al., 2010; Liu et al., 2005). The reason is that they form a unifying link between function approximation, regularization, noisy interpolation, classification, and density estimation. Moreover, training RBF networks is usually faster than training multi-layer perceptron networks. RBF network training usually proceeds in two steps: first, the basis function parameters (corresponding to hidden units) are determined by clustering. Second, the final-layer weights are determined by a least square method which reduces to solve a simple linear system. Thus, the first stage is an unsupervised method which is relatively fast, and the second stage requires the solution of a linear problem, which is also fast.

The other advantages of RBF neural networks, compared to multi-layer perceptron networks, is the possibility of choosing suitable parameters for the units of hidden layer without having to perform a nonlinear optimization of the network parameters. However, the problem of selecting the appropriate number of basis functions remains a critical issue for RBF networks. The number of basis functions controls the complexity, and hence the generalization ability of RBF networks. An RBF network with too few basis functions gives poor predictions on new data, i.e., poor generalization, since the model has limited flexibility. On the other hand, an RBF network with too many basis functions also yields poor generalization since it is too flexible and fits the noise in the training data. A small number of basis functions yield a high bias, low variance estimator, whereas a large number of basis functions yield a low bias but high variance estimator. The best generalization performance is obtained via a compromise between the conflicting requirements of reducing bias while simultaneously reducing variance. This trade-off highlights the importance of optimizing the complexity of the model in order to achieve the best generalization. However, choosing an optimal number of kernels is beyond the focus of this paper.

In the training procedure of RBFNs revealing center of gravity and width is of particular importance for the improvement of the performance of the networks. There are many approaches along the line with their own merits and demerits (Stron and Price, 1995, 1997; Price et al., 2005). This paper discusses the use of differential evolution to reveal hidden centers and spreads. The motivation using differential evolution (DE) over other EAs (Michalewicz, 1996) such as GAs (Goldberg, 1989) is that in DE string encoding is typically represented as real valued vectors, and the perturbation of solution vectors is based on the scaled difference of two randomly selected individuals of the current population. Unlike GA, the resulting step size and orientation during the perturbation process automatically adopt to the fitness function landscape. The justification behind combining the idea of feature selection, data inconsistency removal with classification is to reduce the space, time, and thereby enhancing accuracy.

This paper is set out as follows. Section 2 gives an overview of the RBF network, feature selection, feature consistency, and differential evolution. In Section 3, the proposed method is discussed. Experimental setup, results, and analysis are presented in Section 4. Section 5 concludes the paper with a future line of research.

2. Background

The background of this research work is presented in this section. In Section 2.1, RBF network classifier is discussed. Feature selection and its importance are the focus of Section 2.2. Feature inconsistency and the method used in this paper is the topic of discussion in Section 2.3. Differential evolution of a new meta-heuristic computing paradigm is discussed in Section 2.4.

2.1. RBF network classifier

In the RBF network a supervised learning algorithm is adopted to build novel and potentially useful models (Naveen et al., 2010). RBF network is a popular artificial neural network architecture that has found wide applications in diverse fields of engineering. It is used in pattern recognition, function approximation, and time series prediction. Further, it is the most widely used alternative neural network model with certain advantages over multi-layered feed forward neural network (MFN) for the task of pattern classification.

RBF network is a three layer feed forward network where each hidden unit implements a radial activation function and each output unit implements a weighted sum of hidden units’ output. This network is a special class of neural network in which the activation of a hidden neuron is determined by the distance between the input vector and a prototype vector. Prototype vectors refer to centers of clusters formed by the patterns or vectors in the input space. The centers are determined during RBF training. Fig. 1 shows a typical RBF network.

In the input layer ‘n’ number of input neurons exist that connect the network to the environment. The second layer consists of a set of kernel units that carry out a nonlinear transformation from the input space to the hidden space. Some of the commonly used kernel functions are defined in Table 1. Usually, a nonlinear transformation is made based on Gaussian kernel as described in the following equation:

$$\phi_i(x) = \exp\left(-\frac{||x-\mu_i||^2}{2\sigma_i^2}\right).$$

where $||.||$ represents Euclidean norm, $\mu_i$, $\sigma_i$, and $\phi_i$ are center, spread and the output of ith hidden unit, respectively. The interconnection between the hidden and the output layers forms weighted connection.
connections \( w_i \). The output layer, a summation unit, supplies the response of the network to the outside world.

The radial basis function is so named because the value of the function is same for all points which are at the same distance from the center.

### 2.2. Feature selection

Feature selection (FS) (Novakovic, 2011; Khusba et al. 2008) is an essential task to remove irrelevant and/or redundant features. In other words, feature selection techniques study how to select a subset of potential attributes or variables from a dataset. (Liu and Setiono, 1995). For a given classification problem, the network may become extremely complex if the number of features used to classify the pattern increases. So the reason behind using FS becomes extremely complex if the number of features used to increase the constructed model’s comprehensibility. After feature selection a subset of the original features is obtained which retains sufficient information to discriminate well among classes.

