iThenticate sample report

WORD COUNT 6639
TIME SUBMITTED 24-FEB-2021 12:04AM
PAPER ID 69302980

Soft Computing
https://doi.org/10.1007/s00500-020-05547-7

FOCUS

A transparent rule-based expert system using neural network

Abstract

Classification is one of the foremost machine learning tasks in this modern era. Neural Network (NN) is one of the powerful classification techniques. NN can achieve high classification accuracy on highly imbalanced and complex datasets, but lacks in explanation of its reasoning process which limits its applicability in various domains which require transparent decision along with good accuracy. There are some techniques which extract rules from NN and make it transparent; however, attribute pruning, rule pruning and class overlap algorithms are not sufficiently effective. Therefore, this paper proposes a rule extraction algorithm, called Transparent Rule Extraction using Neural Network (TRENN) to convert NN into white box with greater emphasis on attribute pruning and rule pruning. The proposed TRENN is a pedagogical approach and an extension of one of the existing algorithms named Rule Extraction from Neural Network using Classified and Misclassified data (RxNCM). The proposed TRENN extends the RxNCM with sequential floating backward search for feature and rule selection to improve the comprehensibility of the generated rules. Besides, the proposed TRENN uses probabilistic approach for the treatment of class overlapping problem in the rule updating phase instead of reclassification used in RxNCM where the overlap may persist. Experiments are conducted with eight real datasets collected from the UCI repository. Performance of the TRENN is measured with Precision, Recall, FP-Rate, F-measure, and local and global comprehensibility. It is observed from the experimental results that TRENN performs better than Re-RX, RxNCM and RxREN.

Keywords Machine learning · Neural networks · Rule extraction · Pedagogical approach · Classification

1 Introduction

With the advent of powerful communication and technology, an enormous amount of data is being collected day-by-day. Some useful and valuable information are hidden in this data and it is difficult to extract them correctly. Fortunately, data mining techniques have come up with its super computing capabilities which can extract hidden patterns and information from the data to make them useful in different decision-making tasks (Han and Kambler 2001). Some of the data mining tasks are regression, classification, clustering, association analysis and so on (Sing and Midha 2015; Mann and Kaur 2013; Shridhar and Parmar 2017). Among them classification is the most common and popular data mining task (Sharma and Shani 2011). There are many classification techniques such as Bayesian Classification (BC), Decision Trees (DT), Neural Networks (NN), Support Vector Machine (SVM) (Kaviani and Dhotre 2017; Cohen et al. 2007; Mashayekhi and Gras 2015; Kaikhah and Doddmeti 2006). NN is the most popular among them due to its incomparable capabilities of classifying data with mixed mode attributes, achieving higher accuracy and maintaining a low computational complexity (Dam et al. 2008; Caruana and Niculescu-Mizil 2007). However, the drawback of NN is its black box nature in decision making as it does not explain the decision-making process in human understandable form

Communicated by Vladik Kreinovich.

✉ Abhinaba Dattachaudhuri
ad.chaudhuri1995@gmail.com

Saroj Kr. Biswas
saroj@cse.nits.ac.in

Manomita Chakraborty
mou.look@gmail.com

Sunita Sarkar
sunitasarkar@rediffmail.com

[1] Computer Science and Engineering Department, National Institute of Technology Silchar, Silchar, Assam, India

[2] Assam University, Silchar, Assam, India

Published online: 27 January 2021

(Mantas et al. 2006). Since black box nature is not comprehensible in reasoning and decision making, NN struggles in fields where explanation of a decision is needed. Many times, fields like medical diagnosis, financial decision making, infrastructure management and others require clear explanation of a decision-making process. For example, in medical diagnosis a clear explanation of the causes of a disease is required to spread awareness among the common people and to take some precautionary measures to prevent the disease in advance. Consequently, to convert black box nature of NN into white box many algorithms have been proposed to extract comprehensible rules from NN (Setiono and Liu 1995).

Rule extraction techniques from NN can be categorized as decompositional, pedagogical and eclectic based on the procedure applied to extract the rules (Botari et al. 2019). Decompositional techniques involve analyzing the weights between the units and activation function. Pedagogical techniques extract rules by examining the relationship between the inputs and outputs. Eclectic techniques incorporate both decompositional and pedagogical techniques together (Bologna and Hayashi 2018). The pedagogical techniques among them are widely used because of its less computational demand, simplicity in implementation and higher accuracy than others (Kaviani and Dhotre 2017). Some of the recent and successful pedagogical techniques are Rule Extraction by Reverse Engineering (RxREN) (Augasta and Kathirvalavakumar 2012), Rule Extraction from Neural Network using Classified and Misclassified data (RxNCM) (Biswas et al. 2017), BRAINNE (Sestito and Dillon 1992) and Extended Treepan (X-TREPAN) (Craven and Shavlik 1996; Karim and Zhou 2015). Among them, RxNCM is the most recent one which extracts rules by reverse engineering the NN to prune insignificant input neurons and uses correctly classified and misclassified patterns to generate the rules. However, RxNCM algorithm uses a sequential feature selection method to prune input neurons. Hence RxNCM suffers from nesting effect that means once a feature is pruned, it cannot be considered for further processing and thereby RxNCM loses some potential combinations to consider in future. Therefore, RxNCM cannot find the optimal subset of features and thus degrades the performance. RxNCM algorithm also adopts the sequential rule selection to prune the rules in rule pruning phase and thus keeps the same problem. Therefore, RxNCM cannot find the optimal subset of rule conditions for decision making. Further, RxNCM updates the final rule-set by reclassification which may sometimes improve the training accuracy but the new data range may not be free from class overlapping problem. Keeping in view of all the drawbacks, this paper proposes a pedagogical rule extraction algorithm named as Transparent Rule Extraction using Neural Network (TRENN) which uses backward floating method to prune the input neurons and the rules, and uses a probabilistic approach to overcome the class overlapping problem. The backward floating method takes all the features for further processing including the features that have been deemed insignificant earlier, and thus prevents nesting effect. The probabilistic approach in rule updating removes overlap by shifting the upper and lower range of data according to the class probability of each attribute. This backward floating method makes the features/rules optimal as most of the significant combinations of features/rules are taken into consideration. Besides, the probabilistic rule update eliminates class overlapping problem and hence the proposed TRENN improves the performance.

2 Literature survey

NN is a powerful tool that has the ability to derive meaningful information from complicated or imprecise data and can be used to extract patterns and detect complex trends in the data (Lu et al. 1995; Bologna and Hayashi 2018). But it is inherently black box in nature. However there are a number of approaches to convert the NN into white box by extracting transparent rules from NN (Setiono and Liu 1995). There have been many proposed rule extraction algorithms that reveal the information contained in the NN. Setiono and Liu (1996) proposed a decompositional algorithm called the NeuroRule (NR) algorithm that extracts oblique classification rules from NN with one hidden layer. The rule generation component of NR is called Rule Generation (RG) which generates rules that cover as many examples of a distinct class as possible with minimum number of attributes. Gupta et al. (1999) proposed a Generalized Analytic Rule Extraction method (GLARE) from Feed Forward Neural Network (FFNN) which measures the strength of weights to extract rules. This utilizes the standard network structure and training methods in rule extraction, and also makes a direct mapping between input and output nodes to enhance the comprehensibility. Odajimaa et al. (2008) proposed a decompositional method called Greedy Rule Generation (GRG) for discrete attributes. GRG generates much fewer rules than NeuroRule. Bondarenko et al. (2017) proposed a decompositional approach named Neural Network Knowledge eXtraction (NNKX) that extracts rules from a multilayer FFNN in the guise of binary classification DT. NNKX uses clustering on activation value of neurons to form the decision for each tree branch and hence has a high computational cost for both clustering and rule generation steps. Craven and Shavlik (1996) proposed a method called TREPAN which extracts rules from NN in the form of DT. During the learning phase of the Neural Network, this algorithm queries the network to determine class patterns. These are

