Rule Extraction from Artificial Neural Networks

Rudy Setiono
School of Computing
National University of Singapore
Outline

1. The train problem
2. Motivations
3. Feedforward neural networks for classification
4. Rule extraction from neural networks
5. Examples
6. Different types of classification rules
7. Regression rules
8. Hierarchical rules: The ReRX algorithm
9. Conclusions
1. The train problem

<table>
<thead>
<tr>
<th>Westbound trains</th>
<th>Eastbound trains</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Train Diagram]</td>
<td>![Train Diagram]</td>
</tr>
</tbody>
</table>

Attributes of a train:

- long cars can only be rectangular, and if closed then their roofs are either jagged or flat
- if a short car is rectangular then it is also double sided
- a short closed rectangular car can have either a flat or peaked roof
The train problem

Attributes of a train:

✓ a long car can have either two or three axels
✓ a car can be either open or closed
✓ a train has 2,3 or 4 cars, each can be either short or long
✓ .......
The train problem

Westbound trains

Answers:

✓ if a train has short closed car, then it is westbound, otherwise eastbound

✓ if a train has two cars, or has a car with a jagged roof, then it is eastbound, otherwise westbound.

✓ and many others..

All the above rules can be obtained by neural networks!
2. Motivations

• Neural networks have been applied to solve many application problems involving
  - pattern classification
  - function approximation/data fitting
  - data clustering

• They often give better predictive accuracy than other methods such as regression or decision trees.

• **Data mining using neural networks:** if we can extract rules from a trained network, a better understanding about the data and the problem can be gained.

• How to extract such rules?
3. Feedforward neural networks for pattern classification

- Data is fed into the network input units.
- Pattern classification is determined by the output unit with the largest output value.
- Units in the hidden layer allow the network to separate any number of disjoint sets.
Network hidden units

For each unit:

- Sum of the weighted inputs is computed:
  \[ \text{net} = \mathbf{I}^t \mathbf{W} \]

- A nonlinear function is applied to obtained the unit’s activation value:
  \[ o = f(\text{net}) \]

- This activation function is usually the logistic sigmoid function (unipolar) or the hyperbolic tangent function (bipolar).

$\mathbf{I}_N$ is usually fixed, $\mathbf{I}_N = 1$
- The function is used to approximate the on-off function.
- The sum of the weighted inputs large \( \Rightarrow \) output is close to 1 (on).
- The sum of the weighted inputs small \( \Rightarrow \) output is close to -1 (off).
- Differentiable:
  \[ f'(net) = \frac{(1-o^2)}{2} \]
  where \( o = f(net) \)
- Derivative is largest when \( o = 0 \), that is when \( net = 0 \)
- and approaches 0 as \(|net|\) becomes large.
Neural network training

• Given a set of data, minimise the total errors:

\[ \sum_i (\text{target}_i - \text{predicted}_i)^2 \]

• Supervised learning.

• Nonlinear optimisation problem: find a set of neural network weights that minimises the total errors.

• Optimisation methods used: backpropagation/gradient descent, quasi-Newton method, conjugate gradient method, etc.

• A penalty term is usually added to the error function so that redundant connections have small/zero weights.

• An example of an augmented error function:

\[ \sum_i (\text{target}_i - \text{predicted}_i)^2 + C \sum_j w_j^2 \]

  - \( N = \# \text{ of samples} \)
  - \( K = \# \text{ of weights} \)
  - \( C \) is a penalty parameter
Neural network pruning

- After a network has been trained, redundant connections and units are removed by pruning.
- Pruned networks generalise better: they can predict new patterns better than fully connected networks.
- Simple classification rules can be extracted from skeletal pruned networks.
- Various methods for network pruning can be found in the literature.
Neural network pruning

A Simple Pruning Algorithm:

1. Start with a trained fully connected network.

2. Identify potential connection for pruning (for example, one with small magnitude).

3. Set the weight of this connection to 0.

4. Retrain the network (if necessary).

5. If the network still meets the required accuracy, go to step 2.

6. Otherwise, restore the removed connection and its corresponding weight. Stop.
4. Rule extraction from neural networks

1. Train and prune a network with a single hidden layer

2. Cluster the hidden unit activation values:
   - Original activation values lies in [-1,1]
   - Clustering implies dividing this interval into subintervals, for example [-1,-0.8), [-0.8,0.5), [0.5,1]
   - An algorithm is needed to ensure the network does not lose its accuracy