The selection of features can be achieved in two ways (Yu and Liu, 2004):

**Filter method:** It precedes the actual classification process. The filter approach is independent of the learning algorithm, computationally simple, fast, and scalable. Using filter method, feature selection is done once and then can be provided as inputs to different classifiers. In this method features are ranked according to some criterion and the top \( k \) features are selected.

**Wrapper method:** This approach uses the method of classification itself to measure the importance of feature sets; hence the selected feature depends on the classifier model used (Karegowda et al., 2010). In this method a minimum subset of features is selected without learning performance deterioration.

Wrapper methods generally result in a better performance than filter methods because the feature selection process is optimized for the classification algorithm to be used. However, wrapper methods are too expensive for large dimensional database in terms of computational complexity and time since each feature set that is considered must be evaluated with the classifier algorithm used. Filter based feature selection methods are in general faster than wrapper based methods.

### 2.3. Dataset consistency

Let us define a few terms pertaining to data inconsistency. Let us define a few terms pertaining to data inconsistency (Shin and Xu, 2009; Dash, et al., 2000; Dash and Liu, 2003; Arauzo-Azofra et al., 2008).

**Definition:** a dataset is said to be consistent if it does not contain any inconsistent instances or patterns.
inconsistent. To make the dataset uniform all inconsistent data subsets are made uniform.

Step 5: the most probable class for the inconsistent pattern is found out in this step. The probability of a pattern \( p_i \) belonging to category \( c_0 \) is given by the following equation:

\[
P(c_0|p_i) = \frac{P(p_i|c_0)P(c_0)}{P(p_i)}.
\]

where \( P(c_0) = \frac{|c_0|}{|D|} \), \( P(p_i|c_0) = \frac{|d_{mn} \cap c_0|}{|c_0|} \), \( P(p_i) = \frac{|d_{mn}|}{|D|} \).

The most probable class for the pattern \( p_i \) is the one for which \( P(c_0|p_i) \) is maximum. But if the class distribution of a pattern is quite even then the above probability may give a wrong most probable class. For example, if a pattern \( P_i \) has three class values \( c_1, c_2, \) and \( c_3 \) then the corresponding probability is 0.34, 0.33, and 0.33 respectively. In this case it is inappropriate to classify the pattern as \( c_1 \). A situation a threshold \( \alpha \) is introduced. So, only when \( P(c_0|p_i) > P(c_0|p_j) \) and \( P(c_0|p_i) > \alpha \), then the pattern \( p_i \) is classified as \( c_0 \).

If the probability for all classes for a pattern is below the threshold then the inconsistent data subset containing this pattern is deleted from the dataset. Taking \( \alpha = 0.5 \), the inconsistent data subsets \( d_1 \) and \( d_3 \) can be unified as \( c_1 = 1 \).

### 2.4. Differential evolution

Differential evolution (DE) (Storn and Price, 1995; Storn and Price, 1997; Storn, 1999) is a population based meta-heuristic search algorithm which typically operates on real valued chromosome encodings. Like GAs (Forerest, 1993), DE maintains a pool of potential solutions which are then perturbed in an effort to uncover yet better solutions to a problem in hand. In GAs, the individuals are perturbed based on crossover and mutation. However in DE, individuals are perturbed based on the scaled difference of two randomly chosen individuals of the current population. One of the advantages of this approach is that the resulting ‘step’ size and orientation during the perturbation process automatically adapts to the fitness function landscape.

Over the years, there are many variants of DE algorithms developed (Price et al., 2005; Das and Suganthan, 2011), however, we primarily describe a version of the algorithm based on the DE/rand/1/bin scheme (Storn et al., 1995). The different variants of the DE algorithm are described using the shorthand DE/x/y/z, where \( x \) specifies how the base vector (of real values) is chosen (rand if it is randomly selected, or best if the best individual in the population is selected), \( y \) is the number of difference vectors used, and \( z \) denotes the crossover scheme (bin for crossover based on independent binomial experiments, and exp for exponential crossover).

A population of \( n \), \( d \)-dimensional vectors \( x_i = (x_{i1}, x_{i2}, \ldots, x_{id}) \), \( i = 1 \ldots n \) each of which encodes a solution is randomly initialized and evaluated using a fitness function \( f(\cdot) \). During the search process, each individual \( i \) is iteratively refined. The following three steps are required while execution.

1. **Mutation**: create a donor vector which encodes a solution, using randomly selected members of the population.

2. **Crossover**: create a trial vector by combining the donor vector with \( i \).