used create to a DT which represents the knowledge represented the network. Setiono et al. (2008) devised a decompositional approach called Recursive Rule eXtraction (Re-RX) that generates classification rule from NN using discrete and continuous attributes. Biswas et al. (2017) proposed the RxNCM algorithm that is an improvement of the RxREN algorithm proposed by Augasta and Kathirvalavakumar (2012). The RxREN uses misclassified patterns after removing insignificant neurons to generate the rules. However, RxNCM algorithm considers both the misclassified as well as the properly classified patterns to generate the rules. Hruschka and Ebeckenb (2006) proposed the Rule Extraction from Constructive Genetic Algorithm (Rex-CGA) method that works with multiple hidden layers. The Rex-CGA uses CGA to find clusters of activation values in the hidden layers to generate rules. The Fast Extraction of Rules from Neural Networks (FERNN) proposed by Setiono and Kheng (2000) identifies the significant hidden neurons and the significant input-hidden connections of a fully connected trained single hidden layer network to generate rules. To identify the significant hidden neurons the algorithm uses DT. Iqbal (2012) proposed Hierarchical and Eclectic Rule Extraction via Tree Induction and Combination (HERETIC) that uses DT to generate rules from individual nodes of a network and combines the rules generated from all the nodes to construct final rules. Jivani et al. (2014) compared decompositional, pedagogical and eclectic rule extraction approaches and reported that pedagogical approach is computationally faster than both decompositional and eclectic approaches, while maintaining fairly high accuracy.

3 Proposed TRENN algorithm

The proposed TRENN algorithm consists of six phases: *Optimal Network Architecture phase*, *Network Pruning phase*, *Data Range Calculation phase*, *Rule Construction phase*, *Rule Pruning phase* and *Rule Updating phase*. Figure 1 shows the flowchart of the TRENN algorithm. In the first phase, the algorithm determines the optimal network architecture. In the network pruning phase, it removes the insignificant input neurons from the trained neural network. In the data range calculation phase, the algorithm calculates the data ranges of the significant inputs. In the rule construction phase, it constructs classification rules for each class using the data ranges obtained in the previous phase. In the rule pruning phase, it prunes the constructed rules to remove insignificant ones. Finally, in the rule update phase, the algorithm updates the attribute data ranges of the pruned rules.

Notations used in the paper:

T: Set of classified examples by a neural network on a given training set.
l: Number of input neurons.
h: Number of hidden layer neurons.
n: Number of output neurons.
Acc_o: Accuracy of a trained network on validation dataset.
A: Set of input neurons in network.
B: Set of insignificant input neurons.
Acc_p: Accuracy of a pruned network on validation dataset.
m: Number of input neurons in the pruned network.
l_i: i^{th} input neuron.
C_k: k^{th} target class of a dataset.
err_i: Number of incorrectly classified examples by the trained network without l_i where $i \in [1, M]$.
E_i: Incorrectly classified examples by the trained network without l_i.
P_i: Properly classified examples with only l_i in the network.
UEP_i: Union of P_i and E_i.
ep_i: Total number of examples in UEP_i for l_i.
cep_{ik}: Number of examples in UEP_i for l_i in class C_k.
DRM Data range matrix of order $m \times n$.
DRM_{ik}: Data range of attribute l_i in class C_k.
L_{ik}: Lower range of for input i in class C_k of **DRM.**
U_{ik}: Upper range of input i in class C_k of **DRM.**
min_{ik}: Lower range after rule updation.
max_{ik}: Upper range after rule updation.
R_k: Rule set for class k.
cn_j: j^{th} condition in R_k where $j \in [1, m]$.
D: Set of insignificant rule conditions neurons.
Acc_r: Accuracy of initial rule set R_k.
Acc_j: Accuracy of rule set on removal of cn_j.

All the phases of proposed TRENN algorithm are explained below:

3.1 Optimal network architecture

The algorithm uses a back propagation Neural Network with one hidden layer having h neurons for rule extraction. It selects the number h based on the mean square error of the network. The algorithm varies the network architecture from $l + 1$ to $2*l$ hidden neurons where l is the number of input neurons and choses the architecture with least mean square error. Figure 2 depicts the whole process.

3.2 Network pruning

The TRENN removes the insignificant input neurons from the network using backward floating technique as shown in Fig. 3. This backward floating method makes the feature/

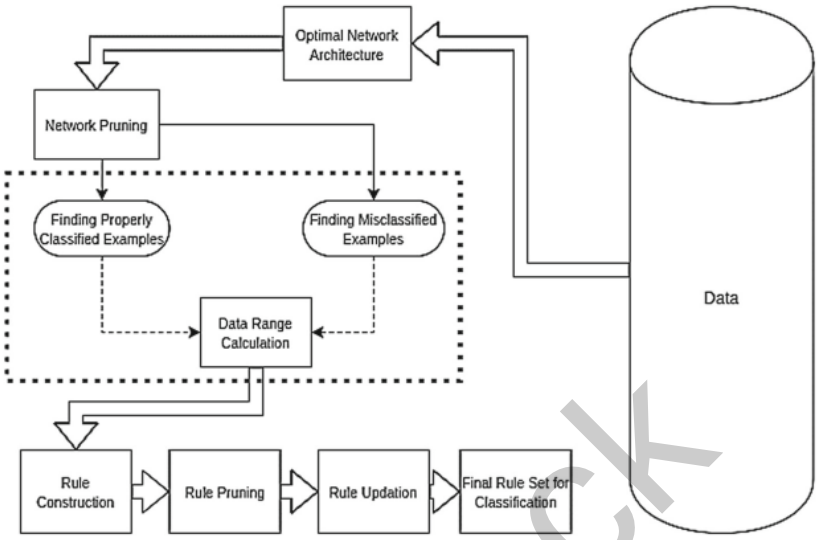

Fig. 1 Flowchart for TRENN algorithm

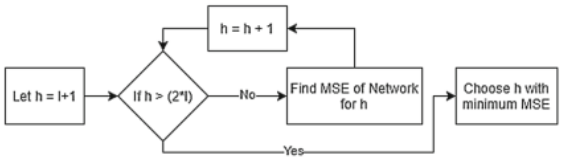

Fig. 2 Flowchart for optimal network architecture selection

rule set more reliable as all combinations of features/rules are taken into consideration and hence improves the performance of the final set of rules extracted.