3. **Selection**: by the process of selection, determine whether the newly created trial vector replaces \( i \) in the population or not.

Under the mutation operator, for each vector \( x_i(t) \), a donor vector \( v_i(t+1) \) is obtained by the following equation:

\[
v_i(t+1) = x_i(t) + f_{m} \theta(x_i(t) - x_m(t)).
\]

where \( k, l, m \in 1 \ldots n \) are mutually distinct, randomly selected indices, and all the indices \( \neq i \). \((x_k(t))\) are referred to as the base vector and \((x_l(t) - x_m(t))\) is referred as difference vector. Selecting three indices randomly implies that all members of the current population have the same chance of being selected, and therefore influencing the creation of the difference vector. The difference between vectors \( x_l \) and \( x_m \) is multiplied by a scaling parameter \( f_m \) called mutation factor and a range for the parameter must be associated to it, that is \( f_m \in [0, 2] \). The mutation factor controls the amplification of the difference between \( x_l \) and \( x_m \) which is used to avoid stagnation of the search process. There are several alternative versions of the above process for creating a donor vector (for details see Price et al., 2005; Das and Suganthan, 2011).

A notable feature of the mutation step in DE is that it is self-scaling. The size/rate of mutation along each dimension stems solely from the location of the individuals in the current population. The mutation step self-adapts as the population converges leading to a finer-grained search. In contrast, the mutation process in GA is typically based on (or draws from) a fixed probability density function.

Following the creation of the donor vector, a trial vector \( u_i(t+1) = (u_{i1}, u_{i2}, u_{i3}, \ldots, u_{id}) \) is obtained by the following equation:

\[
u_i(t+1) = \begin{cases} v_p(t+1) \text{ if } (\text{rand} \leq c_1) \\ x_p(t) \text{ if } (\text{rand} > c_1) \text{ and } (i \neq \text{rand}(\text{ind})) \end{cases}
\]

where \( p = 1, 2, \ldots, d \), \( \text{rand} \) is a random number generated in the range \((0,1)\), \( c_1 \) is the user-specific crossover constant from the range \((0, 1)\), and \( \text{rand}(\text{ind}) \) is a randomly chosen index, selected from the range \((1, 2, \ldots, d)\). The random index is used to ensure that the trial vector differs by at least one element from \( x_i(t) \). The resulting trial (child) vector replaces its parent if it has higher fitness (a form of selection); otherwise the parent survives unchanged into the next iteration of the algorithm.

Finally, if the fitness of the trial vector exceeds that of the fitness of the parent then it replaces the parent as described in the following equation:

\[
x_i(t+1) = \begin{cases} u_i(t+1) \text{ if } (f(u_i(t+1)) > f(x_i(t))) \\ x_i(t) \text{ otherwise} \end{cases}
\]

Price, et al. (2005) provide a comprehensive comparison of the performance of DE with a range of other optimizers, including GA, and report that the results obtained by DE are consistently as good as the best obtained by other optimizers across a wide range of problem instances.

**Rationale for a DE+RBFN integration**: there are a number of reasons to suppose that an evolutionary methodology, (particularly GA) coupled with an RBF can prove fruitful in classification tasks. The selection of quality parameters for classifiers represents a high-dimensional problem, giving rise to the potential use of evolutionary methodologies. Combining these with the universal approximation qualities of an RBF produces a powerful modeling methodology.

### 3. Proposed method

The proposed approach is combining the idea of Bayesian statistics based inconsistency removal, filter based feature selection and DE based RBFNs classifier (Dash et al., 2013). It is a three-phase method. In the first phase we are making the dataset consistent in case it was inconsistent and then dividing it into training and testing sets. In phase two we are selecting a set of relevant features by using the entropy measure and again checking for inconsistency and removing it if any inconsistency arises, and in the third phase the parameters of RBFNs are trained using differential evolution. Fig. 2 depicts the overall architecture of the approach.
In the first phase we check the dataset for inconsistency. If it is inconsistent then it is made consistent by using the procedure mentioned in Section 2.3 and then divided into training and testing sets.

In the second phase, we rank the features or attributes according to the information gain ratio and then delete an appropriate number of features which have the least gain ratio (Aruna et al., 2012) and then again check for inconsistency and remove it if present. The exact number of features deleted varies from dataset to dataset. The expected information needed to classify a tuple in \( D \) is given by the following equation:

\[
\text{Info}(D) = - \sum_{i=1}^{v} p_i \log_2(p_i),
\]

where \( p_i \) is the non-zero probability that an arbitrary tuple in \( D \) belongs to class \( C_i \) and is estimated by \( |C_i|/|D| \). A log function to the base 2 is used, because the information is encoded in bits. \( \text{Info}(D) \) is the average amount of information needed to identify the class level of a tuple in \( D \). \( \text{Info}(D) \) is also known as an entropy of \( D \) and is based on only the properties of classes.

For an attribute ‘A’ entropy “\( \text{Info}_A(D) \)” (Eq. (8)) is the information still required to classify the tuples in \( D \) after partitioning tuples in \( D \) into groups only on its basis:

\[
\text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \text{Info}(D_j),
\]

where \( v \) is the number of distinct values in the attribute \( A \), \( |D| \) is the total number of tuples in \( D \) and \( |D_j| \) is the number of repetitions of the \( j \)th distinct value.

Information gain (\( \text{Gain}(A) \)) (Eq. (9)) is defined as the difference between the original information requirement and new requirement (after partitioning on \( A \)):

\[
\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)
\]

Information gain applies a kind of normalization to information gain using split information value defined analogously with \( \text{Info}(D) \) as follows:

\[
\text{SplitInfo}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2 \left( \frac{|D_j|}{|D|} \right)
\]

This value represents the potential information generated by splitting the training data set, \( D \), into \( v \) partitions, corresponding to the \( v \) outcomes of test on attribute \( A \). For each outcome, it considers the number of tuples having the outcome with respect to the total number of tuples in \( D \). The gain ratio is defined as the following:

\[
\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}_A(D)}.
\]

The more the gain ratio of an attribute the more important it is.

Notice that, in the third phase, we are focusing on the learning of the classifier. As mentioned there are two steps within the learned procedure. In step one differential evolution is employed to reveal the centers and width of the RBFNs. Although centers, widths, and weights that connect kernel nodes and output node simultaneously are evolved using DE, but here we restrict ourselves with evolving only centers and spreads. This ensures efficient representation of an individual of DE. If all these parameters are encoded, then the length of the individual would be too long and hence the search space size becomes too large, which results in a very slow convergence rate. Since the performance of the RBFNs mainly depends on center and width of the kernel, we just encode the centers and widths into an individual for stochastic search.

Suppose the maximum number of kernel nodes is set to \( K_{\text{max}} \), then the structure of the individual is represented as follows (Fig. 3):

In other words each individual has three constituent parts such as center, width and bias. The length of the individual is \( 2k_{\text{max}} + 1 \).

The fitness function which is used to guide the search process is defined in the following equation:

\[
f(x) = \frac{1}{N} \sum_{i=1}^{N} (t_i - \hat{y}(x_i))^2
\]

where \( N \) is the total number of training sample, \( t_i \) is the actual output and is the estimated output of RBFNs. Once the centers and widths are fixed, the task of determining weights in second phase of learning reduces to solving a simple linear system. In this work the pseudo-inverse method is adopted to find out a set off optimal
weights. Fig. 4 abstractly illustrates the two step tightly coupled learning procedure adopted in this work.

The algorithmic framework of the proposed method is described as follows:

Initially, a set of \( n_p \) individuals (i.e. \( n_p \) is the size of the population) pertaining to networks centers, width, and bias are created

\[
x_i^{(t)} = (x_{1i}^{(t)}, x_{2i}^{(t)}, ..., x_{di}^{(t)}), \quad i = 1, 2, ..., n_p,
\]

\( d = 2K_{\text{max}} + 1 \) where \( t \) is the iteration number.

At the start of the algorithm, this \( n_p \) set of individuals is initialized randomly and then evaluated using the fitness function \( f(\cdot) \).

In each iteration, e.g., iteration \( t \), for individual \( x_i^{(t)} \), undergoes mutation, crossover and selection as follows:

**Mutation:** for vector \( x_i^{(t)} \), a perturbed vector \( V_i^{(t+1)} \) called donor vector is generated according to the following equation:

\[
V_i^{(t+1)} = x_i^{(t)} + m_f(x_{2i}^{(t)} - x_{3i}^{(t)}),
\]

where \( m_f \) is the mutation factor drawn from \([0,2]\), the indices \( r_1, r_2 \) and \( r_3 \) are selected randomly from \([1,2,3,..., n_p]\), \( r_1 \neq r_2 \neq r_3 \neq i \).

**Crossover:** the trial vector is generated as follows:

\[
\begin{align*}
u_j^{(t+1)} &= u_j^{(t+1)}, i = 1, 2, \ldots, \hat{d} \\
u_j^{(t+1)} &= \begin{cases} 
\psi_j^{(t)} & \text{if} \quad \text{rand} \leq c_i \\
\chi_j^{(t)} & \text{if} \quad \text{rand} > c_i
\end{cases}, \quad j = 1, 2, \ldots, d,
\end{align*}
\]

where \( \psi_j^{(t)} \) is a random number generated in the range \([0,1]\), \( c_i \) is the user-specified crossover constant from the range \([0,1]\) and \( \text{rand}(1,2,\ldots,d) \) is a randomly chosen index from the range \([1,2,\ldots,d]\). The random index is used to ensure that the trial solution vector differs by at least one element from \( x_i^{(t)} \). The resulting trial (child) solution replaces its parent if it has a higher accuracy (a form of selection), otherwise the parent survives unchanged into the next iteration of the algorithm. Fig. 5 illustrates this operation in the context of revealing center, width, and bias.