For all the input neurons l_i, TRENN finds the number of misclassified examples err_i after removing the *ith* input neuron. The input neuron with lowest err_i value for the network is removed and a temporary pruned network is created. The accuracy of this temporary pruned network

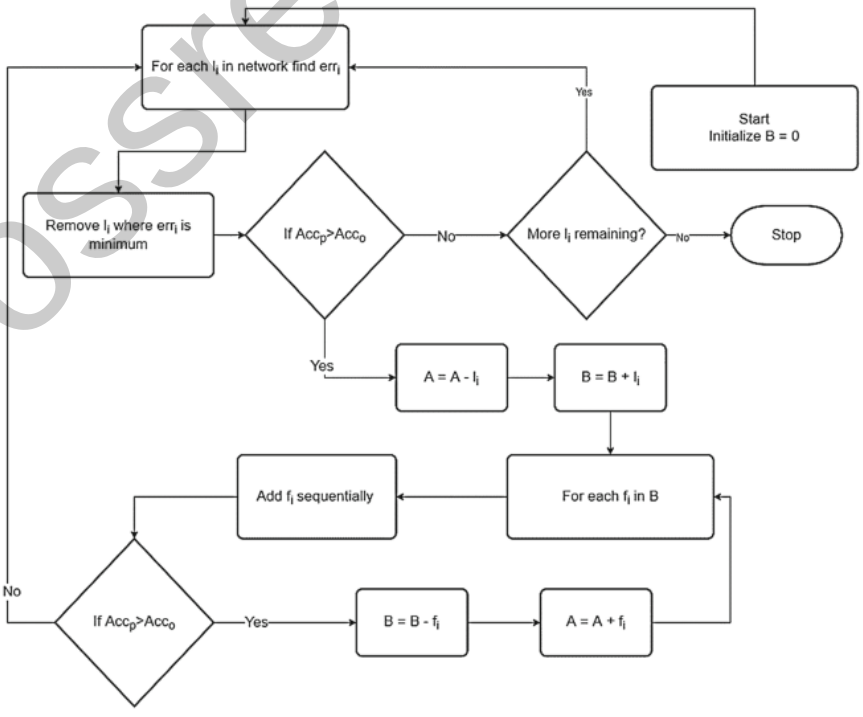

Fig. 3 Flowchart for network pruning

Acc_p is calculated. This Acc_p is compared with accuracy Acc_o. If $Acc_p > Acc_o$ then the algorithm considers this temporary pruned network as the pruned network and sets $Acc_o = Acc_p$. The algorithm discards the removed input neuron from the network set A and is keeps in set B for future consideration.

TRENN sequentially adds the removed input neurons (in set B) one by one, to the pruned network (set A). If the new accuracy (say Acc_p) is strictly greater ($Acc_p > Acc_o$) than the current accuracy, the algorithm adds that input neuron back to the network and updates the accuracy ($Acc_o = Acc_p$). The algorithm executes he similar procedure for all the remaining input neurons.

The algorithm for the network pruning step is given below:

//Network Pruning algorithm//

Step 1. *Initialize* $B = 0$
For each l_i *of trained ANN:*
Step 1.1. *Remove* l_i *and find* err_i

Step 2. *Remove* l_i, *where* err_i *is minimum*
Step 2.1. *Set* Acc_p *as accuracy of the pruned network*

Step 3. *If* $Acc_p \geq Acc_o$ *then:*
$Acc_o = Acc_p$
$A = A - l_i$
Goto ***Step 4.***
Else
Goto ***Step 7.***

Step 4. *Initialize the set B as set of pruned input neurons*
Step 4.1. $B = B + l_i$

Step 5. *For each* f_i *in B*
Step 5.1. *Add* f_i *to the temporary pruned network*
Step 5.2. *Set* Acc_p

Step 6. *If* $Acc_p > Acc_o$ *then:*
$Acc_o = Acc_p$
$B = B - f_i$
$A = A + f_i$
Goto ***Step 5.***
Else
Goto ***Step 2.***

Step 7. *Stop.*

The final output of this phase is a pruned neural network.

3.3 Data range calculation

The TRENN algorithm learns the functionality and importance of each significant input neuron l_i by analyzing the misclassified patterns E_i in absence of each l_i and the properly classified patterns P_i in presence of each l_i. To find the mandatory data range of l_i for each class C_k, TRENN groups the examples in set UEP_i with respect to each target class C_k and finds the number of examples cep_{ik} in each class as shown in Fig. 4. The matrix formed is named as data length matrix. For the input neuron l_i, the number of examples in set UEP_i is ep_i. Here k lies in the range of $[1, n]$.

All the attributes may not be necessary for classifying patterns in every class i.e., a particular attribute may not be significant to classify patterns in all the n classes. Therefore, the algorithm selects the data ranges of those attributes which satisfy the following condition (1), to create a Data Range Matrix (DRM).

$$cep_{ik} > \alpha * ep_i \text{ for class } k, \text{ where } \alpha \in [0.1, 0.5] \quad (1)$$

Figure 5 shows a ***DRM***. The algorithm calculates each element of a ***DRM*** using Eq. (2).

$$\text{DRM}_{ik} = \begin{cases} [L_{ik}, U_{ik}], \text{cep}_{ik} > \alpha * ep_i \\ 0, \text{ otherwise} \end{cases} \quad (2)$$

Figure 6 shows the flowchart for the data range calculation phase.

The algorithm of the data range calculation phase is given below:

//Data Range Calculation//

Step 1. *For each* l_i *in the pruned network:*
Step 1.1. *Find* UEP_i *where* $UEP_i = P_i \cup E_i$

Step 2. *Group the examples belonging to* UEP_i *for each* C_k
Step 2.1. *Find* cep_{ik} *in each group where* $1 \leq k \leq n$

Step 3. *If* $cep_{ik} > \alpha * ep_i$, *where* $\alpha \in [0.1, 0.5]$ *then:*
$DRM_{ik} = [L_{ik}, U_{ik}]$
Else
$DRM_{ik} = 0$

	C_1	C_2	C_3	...	C_n
l_1	cep_{11}	cep_{12}	cep_{13}	...	cep_{1n}
l_2	cep_{21}	cep_{22}	cep_{23}	...	cep_{2n}
l_3	cep_{31}	cep_{32}	cep_{33}	...	cep_{3n}
...	...	...	...	...	...
l_m	cep_{m1}	cep_{m2}	cep_{m3}	...	cep_{mn}

Fig. 4 Data length matrix

3.4 Rule construction

The algorithm constructs rules for each class using the data ranges obtained in the previous phase. It considers a rule for a class **k** if the data range for that class is nonzero. So, for **n** classes and **m** significant attributes the rules have the following general outline:

if$((data(l_1) \geq L_{11} \wedge data(l_1) \leq U_{11}) \wedge (data(l_2) \geq L_{21} \wedge data(l_2) \leq U_{21}) \wedge \ldots \wedge (data(l_m) \geq L_{m1} \wedge data(l_m) \leq U_{m1}))$ *then*
class $= C_1$
else
if$((data(l_1) \geq L_{12} \wedge data(l_1) \leq U_{12}) \wedge (data(l_2) \geq L_{22} \wedge data(l_2) \leq U_{22}) \wedge \ldots \wedge (data(l_m) \geq L_{m2} \wedge data(l_m) \leq U_{m2}))$ *then*
class $= C_2$
else
...
if$((data(l_1) \geq L_{1(n-1)} \wedge data(l_1) \leq U_{1(n-1)}) \wedge (data(l_2) \geq L_{2(n-1)} \wedge data(l_2) \leq U_{2(n-1)}) \wedge \ldots \wedge (data(l_m) \geq L_{m(n-1)} \wedge data(l_m) \leq U_{m(n-1)}))$ *then class* $= C_{(n-1)}$
else
class $= C_n$

In the above rules L_{ik} and U_{ik} are the corresponding data ranges in the ***DRM***.

In order to have a better classification, the rules are written in descending order based on the number of attributes covered by them, i.e., a rule with the highest number of attributes is given preference. The algorithm for the rule construction phase is given below:

//Rule Construction//

Step 1. *Arrange k in descending order of the number of attributes corresponding to each class* C_k

Step 2. *For* $k = 1$ *to* n ***do steps 3 to 4***

Step 3. *Set* $j = 1$

Step 4. *For* $i = 1$ *to* m
Step 4.1. *If* $cep_{ik} > \alpha * ep_i$, *then* $cn_j = (data(l_i) \geq L_{ik} \wedge data(l_i))$
Step 4.1.1. *If* $j = 1$, *then* $cn = cn_j$
Else, $cn = cn \wedge cn_j$
Step 4.1.2. *Increment* j *by* 1.