Finally, we use selection operation and obtain target vector \( x_i^{(t+1)} \) as follows:

\[
x_i^{(t+1)} = \begin{cases} 
u_i^{(t+1)} & \text{if} \quad f(x_i^{(t+1)}) \leq f(x_i^{(t)}) \quad \text{for all} \quad d. \\
n_i^{(t)} & \text{otherwise.}
\end{cases}
\]

Given that the centers, widths, and bias are computed from training vectors, the weight of the learning network is computed by the following pseudo-inverse matrix manipulation method:

\[
Y = W\Phi \\
\Rightarrow W = (\Phi^T\Phi)^{-1}\Phi^TY.
\]

The pseudo-code for two steps learning is as follows:

1. **INITIALIZATION:** randomly create a pool of individuals (an individual represents centers, spreads, and bias).
2. **WEIGHT DETERMINATION:** by pseudo-inverse method (i.e., by using Eq. (16)) determine weight vectors.
3. **FITNESS COMPUTATION:** by using Eq. (12) compute the fitness of every individual.
4. **WHILE** (accuracy not acceptable) DO
5. **DONER VECTOR:** create a donor vector which encodes a solution by randomly selected members of the population.
6. **TRIAL VECTOR:** create a trial vector by combining with the donor vector.
7. **SELECTION:** by this process determine whether the newly created trial vector survives or not.
8. **WEIGHT DETERMINATION:** by pseudo-inverse method (i.e., by using Eq. (16)) determine weight vectors.
Table 6
Results obtained from DE + RBFN (95% confidence level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average accuracy of 10 independent runs</th>
<th>Average accuracy of 20 independent runs</th>
<th>Average accuracy of 30 independent runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAMMOGRAPHIC MASSES</td>
<td>81.6744 ± 0.034</td>
<td>81.5233 ± 0.034</td>
<td>81.7442 ± 0.034</td>
</tr>
<tr>
<td>HABERMEN</td>
<td>88.7097 ± 0.050</td>
<td>88.7097 ± 0.050</td>
<td>88.7097 ± 0.050</td>
</tr>
<tr>
<td>TRANSFUSION</td>
<td>85.7664 ± 0.035</td>
<td>85.7664 ± 0.035</td>
<td>85.7664 ± 0.035</td>
</tr>
</tbody>
</table>

Table 7
Results obtained from DE + RBFN (98% confidence level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average accuracy of 10 independent runs</th>
<th>Average accuracy of 20 independent runs</th>
<th>Average accuracy of 30 independent runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAMMOGRAPHIC MASSES</td>
<td>81.6744 ± 0.041</td>
<td>81.5233 ± 0.041</td>
<td>81.7442 ± 0.041</td>
</tr>
<tr>
<td>HABERMEN</td>
<td>88.7097 ± 0.059</td>
<td>88.7097 ± 0.059</td>
<td>88.7097 ± 0.059</td>
</tr>
<tr>
<td>TRANSFUSION</td>
<td>85.7664 ± 0.035</td>
<td>85.7664 ± 0.035</td>
<td>85.7664 ± 0.035</td>
</tr>
</tbody>
</table>

Table 8
Results obtained from DE + RBFN with feature selection (95% confidence level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Feature removed from the dataset</th>
<th>Average accuracy of 10 independent runs</th>
<th>Average accuracy of 20 independent runs</th>
<th>Average accuracy of 30 independent runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAMMOGRAPHIC MASSES</td>
<td>5</td>
<td>82.3023 ± 0.034</td>
<td>82.2907 ± 0.034</td>
<td>82.1318 ± 0.034</td>
</tr>
<tr>
<td>HABERMEN</td>
<td>2</td>
<td>88.7097 ± 0.050</td>
<td>88.7097 ± 0.050</td>
<td>88.7097 ± 0.050</td>
</tr>
<tr>
<td>TRANSFUSION</td>
<td>2</td>
<td>85.7664 ± 0.035</td>
<td>85.7664 ± 0.035</td>
<td>85.7664 ± 0.035</td>
</tr>
</tbody>
</table>