Step 5. *Write the rule for class k using if-then rule format*
Step 5.1. $R_k = (if\ cn\ then\ class = C_k)$

3.5 Rule pruning

The rule pruning step removes the irrelevant conditions from the rule set as shown in Fig. 7.

Acc_r is the accuracy of the initial rule R_k. The TRENN algorithm calculates current accuracy Acc_j by removing a condition cn_j from R_k. If $c_j > Acc_r$, the algorithm adds the removed condition to set ***D***. Next, it sequentially adds back the removed conditions from the set ***D*** one by one, if the accuracy strictly increases after adding a removed condition. The algorithm for the rule pruning phase is given below:

	C_1	C_2	C_3	...	C_n
l_1	$[L_{11}, U_{11}]$	$[L_{12}, U_{12}]$	$[L_{13}, U_{13}]$	...	$[L_{1n}, U_{1n}]$
l_2	$[L_{21}, U_{21}]$	$[L_{22}, U_{22}]$	$[L_{23}, U_{23}]$	...	$[L_{2n}, U_{2n}]$
l_3	$[L_{31}, U_{31}]$	$[L_{32}, U_{32}]$	$[L_{33}, U_{33}]$	...	$[L_{3n}, U_{3n}]$
...	...	...	...	...	...
l_m	$[L_{m1}, U_{m1}]$	$[L_{m2}, U_{m2}]$	$[L_{m3}, U_{m3}]$	...	$[L_{mn}, U_{mn}]$

Fig. 5 Data range matrix

//Rule pruning//

Step 1. *For each R_k do **Step 2.** to **Step 6.***

Step 2. *Initialize $D = 0$*
For each cn_j in R_k:
Step 2.1. *Find maximum Acc_j after removing cn_j*

Step 3. *If $Acc_j \geq Acc_r$ then:*
Remove cn_j from R_k
$D = D + cn_j$
Else
*Goto **Step 6.***

Step 4. *For each cn_j in D*
Step 4.1. *Add cn_j to the temporary network*
Step 4.2. *Set Acc_n as new accuracy*

Step 5. *If $Acc_j > Acc_n$ then:*
$Acc_r = Acc_j$
$D = D - cn_j$
*Goto **Step 4.***

Else
*Goto **Step 3.***

Step 6. *Stop.*

3.6 Rule update

The data range generated for an attribute may contain some overlap among different classes. The rule update phase of TRENN improves the accuracy by shifting the upper and lower range of data using a probabilistic approach. Each condition, $\boldsymbol{cn_j}$ in a rule represents an attribute. This attribute consists of one lower limit value ($\boldsymbol{L}$) and one upper limit value ($\boldsymbol{U}$). The overlap occurs when the data range of one class coincides with the data range of another. The TRENN considers each value of an attribute in case of discrete and specific range of values in case of continuous. If the data range exists for both classes, TRENN finds the probability (say $\boldsymbol{P_k}$, where $\boldsymbol{k} \in [1, \boldsymbol{n}]$) of each value of the attribute belonging to more than one class. If $\boldsymbol{P_k} > \frac{1}{\boldsymbol{n}}$, where $\boldsymbol{n} = \boldsymbol{numberofclasses}$, then value is assigned to the data range of that attribute for class k. Let min_{ik} and max_{ik} be the new minimum and maximum values of the attribute $\boldsymbol{l_i}$ for class $\boldsymbol{C_k}$. Let $\boldsymbol{NewDRM_{ik}} = [\boldsymbol{min_{ik}}, \boldsymbol{max_{ik}}]$. The classification accuracy of the new rule set is $\boldsymbol{R_{newacc}}$. The algorithm modifies the condition cn_j if $\boldsymbol{R_{newacc}} > \boldsymbol{R_{acc}}$ where $\boldsymbol{R_{acc}}$ be the classification accuracy before updating of rule set $\boldsymbol{R}$. The rule update is repeated for all the attributes.

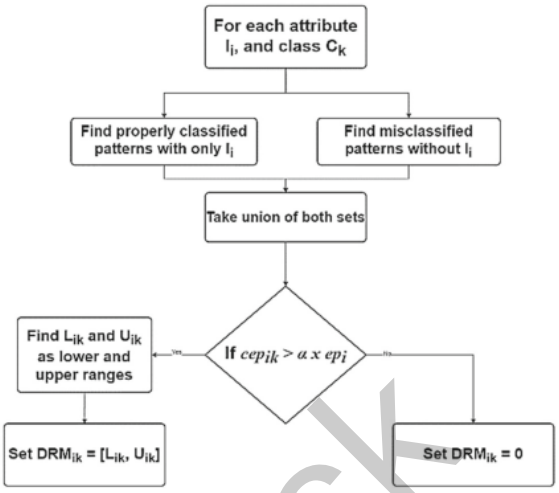

Fig. 6 Flowchart for data range calculation

4 Illustrative example

The working principle of the proposed algorithm is illustrated with the Thoracic Surgery dataset. This dataset consists of four continuous attributes, 12 categorical attributes, and one binary attribute denoting class with labels 'True' and 'False'. 'True' signifies, the target has survived with a successful surgery and 'False' signifies, the target has died due to an unsuccessful surgery. The dataset comprises of 470 examples, and out of them 400 examples have class label 'False' and 70 examples have class label 'True'. The working principle of the proposed TRENN is illustrated with the results of a 80–20 fold, i.e., 80% patterns are taken as a training set and 20% as a testing set.

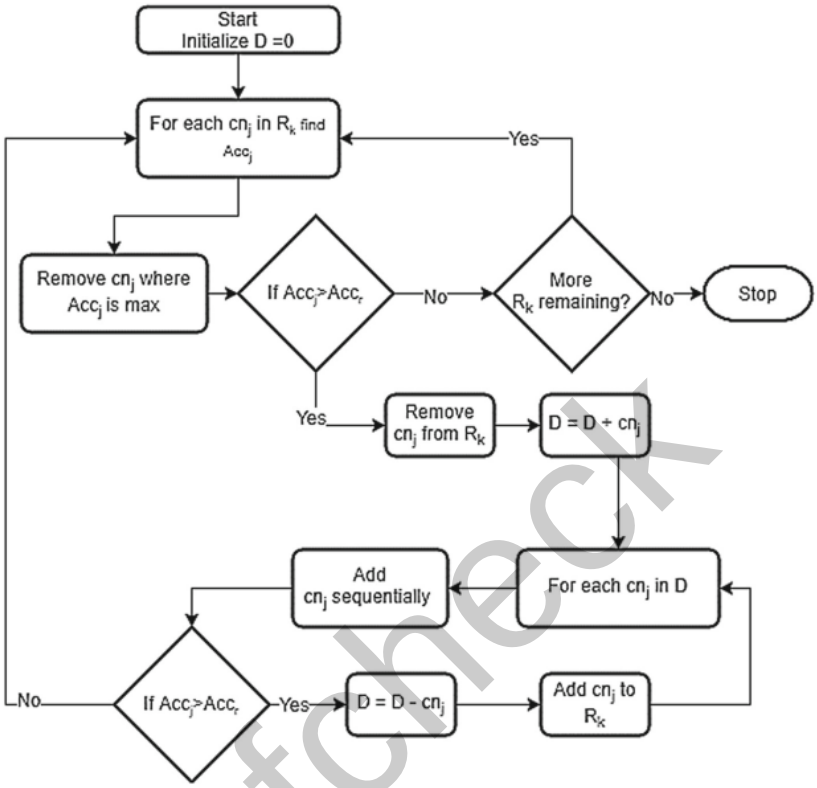

Fig. 7 Flowchart for rule pruning

4.1 Optimal network architecture

Initially, the optimal NN is determined by calculating least mean square error. The number of input neurons is equal to the number of attributes in the dataset along with one extra neuron as a bias input and the number of output neurons is equal to the number of target classes. Given 16 input neurons and 1 bias input, the number of hidden layer neurons varies from 17 to 32 and the architecture 17–27–2 with least mean square error is selected as the optimal network architecture. The NN is trained by the backpropagation learning algorithm with an adaptive learning rate and with a tolerance of 0.00000001. The trained NN produces an accuracy of 82.71% for this fold.

4.2 Network pruning

The TRENN algorithm analyzes the error of the network after removing each input neuron separately. The algorithm removes the neuron with least error from the network, if the training accuracy improves on its removal. The algorithm continues this process for several iterations until training accuracy increases. It keeps the removed neurons to set **B**. For the Thoracic Surgery dataset, the input neurons I_2, I_9, I_7, I_{10}, I_{11}, I_{13}, and I_{16} are found to be insignificant, so the algorithm removes them and adds them to set **B**. Furthermore, the algorithm sequentially adds back the contents of set **B** to the Network to test with different combinations of neurons. The input neurons, I_9 and I_{11} are added back to the network as they improve the accuracy of the Network. After the exclusion and conditional inclusion step the pruned Network is obtained with 11 significant neurons, namely I_1, I_3, I_4, I_5, I_6, I_8, I_9, I_{11}, I_{12}, I_{14} and I_{15}. The pruned Network achieves 85.23% accuracy for this fold. These attributes are used for data range calculation.

4.3 Data range calculation

The data range of each significant attribute for each class is calculated by selecting the misclassified and properly classified patterns using the pruned network. An attribute may or may not be necessary for classifying patterns in all the classes. Therefore, the importance of each attribute in classifying a pattern in the respective class is found using the condition given in Eq. (1).

The data ranges of the significant attributes that satisfy the condition in Eq. (1) are shown in Table 1.

4.4 Rule construction

The initial rule is generated using the lower and upper bound i.e., L_{ik} and U_{ik}, respectively. The rule for a class with the higher number of attributes is constructed first. The rule constructed for this dataset is given below.

if((data(l_1) ≥ 2 ∧ data(l_1) ≤ 6) ∧ (data(l_3) ≥ 1.24 ∧ data(l_3) ≤ 86.3) ∧ (data(l_4) ≥ 0 ∧ data(l_4) ≤ 2) ∧ (data(l_5) ≥ 1 ∧ data(l_5) ≤ 2) ∧ (data(l_6) ≥ 1 ∧ data(l_6) ≤ 2) ∧ (data(l_8) ≥ 2 ∧ data(l_8) ≤ 2) ∧ (data(l_9) ≥ 2 ∧ data(l_9) ≤ 2) ∧ (data(l_{11}) ≥ 1 ∧ data(l_{11}) ≤ 1) ∧ (data(l_{15}) ≥ 1 ∧ data(l_{15}) ≤ 1)) *then*
class = *'False'*
else
class = *'True'*

The initial rule set produces a testing accuracy of 87.54%.

4.5 Rule pruning

The accuracy and comprehensibility of the rule set can be improved by rule pruning. The pruned rule set is shown below.

if((data(l_1) ≥ 2 ∧ data(l_1) ≤ 6) ∧ (data(l_3) ≥ 1.24 ∧ data(l_3) ≤ 86.3) ∧ (data(l_5) ≥ 1 ∧ data(l_5) ≤ 2) ∧ (data(l_6) ≥ 1 ∧ data(l_6) ≤ 2) ∧ (data(l_8) ≥ 2 ∧ data(l_8) ≤ 2) ∧ (data(l_9) ≥ 2 ∧ data(l_9) ≤ 2) ∧ (data(l_{11}) ≥ 1) *then*
class = *'False'*
else
class = *'True'*

This pruned rule produces a testing accuracy of 90.66%.

4.6 Rule updation

The accuracy of the rule set is increased through rule updation. After updation, the accuracy increases to 92.67%. The updated rule set is shown below:

if((data(l_1) ≥ 3 ∧ data(l_1) ≤ 6) ∧ (data(l_3) ≥ 1.24 ∧ data(l_3) ≤ 80.6) ∧ (data(l_5) ≥ 1 ∧ data(l_5) ≤ 1) ∧ (data(l_6) ≥ 1 ∧ data(l_6) ≤ 2) ∧ (data(l_8) ≥ 2 ∧ data(l_8) ≤ 2) ∧ (data(l_9) ≥ 2 ∧ data(l_9) ≤ 2) ∧ (data(l_{11}) ≥ 1) *then*
class = *'False'*
else
class = *'True'*

The local comprehensibility of the rule set is 7. TRENN is an extension to the RxNCM algorithm, so the detailed comparison between the algorithms with the 80–20 fold is shown below in Table 2. A distinction point is observed after the Network Pruning phase, where TRENN results in a better predictive accuracy with a lower number of input features. After the Rule Pruning phase, TRENN results in classification rules with fewer features and higher predictive accuracy.

5 Experimental results

The proposed TRENN algorithm is implemented on eight datasets. The detailed description of the datasets is shown in Table 3. The performance of the algorithm is validated with the results of fivefold cross-validation.

The optimal architectures for the datasets are obtained by calculating the mean square error of the network architectures with hidden nodes ranging from (h = l + 1) to (h = 2*l), where h is the number of hidden layer neurons and l is the number of input neurons. The optimal network architectures for all the datasets are shown in Table 4.

5.1 Results and comparisons

Other than RxNCM, the performance of TRENN is compared with two other popular Rule extraction algorithms: Re-RX (Setiono et al. 2008) and RxREN) (Augasta and

Table 1 Data range of misclassified and properly classified patterns

Input neurons (attributes)	Data range of misclassified and properly classified patterns	
	Class = 'True'	Class = 'False'
I_1	–	[2, 6]
I_3	–	[1.24, 86.3]
I_4	–	[0, 2]
I_5	–	[1, 2]
I_6	–	[1, 2]
I_8	–	[2, 2]
I_9	–	[2,2]
I_{11}	–	[1, 1]
I_{12}	[1,2]	–
I_{14}	[1,1]	–
I_{15}	–	[1,1]

Kathirvalavakumar 2012). The performances of TRENN, RxNCM, RxREN, and Re-RX are evaluated in terms of accuracy, local comprehensibility, and global comprehensibility.

Accuracy: Accuracy is defined as the percentage of the examples correctly classified.

Local Comprehensibility (LC): LC measures the number of conditions per rule.

Global Comprehensibility (GC): GC measures the number of rules.

Table 5 shows the comparison between the algorithms with fivefold cross-validation accuracy. The results show that the TRENN performs better than Re-RX, RxREN and RxNCM in all datasets except ILPD, Liver Disorder. In the case of ILPD after the initial pruning, only one significant attribute remains for TRENN, RxREN, and RxNCM, so their results are the same. But, Re-RX uses recursive rule generation, so it obtains better predictive accuracy for ILPD. In the case of Liver Disorder, Re-RX again has the highest predictive accuracy which can be credited to its recursive rule generation approach. TRENN produces less accuracy than RxREN and RxNCM for ILPD and Liver Disorder datasets because the final rule obtained in the rule updating phase for TRENN uses a probabilistic approach for shifting the data ranges. This procedure resolves the class overlap but may not improve accuracy. For the Breast Cancer dataset, the accuracy for TRENN is better than Re-RX and RxREN and similar to that of RxNCM. This is due to the fact that the initial pruning done for TRENN in this dataset gives the same result as that of RxNCM. Figure 8 shows the graphical comparison between the algorithms with accuracy for better understanding.