Table 9
Results obtained from DE + RBFN with feature selection (98% confidence level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Feature removed from the dataset</th>
<th>Average accuracy of 10 independent runs</th>
<th>Average accuracy of 20 independent runs</th>
<th>Average accuracy of 30 independent runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAMMOGRAPHIC MASSES</td>
<td>5</td>
<td>82.3023 ± 0.040</td>
<td>82.2907 ± 0.040</td>
<td>82.1318 ± 0.040</td>
</tr>
<tr>
<td>HABERMEN</td>
<td>2</td>
<td>88.7097 ± 0.059</td>
<td>88.7097 ± 0.059</td>
<td>88.7097 ± 0.059</td>
</tr>
<tr>
<td>TRANSFUSION</td>
<td>2</td>
<td>85.7664 ± 0.042</td>
<td>85.7664 ± 0.042</td>
<td>85.7664 ± 0.042</td>
</tr>
</tbody>
</table>

Table 10
Results obtained from DE + RBFN after data consistency (95% confidence level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average accuracy of 10 independent runs</th>
<th>Average accuracy of 20 independent runs</th>
<th>Average accuracy of 30 independent runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAMMOGRAPHIC MASSES</td>
<td>85.12 ± 0.031</td>
<td>85.06 ± 0.031</td>
<td>85.0933 ± 0.031</td>
</tr>
<tr>
<td>HABERMEN</td>
<td>94.5578 ± 0.035</td>
<td>94.5578 ± 0.035</td>
<td>94.5578 ± 0.035</td>
</tr>
<tr>
<td>TRANSFUSION</td>
<td>92.9825 ± 0.025</td>
<td>92.9825 ± 0.025</td>
<td>94.5578 ± 0.025</td>
</tr>
</tbody>
</table>

Table 11
Results obtained from DE + RBFN after data consistency (98% confidence level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average accuracy of 10 independent runs</th>
<th>Average accuracy of 20 independent runs</th>
<th>Average accuracy of 30 independent runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAMMOGRAPHIC MASSES</td>
<td>85.12 ± 0.037</td>
<td>85.06 ± 0.037</td>
<td>85.0933 ± 0.037</td>
</tr>
<tr>
<td>HABERMEN</td>
<td>94.5578 ± 0.042</td>
<td>94.5578 ± 0.042</td>
<td>94.5578 ± 0.042</td>
</tr>
<tr>
<td>TRANSFUSION</td>
<td>92.9825 ± 0.030</td>
<td>92.9825 ± 0.030</td>
<td>94.5578 ± 0.030</td>
</tr>
</tbody>
</table>

Table 12
Results obtained from DE + RBFN with data consistency and feature selection (95% confidence level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Feature removed from the dataset</th>
<th>Average accuracy of 10 independent runs</th>
<th>Average accuracy of 20 independent runs</th>
<th>Average accuracy of 30 independent runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAMMOGRAPHIC MASSES</td>
<td>5</td>
<td>86.2174 ± 0.030</td>
<td>86.1087 ± 0.030</td>
<td>86.4638 ± 0.030</td>
</tr>
<tr>
<td>HABERMEN</td>
<td>2</td>
<td>96.8421 ± 0.027</td>
<td>96.8421 ± 0.027</td>
<td>96.8421 ± 0.027</td>
</tr>
<tr>
<td>TRANSFUSION</td>
<td>1</td>
<td>94.3662 ± 0.023</td>
<td>94.3662 ± 0.023</td>
<td>94.3662 ± 0.023</td>
</tr>
</tbody>
</table>
Table 13
Results obtained from DE+RBFN with data consistency and feature selection (98% confidence level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Feature removed from the dataset</th>
<th>Average accuracy of 10 independent runs</th>
<th>Average accuracy of 20 independent runs</th>
<th>Average accuracy of 30 independent runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAMMOGRAPHIC MASSES</td>
<td>5</td>
<td>86.2174 ± 0.036</td>
<td>86.1087 ± 0.036</td>
<td>86.4638 ± 0.036</td>
</tr>
<tr>
<td>HABERMEN</td>
<td>2</td>
<td>96.8421 ± 0.032</td>
<td>96.8421 ± 0.032</td>
<td>96.8421 ± 0.032</td>
</tr>
<tr>
<td>TRANSFUSION</td>
<td>1</td>
<td>94.3662 ± 0.027</td>
<td>94.3662 ± 0.027</td>
<td>94.3662 ± 0.027</td>
</tr>
</tbody>
</table>