Table 2 Comparison of RxNCM and TRENN for 80–20-fold

Process	RxNCM	TRENN
Initial accuracy	82.71%	82.71%
Number of Attributes after pruning	14	12
Accuracy after pruning	83.74%	85.23%
Initial rule set accuracy	84.94%	87.54%
Initial number of rule conditions	14	11
Rule set accuracy after rule pruning	88.57%	90.66%
Conditions after rule pruning	9	7
Accuracy of rule set after updating	90.42%	92.67%

Table 6 shows the comparison between TRENN, RxNCM, RxREN, and Re-RX in terms of LC. If the number of attributes is lesser then comprehensibility is better. The TRENN shows better comprehensibility for the four datasets—Bank Marketing, Thoracic Surgery, Ionosphere and Heart. For the rest, it shows equal. Figure 9 depicts the graphical comparison with LCs.

Table 7 shows the global comprehensibility for all the algorithms. It is observed from the results that global comprehensibility of TRENN is better or equal compared to other algorithms in all the datasets.

The performance of the proposed algorithm is also shown with some additional performance measures, namely Precision, Recall, FP-Rate, and F-measure.

Precision: Precision is the ratio of the correctly labeled positive class to all the positive class labeled.

Recall: Recall is the ratio of the correctly labeled positive class to all the actually positive classes.

FP-Rate: False Positive rate is the probability of the model making a false classification.

F-measure: F-measure is the harmonic mean of Precision and Recall. This measures the balance of Precision and Recall obtained by the model.

Tables 8, 9, 10, and 11 show a comparison result of fivefold cross-validation between the algorithms with

Table 3 Description of the datasets

Dataset	Number of patterns	Number of attributes	Number of classes
Bank marketing	4119	16	2
Sonar	200	60	2
ILPD	583	9	2
Liver disorder	345	6	2
Thoracic surgery	470	16	2
Ionosphere	690	34	2
Heart	270	13	2
Breast cancer	683	9	2

Table 4 Optimal network architecture

Dataset	Optimal architecture
Bank marketing	16-23-1
Sonar	60-93-1
ILPD	9-14-1
Liver disorder	6-9-1
Thoracic surgery	16-27-1
Ionosphere	33-40-1
Heart	13-21-1
Breast cancer	9-13-1

Precision, Recall, F-measure, and FP-Rate, respectively. Table 8 shows that for the Bank Marketing, ILPD, Ionosphere, and Breast Cancer datasets, TRENN produces better Precision compared to others. In case of Sonar, TRENN performs second best and only slightly less than RxrREN. For Thoracic Surgery, TRENN has slightly lesser Precision than RxNCM but better than the others. For Liver Disorder and Heart, Re-RX has the highest Precision and TRENN obtains the second highest Precision. Table 9 shows that for the Bank Marketing, Sonar, Liver Disorder, and Thoracic Surgery datasets, TRENN produces better Recall compared to others. For ILPD, TRENN has second best Recall after RxREN. For Ionosphere, TRENN has

Table 5 Comparison of accuracy for fivefold cross-validation

Dataset	Re-RX (%)	RxREN (%)	RxNCM (%)	TRENN (%)
Bank marketing	89.84	84.27	88.57	90.00
Sonar	70.24	76.04	77.14	80.00
ILPD	72.41	71.18	71.18	71.18
Liver disorder	55.35	54.66	54.66	54.28
Thoracic surgery	90.09	89.36	89.36	91.48
Ionosphere	72.28	90.85	91.43	91.66
Heart	72.59	71.78	70.37	74.07
Breast cancer	90.66	91.73	95.58	95.58

Fig. 8 Graphical comparison with accuracy

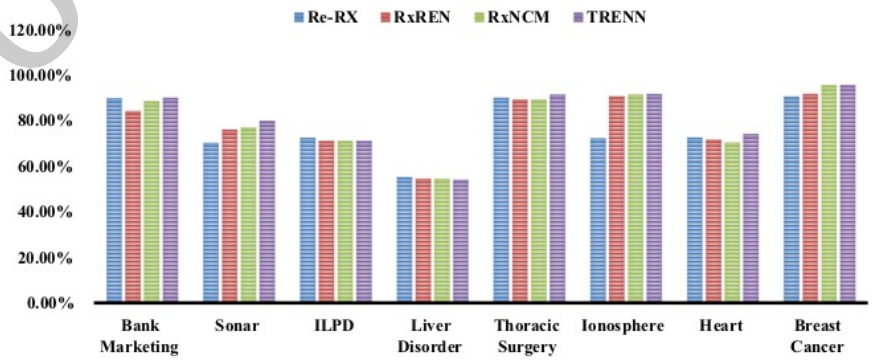

Table 6 Comparison of local comprehensibility for fivefold cross-validation

Dataset	Re-RX	RxREN	RxNCM	TRENN
Bank marketing	9	8	9	7
Sonar	9	4	3	3
ILPD	1	1	1	1
Liver disorder	3	2	2	2
Thoracic surgery	9	9	9	8
Ionosphere	6	4	4	3
Heart	28	3	3	2
Breast cancer	3	3	2	2

Table 7 Comparison of global comprehensibility for fivefold cross-validation

Dataset	Re-RX	RxREN	RxNCM	TRENN
Bank marketing	5	2	2	2
Sonar	2	2	2	2
ILPD	2	2	2	2
Liver disorder	3	2	2	2
Thoracic surgery	5	2	2	2
Ionosphere	2	2	2	2
Heart	10.4	2	2	2
Breast cancer	2	2	2	2

second best Recall after RxNCM. For Heart and Breast Cancer, TRENN has second best Recall after Re-RX. Table 10 shows that for the Bank Marketing, Sonar, ILPD, Liver Disorder, and Thoracic Surgery datasets, TRENN produces better F-measure compared to others. For Heart, TRENN has second best performance after Re-RX. For Ionosphere and Breast Cancer, TRENN has second best performance after RxNCM. Table 11 shows that for the Liver Disorder, Thoracic Surgery, Ionosphere, Heart and Breast Cancer datasets, TRENN produces better FP-Rate compared to others. For Bank Marketing and Sonar, TRENN has third best FP-Rate after Re-RX and RxREN. For ILPD, TRENN has second best FP-Rate after Rx-RX. Overall, the results show that TRENN produces better performance on the majority of the datasets.

Table 8 Comparison of precision for fivefold cross-validation

Dataset	Re-RX	RxREN	RxNCM	TRENN
Bank marketing	0.8943	0.8542	0.9104	0.9117
Sonar	0.5400	0.6741	0.6666	0.6666
ILPD	0.6981	0.7333	0.7547	0.7547
Liver disorder	0.5566	0.5473	0.5526	0.5517
Thoracic surgery	0.9291	0.9566	0.9761	0.9756
Ionosphere	0.6874	0.9174	0.9411	0.9444
Heart	0.7423	0.6539	0.6363	0.7
Breast cancer	0.8141	0.9188	0.9365	0.9565

6 Conclusion

The proposed TRENN algorithm converts Neural Network from black box system to white-box system by extracting the knowledge learned by the network in the form of

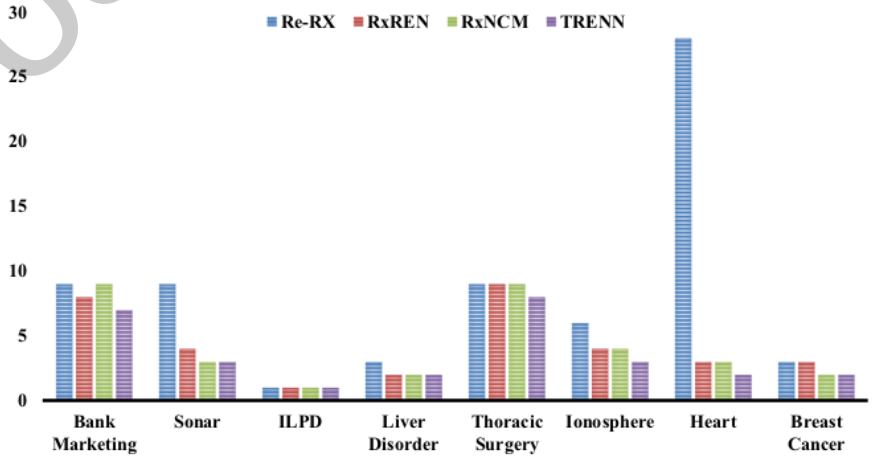

Fig. 9 Graphical comparison with LC

Table 9 Comparison of recall for fivefold cross-validation

Dataset	Re-RX	RxREN	RxNCM	TRENN
Bank marketing	0.9216	0.8814	0.9682	0.9841
Sonar	0.4202	0.5397	0.7692	0.8333
ILPD	0.8853	0.9147	0.9090	0.9090
Liver disorder	0.7913	0.7362	0.8421	0.8444
Thoracic surgery	0.8867	0.9011	0.9111	0.9302
Ionosphere	0.9054	0.8934	0.8947	0.8888
Heart	0.6559	0.6149	0.6363	0.6363
Breast cancer	0.9521	0.8733	0.9278	0.9166

Table 10 Comparison of *F*-measure for fivefold cross-validation

Dataset	Re-RX	RxREN	RxNCM	TRENN
Bank marketing	0.9077	0.8676	0.9384	0.9465
Sonar	0.4712	0.5994	0.7142	0.7406
ILPD	0.7806	0.8247	0.8246	0.8246
Liver disorder	0.6535	0.6278	0.6666	0.6666
Thoracic surgery	0.9074	0.9280	0.9424	0.9523
Ionosphere	0.7519	0.9052	0.9188	0.9142
Heart	0.6763	0.6338	0.6363	0.6666
Breast cancer	0.8747	0.8955	0.9461	0.9361

Table 11 Comparison of FP-Rate for fivefold cross-validation

Dataset	Re-RX	RxREN	RxNCM	TRENN
Bank marketing	0.7185	0.7654	0.8571	0.8571
Sonar	0.1938	0.2077	0.2272	0.2173
ILPD	0.8542	0.8719	0.8666	0.8666
Liver disorder	0.8191	0.8333	0.8254	0.8125
Thoracic surgery	0.4132	0.4761	0.5	0.25
Ionosphere	0.5421	0.0695	0.0588	0.0555
Heart	0.2168	0.2336	0.25	0.1875
Breast cancer	0.1174	0.0579	0.0217	0.0217

human-understandable rules. The TRENN algorithm extracts rules by mapping the relationship between the input neurons and the output neurons. The algorithm initiates with finding the optimal network architecture, followed by pruning the network to remove the irrelevant features or attributes and calculating the data ranges of the significant features to construct rules. The algorithm further refines the constructed rules by pruning the rules and updating the data ranges of the features. The TRENN algorithm extends the RxNCM algorithm. The novelty of the TRENN algorithm lies in the network pruning, rule pruning, and rule updating. The TRENN algorithm employs the backward floating search technique for pruning the network and the rules to improve the performance of the RxNCM algorithm, specifically the comprehensibility of the generated rules. The backward floating search prevents nesting effects that take place in the sequential feature selection search. TRENN also uses a probabilistic approach in the rule updation phase to overcome the inherent overlap of classes. TRENN removes the overlap by shifting the upper and lower ranges of data according to the class probability of each value of an attribute.

The performance of the algorithm is validated with 8 real-life datasets taken from the UCI repository. Results show the effectiveness of the proposed algorithm in terms of accuracy along with different performance measures. The rules generated by the proposed TRENN algorithm are more comprehensible and accurate compared to RxNCM algorithm. Other than RxNCM, TRENN is also compared with two other algorithms Re-RX and RxREN. All the results support that TRENN is an effective algorithm for interpreting the decisions made by a Neural Network in the form of human-understandable form. The algorithm can be used in many applications like medical diagnosis, banking problems and others. This rule extraction algorithm can be further extended by adopting a novel pruning technique and a better technique to solve the overlapping of data ranges of attributes in different classes.

Compliance with ethical standards

Conflict of interest Abhinaba Dattachaudhuri declares that he has no conflict of interest. Saroj Kr. Biswas declares that he has no conflict of interest. Manomita Chakraborty declares that she has no conflict of interest. Sunita Sarkar declares that she has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Augasta MG, Kathirvalavakumar T (2012) Reverse engineering the neural networks for rule extraction in classification problems. Neural Process Lett 35:131–150

Biswas SK, Chakraborty M, Purkayastha B, Roy P, Thounaojam DM (2017) Rule extraction from training data using neural network. Int J Artif Intell Tools 26:1–26

Bologna G, Hayashi Y (2018) A comparison study on rule extraction from neural network ensembles, boosted shallow trees, and SVMs. Appl Comput Intell Soft Comput 2018:1–20

Bondarenko A, Aleksejeva L, Jumutc V, Borisov A (2017) Classification tree extraction from trained artificial neural networks. Proc Comput Sci 104:556–563

Botari T, Izbicki R, Carvalho A (2019) Local interpretation methods to machine learning using the domain of the feature space. ECML PKDD 2019:241–252

Caruana R, Niculescu-Mizil A (2007) An empirical comparison of supervised learning algorithms. In: ICML '06 proceedings of the 23rd international conference on machine learning, pp 161–162

Cohen S, Rokach L, Maimon O (2007) Decision-tree instance-space decomposition with grouped gain ratio. Inf Sci 177:3592–3612

Craven M, Shavlik J (1996) Extracting tree-structured representations of trained networks. Adv Neural Inform Process Syst NIPS 8:24–30

Dam HH, Abbass HA, Lokan C, Yao X (2008) Neural-based learning classifier systems. IEEE Trans Knowl Data Eng 20:26–39

Gupta A, Park S, Lam SW (1999) Generalized analytic rule extraction for feedforward neural networks. IEEE Trans Knowl Data Eng 11:985–991

Han J, Kambler M (2001) Data mining: concepts and techniques. Morgan Kaufmann, San Francisco

Hruschka ER, Ebeckenb NFF (2006) Extracting rules from multilayer perceptrons in classification problems: a clustering-based approach. Neurocomputing 70:384–397

Iqbal RA (2012) Eclectic rule extraction from neural networks using aggregated DTs. In: IEEE, 7th international conference on electrical and computer engineering (ICECE), pp 129–132

Jivani K, Ambasana J, Kanani S (2014) A survey on rule extraction approaches based techniques for data classification using neural network. Int J Futur Trends Eng Technol 1:4–7

Kaikhah K, Doddmeti S (2006) Discovering trends in large datasets using neural network. Appl Intel 29:51–60

Karim A, Zhou S (2015) X-TREPAN: a multi class regression and adapted extraction of comprehensible decision tree in artificial neural networks. Comput Sci Inform Technol 5:37–54

Kaviani P, Dhotre S (2017) Short survey on Naive Bayes algorithm. Int J Adv Res Comput Sci Manag 4:607–611

Lu H, Setiono R, Liu H (1995) NeuroRule: a connectionist approach to data mining. In: Proceedings of the 21st VLDB, pp 478–489

Mann AK, Kaur N (2013) Survey paper on clustering techniques. Int J Sci Eng Technol Res IJSETR 2:803–806

Mantas CJ, Puche JM, Mantas JM (2006) Extraction of similarity based fuzzy rules from artificial neural networks. Int J Approx Reason 43:202–221