Table 14
Average results of 10, 20, and 30 independent runs obtained from DE+RBFN, DE+RBFN with feature selection, DE+RBFN with data consistency and DE+RBFN with data consistency and feature selection (95% confidence level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average accuracy of DE+RBFN</th>
<th>Average accuracy of DE+RBFN with feature selection</th>
<th>Average accuracy of DE+RBFN with data consistency</th>
<th>Average accuracy of DE+RBFN with data consistency and feature selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAMMOGRAPHIC MASSES</td>
<td>81.6473 ± 0.034</td>
<td>82.2416 ± 0.034</td>
<td>85.0911 ± 0.031</td>
<td>86.2633 ± 0.030</td>
</tr>
<tr>
<td>HABERMEN</td>
<td>88.7097 ± 0.050</td>
<td>88.7097 ± 0.050</td>
<td>94.5578 ± 0.035</td>
<td>96.8421 ± 0.027</td>
</tr>
<tr>
<td>TRANSFUSION</td>
<td>85.7664 ± 0.035</td>
<td>85.7664 ± 0.035</td>
<td>92.9825 ± 0.025</td>
<td>94.3662 ± 0.023</td>
</tr>
</tbody>
</table>

Table 15
Average results of 10, 20, and 30 independent runs obtained from DE+RBFN, DE+RBFN with feature selection, DE+RBFN with data consistency and DE+RBFN with data consistency and feature selection (98% confidence level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average accuracy of DE+RBFN</th>
<th>Average accuracy of DE+RBFN with feature selection</th>
<th>Average accuracy of DE+RBFN with data consistency</th>
<th>Average accuracy of DE+RBFN with data consistency and feature selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAMMOGRAPHIC MASSES</td>
<td>81.6473 ± 0.041</td>
<td>82.2416 ± 0.040</td>
<td>85.0911 ± 0.037</td>
<td>86.2633 ± 0.036</td>
</tr>
<tr>
<td>HABERMEN</td>
<td>88.7097 ± 0.059</td>
<td>88.7097 ± 0.059</td>
<td>94.5578 ± 0.042</td>
<td>96.8421 ± 0.032</td>
</tr>
<tr>
<td>TRANSFUSION</td>
<td>85.7664 ± 0.041</td>
<td>85.7664 ± 0.041</td>
<td>92.9825 ± 0.030</td>
<td>94.3662 ± 0.027</td>
</tr>
</tbody>
</table>

Fig. 6. (a–d) Iteration number vs. error plot for MAMMOGRAPHIC MASSES dataset.
9. FITNESS COMPUTATION: by using Eq. (12) compute the fitness of every individual.

10. ENDDO

4. Experimental study

In subsection 4.1, we briefly describe about datasets and parameters required to set in the experimental study. Subsection 4.2 offers results and analysis.

4.1. Description of datasets and parameters

The datasets used in this work were obtained from the UCI machine learning repository (Frank and Asuncion, 2010). Three medical related datasets have been chosen to validate the proposed method. The details about the three datasets are given in Table 4. The last two columns of Table 4 indicate whether a dataset is balanced or not. In particular, the datasets like HABERMEN and BLOOD TRANSFUSION are imbalanced, whereas HAMMOGRAPHIC MASSES are balanced. The imbalanced problem occurs when the number of instances of certain classes is much lower than the instances of other classes. Standard classification algorithms usually have a bias towards the majority classes (e.g., class 1 of HABERMAN contains 225, whereas class 2 contains 81 samples) during the learning process in favor of the accuracy metric which does not take into account the class distribution of the data. Consequently the instances belonging to the minority class are misclassified more often than those belonging to the majority class.

The approach proposed so far for addressing the imbalanced problem has been classified into two groups: (i) the algorithm based approaches, which create new algorithms or modify existing ones and (ii) data-based approaches which pre-process the data in order to diminish the effect caused by their class imbalance. The proposed approach belongs to the second group.

In our experiment, every dataset is divided into two mutually exclusive parts: training sets and test sets. The parameters value used for validating our proposed method is listed in Table 5. In connection to RBFNs the number of kernels is kept 5 for all datasets along with all runs. After thorough and rigorous experimentation with respect to different datasets and varying number of independent runs, we fixed the parameters $f_m$, $c$, size of the population, and maximum iterations.

4.2. Results and analysis

The summary of the experimental results are presented in Tables 6–15. Through the empirical validation, we have reported the results of the classifier under five categories:

(i) Average classification accuracy of 10, 20, and 30 independent runs with respect to 95% and 98% confidence level without selecting any features (cf. Tables 6 and 7);
(ii) Average classification accuracy of 10, 20, and 30 independent runs with respect to 95% and 98% confidence level after removing one-third of the features based on filter approach (cf. Tables 8 and 9).

(iii) Average classification accuracy of 10, 20, and 30 independent runs with respect to 95% and 98% confidence level after adapting data consistency (cf. Tables 10 and 11).

(iv) Average classification accuracy of 10, 20, and 30 independent runs with respect to 95% and 98% confidence level after adapting data consistency and feature selection (cf. Tables 12 and 13).

(v) Finally, average accuracy result of the above methods of 10, 20 and 30 independent runs (cf. Tables 14 and 15).