Mashayekhi M, Gras R (2015) Rule extraction from random forest: the RF+HC methods. Springer International Publishing, Switzerland, pp 223–237

Odajimaa K, Hayashi Y, Tianxia G, Setiono R (2008) Greedy rule generation from discrete data and its use in neural network rule extraction. Neural Netw 21:1020–1028

Sestito S, Dillon TS (1992) Automated knowledge acquisition of rules with continuously valued attributes. In: 12th international conference on expert systems and their applications, pp 645–656

Setiono R, Liu H (1995) Understanding neural networks via rule extraction. In: 14th international joint conference on artificial intelligence, pp 480–485

Setiono R, Liu H (1996) Symbolic representation of neural networks. IEEE Comput 29:71–77

Setiono R, Kheng W (2000) FERNN: an algorithm for fast extraction of rules from neural networks. Appl Intell 12:15–25

Setiono R, Baesens B, Mues C (2008) Recursive neural network rule extraction for data with mixed attributes. IEEE Trans Neural Netw 19:299–307

Sharma AK, Shani S (2011) A comparative study of classification algorithms for spam email data analysis. Int J Comput Sci Eng 3:1890–1895

Shridhar M, Parmar M (2017) Survey on association rule mining and its approaches. Int J Comput Sci Eng 5:129–135

Sing V, Midha N (2015) A survey on classification techniques in data mining. Int J Comput Sci Manag Stud 16:9–12

iThenticate sample report

ORIGINALITY REPORT

24%

SIMILARITY INDEX

PRIMARY SOURCES

1	Saroj Kumar Biswas, Manomita Chakraborty, Biswajit Purkayastha, Pinki Roy, Dalton Meitei Thounaojam. "Rule Extraction from Training Data Using Neural Network", International Journal on Artificial Intelligence Tools, 2016 Crossref	289 words	5%
2	M. Gethsiyal Augasta, T. Kathirvalavakumar. "Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems", Neural Processing Letters, 2011 Crossref	206 words	3%
3	link.springer.com Internet	162 words	3%
4	Abhinaba Dattachaudhuri, Saroj Biswas, Sunita Sarkar, Arpita Nath Boruah. "Transparent Decision Support System for Credit Risk Evaluation: An automated credit approval system", 2020 IEEE-HYDCON, 2020 Crossref	112 words	2%
5	Manomita Chakraborty, Saroj Kumar Biswas, Biswajit Purkayastha. "Data Mining Using Neural Networks in the form of Classification Rules: A Review", 2020 4th International Conference on Computational Intelligence and Networks (CINE), 2020 Crossref	75 words	1%
6	Abhinaba Dattachaudhuri, Saroj Kr. Biswas, Sunita Sarkar, Arpita Nath Boruah, Manomita Chakraborty, Biswajit Purkayastha. "Transparent Neural based Expert System for Credit Risk (TNESCR): An Automated Credit Risk Evaluation System", 2020 International Conference on Computational	65 words	1%

Performance Evaluation (ComPE), 2020
Crossref

7	Manomita Chakraborty, Saroj Kumar Biswas, Biswajit Purkayastha. "Rule Extraction from Neural Network Using Input Data Ranges Recursively", New Generation Computing, 2018 Crossref	56 words — 1%
8	Saroj Kr. Biswas, Manomita Chakraborty, Biswajit Purkayastha. "A rule generation algorithm from neural network using classified and misclassified data", International Journal of Bio-Inspired Computation, 2018 Crossref	49 words — 1%
9	Manomita Chakraborty, Saroj Kr. Biswas, Biswajit Purkayastha. "Recursive Rule Extraction from NN using Reverse Engineering Technique", New Generation Computing, 2018 Crossref	43 words — 1%
10	ukdiss.com Internet	36 words — 1%
11	Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, Jun Sakuma. "Model-based and actual independence for fairness-aware classification", Data Mining and Knowledge Discovery, 2017 Crossref	22 words — < 1%
12	Manomita Chakraborty, Saroj Kumar Biswas, Biswajit Purkayastha. "Rule extraction from neural network trained using deep belief network and back propagation", Knowledge and Information Systems, 2020 Crossref	22 words — < 1%
13	Lisa Goberdhan, Stuart Kininmonth. "Insights into coral growth rate trends in Fiji", Coral Reefs, 2021 Crossref	20 words — < 1%
14	www.tandfonline.com Internet	18 words — < 1%

15	research.cs.queensu.ca Internet	15 words — < 1%
16	A. Gupta, Sang Park, S.M. Lam. "Generalized Analytic Rule Extraction for feedforward neural networks", IEEE Transactions on Knowledge and Data Engineering, 1999 Crossref	15 words — < 1%
17	"Progress in Advanced Computing and Intelligent Engineering", Springer Science and Business Media LLC, 2021 Crossref	15 words — < 1%
18	Khosrow Kaikhah, Sandesh Doddameti. "Discovering Trends in Large Datasets Using Neural Networks", Applied Intelligence, 2006 Crossref	14 words — < 1%
19	Lecture Notes in Computer Science, 2010. Crossref	14 words — < 1%
20	qmro.qmul.ac.uk Internet	13 words — < 1%
21	"Hybrid Artificial Intelligent Systems", Springer Science and Business Media LLC, 2020 Crossref	11 words — < 1%
22	cava.cs.utsa.edu Internet	11 words — < 1%
23	mafiadoc.com Internet	11 words — < 1%
24	www.biostat.wisc.edu Internet	10 words — < 1%
25	Veronica K.H. Chan, Christine W. Chan. "Towards explicit representation of an artificial neural network model: Comparison of two artificial neural network rule extraction	10 words — < 1%

approaches", Petroleum, 2019
Crossref

26	www.ogretmenx.com Internet	10 words — < 1%
27	Veronica Chan, Christine W. Chan. "Development and application of an algorithm for extracting multiple linear regression equations from artificial neural networks for nonlinear regression problems", 2016 IEEE 15th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC), 2016 Crossref	9 words — < 1%
28	www.igi-global.com Internet	9 words — < 1%
29	Rafael Alcalá, Yusuke Nojima, Francisco Herrera, Hisao Ishibuchi. "Multiobjective genetic fuzzy rule selection of single granularity-based fuzzy classification rules and its interaction with the lateral tuning of membership functions", Soft Computing, 2010 Crossref	9 words — < 1%
30	ixa.si.ehu.es Internet	9 words — < 1%
31	epdf.tips Internet	8 words — < 1%
32	Christie M. Fuller, Rick L. Wilson. "chapter 25 Extracting Knowledge from Neural Networks", IGI Global, 2006 Crossref	8 words — < 1%
33	arxiv4.library.cornell.edu Internet	8 words — < 1%
34	Azza Allouch, Anis Koubaa, Tarek Abbes, Adel Ammar. "RoadSense: Smartphone Application to Estimate Road Conditions Using Accelerometer and Gyroscope", IEEE Sensors Journal, 2017 Crossref	7 words — < 1%

35	Dounia Yedjour, Abdelkader Benyettou. "Symbolic interpretation of artificial neural networks based on multiobjective genetic algorithms and association rules mining", Applied Soft Computing, 2018 Crossref	7 words — < 1%
36	arxiv.org Internet	7 words — < 1%
37	"Data Mining and Knowledge Discovery Handbook", Springer Science and Business Media LLC, 2010 Crossref	6 words — < 1%
38	Manomita Chakraborty, Saroj Kumar Biswas, Biswajit Purkayastha. "A novel ensembling method to boost performance of neural networks", Journal of Experimental & Theoretical Artificial Intelligence, 2019 Crossref	6 words — < 1%

EXCLUDE QUOTES	ON	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	ON		