From Tables 6 and 7 it has been noticed that in MAMMOGRAPHIC MASSES the average accuracy obtained in the series of different independent runs are varying marginally, whereas in the case of other two datasets it is constant.

From Tables 8 and 9, it is realized that feature selection is important in Mammographic Masses, whereas it is not mandatory to apply feature selection in the case of other two datasets. Particularly, in the case of HABERMEN and BLOOD TRANSFUSION no such improvement has been noticed even though one-third feature is removed. The column 2 of Tables 8 and 9 is indicating the feature removed from the dataset (e.g., 5 mean fifth feature is removed). Based on the empirical study the designated feature is selected for removing from the dataset.

From Tables 10 and 11 it is observed that, in all the datasets with 95% and 98% confidence level, DE + RBFNs with data consistency (threshold parameter \( \alpha = 0.5 \)) is outperforming than DE + RBFNs without data consistency over all independent runs.

Similarly, Tables 12 and 13 indicate that it is important to consider both feature selection and removal of inconsistency from the dataset while designing a classifier. The explanation of these results with 95% and 98% confidence level flags a new type of methods for tackling the problem of imbalanced class.

Tables 14 and 15 give overall comparative results. Column 2 describes the average accuracy of DE + RBFNs, similarly, columns 3, 4, and 5 illustrate the average accuracy of DE + RBFN with feature selection, with data consistency, and with both feature selection and data consistency.

The top left hand corner (a) of Fig. 6 shows error vs. iteration for DE + RBFN with inconsistency removal; top right hand corner (b) of Fig. 6 shows error vs. iteration for DE + RBFN with inconsistency removal and feature selection; bottom left hand corner (c) of Fig. 6 shows error vs. iteration for DE + RBFN with feature selection; and bottom right hand corner (d) of Fig. 6 shows error vs. iteration for DE + RBFN.

The top left hand corner (a) of Fig. 7 shows error vs. iteration for DE + RBFN with inconsistency removal; top right hand corner (b) of Fig. 7 shows error vs. iteration for DE + RBFN; bottom left hand corner (c) of Fig. 7 shows error vs. iteration for DE + RBFN with inconsistency removal and feature selection; and bottom right hand corner (d) of Fig. 7 shows error vs. iteration for DE + RBFN.
right hand corner (d) of Fig. 7 shows error vs. iteration for DE + RBFN with feature selection.

The top left hand corner (a) of Fig. 8 shows error vs. iteration for DE + RBFN; top right hand corner (b) of Fig. 8 shows error vs. iteration for DE + RBFN with inconsistency removal and feature selection; bottom left hand corner (c) of Fig. 8 shows error vs. iteration for DE + RBFN with inconsistency removal; and bottom right hand corner (d) of Fig. 8 shows error vs. iteration for DE + RBFN with feature selection.

5. Conclusions

In this paper, a synergy of Bayesian statistics based inconsistency removal, filter based feature selection, and differential evolution trained RBFNs is functioning towards the removal of inconsistency, the reduction of irrelevant features, and the maximization of predictive accuracy of the classifier. The method of encoding an RBF network into an individual is given, where only the centers and spreads of the hidden units are encoded along with the bias of the network. The connection weights between hidden layer and output layer are obtained by pseudo-inverse method. The performance under synergistic approach is promising and very consistent. With 95% and 98% confidence level the comparative performance shows that the average accuracy of DE + RBFNs along with feature selection and consistency is superior to its counterpart. Interestingly, it is noticed that in the case of imbalanced dataset (e.g., HABERMAN) the accuracy of the proposed approach almost enhanced 5% than DE + RBFNs without feature selection and consistency removal. Hence, we can conclude that that removal of irrelevant features and inconsistency samples may lead to a solution to cope with the class imbalanced problem.

Of course, what we have observed is limited by the datasets, we employed DE + RBFNs, Bayesian approach of feature inconsistency removal, filter based feature selection, and the simulation tool MATLAB 6.5. More imbalanced datasets should be examined with the proposed approach to further justify (or refute) our findings. Furthermore, in the future scope of research, lots of avenues are here, e.g., (i) the synergy of fuzzy entropy based feature selection and DE trained RBFNs with inconsistency removal and (ii) a very rigorous comparative analysis with other classifiers who are simultaneously reducing the features and maximizing the classification accuracy.

Acknowledgements

S.-B. Cho is gratefully acknowledge the support of the Original Technology Research Program for Brain Science through the National Research Foundation (NRF) of Korea (NRF: 2010-0018948) funded by the Ministry of Education, Science, and Technology. G.-N. Wang acknowledges the support of Defense Acquisition Program Administration and Agency for Defense Development under the contract UD1100006MD and the Industrial Strategic Technology Development Program, 10047046, funded by the Ministry of Science, ICT & Future Planning (MSIP), Korea.

References