3. Generate classification rules in terms of clustered activation values

4. Generate rules which explain the clustered activation values in terms of the input data attributes

5. Merge the two sets of rules

Decompositional approach!
Rule extraction by decompositional approach
5. Example: Iris classification problem

- 150 instances.
- 4 continuous attributes: sepal length, sepal width, petal length, petal width.
- Three different iris flowers:

  setosa  versicolor  virginica
A network with 2 hidden units

- Three class problem: 3 output units.
- Four input attributes: 4 input units + 1 for bias.
- The network has only 2 hidden units and 10 connections after pruning.
- It correctly classifies all but one training pattern.
- 2-dimensional plot of the activation values.
Rule in terms of the hidden unit activations:
If $H_1 > -0.7$: Iris setosa
Else if $H_2 \leq -0.55$: Iris versicolor
Else: Iris virginica
Iris classification rules

If petal length $> 2.23$, then Iris *setosa*.

Else if

$$3.57 \text{ petal length} + 3.56 \text{ petal width} - \text{sepal length} - 1.57 \text{ sepal width} > 12.63,$$
then Iris *versicolor*.

Else Iris *virginica*.
Example: Breast cancer diagnosis

- Nine measurements taken from fine needle aspirates of human breast tissues: clump thickness, uniformity of cell size, uniformity of cell shape, etc.
- Each measurement integer valued 0 to 10.
- 458 benign samples and 241 malignant samples from 699 patients.
- Data is split into 350 training samples and 349 test samples.
- 100 neural networks were trained:
  - Original number of hidden units: 5
  - Original number of connections: 460
- After pruning:
  - Average number of connections: 10.70
  - Average predictive accuracy: 92.70%.
Breast Cancer Diagnosis: Example 1

- Extracted rules:
  
  If uniformity of cell size ≤ 4 and bare nuclei ≤ 5, then benign.
  Else malignant.

- Predictive accuracy: 93.98%
Breast Cancer Diagnosis: Example 2

- If
  - clump thickness ≤ 6,
  - bland chromatin ≤ 3, and
  - normal nucleoli ≤ 9, then benign.
Else malignant.

- Predictive accuracy: 93.12%.
Example: Application to hepatobiliary disorders

- Data collected from 536 patients in a Japanese hospitals.
- Nine real-valued measurements obtained from biomedical tests.
- Patients are diagnosed as having one of the 4 liver disorders (ALD, PH, LC, or C).
- Accuracy from different methods:

<table>
<thead>
<tr>
<th></th>
<th>Linear discriminant analysis</th>
<th>Fuzzy neural networks</th>
<th>Neural network extracted rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALD</td>
<td>57.6%</td>
<td>69.7%</td>
<td>87.9%</td>
</tr>
<tr>
<td>PH</td>
<td>64.7%</td>
<td>82.4%</td>
<td>92.2%</td>
</tr>
<tr>
<td>LC</td>
<td>65.7%</td>
<td>71.4%</td>
<td>80.0%</td>
</tr>
<tr>
<td>C</td>
<td>63.6%</td>
<td>81.8%</td>
<td>90.9%</td>
</tr>
<tr>
<td>Total</td>
<td>63.2%</td>
<td>77.3%</td>
<td>88.3%</td>
</tr>
</tbody>
</table>
Example: LED display recognition

An LED (Light Emitting Diode) device and digits 0, 1, .. 9:
Example: LED display recognition

Must be on
Must be off
Doesn’t matter

0

= 0
Example: LED display recognition

Must be on
Must be off
Doesn’t matter
= 1
Example: LED display recognition

= 2

- **Must be on**
- **Must be off**
- **Doesn’t matter**
Other applications

- Bankruptcy prediction.
- Distinguishing between organisations that adopt IT from those that do not.
- Predicting gene sequences.
- Predicting protein secondary structures.
- Analysis of marketing survey data.
6. Various types of classification rules

1. Propositional: IF .... THEN ...., ELSE ....

2. MofN rules: IF M of the given N conditions are satisfied, then ..... 

3. Fuzzy rules: IF X is large, THEN ......, ELSE IF X is medium, THEN ..... 

4. Oblique rules: If $X_1 + 2X_2 + 5X_3 \geq 100$, THEN ..... 

5. Hierarchical rule set (more on this later).

**Variants of the network rule extraction algorithms have been developed to extract different types of rules!**
7. Regression rules

Least square regression line

Data points

Neural network output

Piecewise regression line

Piecewise linear approximation!
Regression rules (continued)

• Neural network fits the data better than linear regression line.

• The knowledge embedded in a neural network is hard to explain due to the network’s complex nonlinear input-output mapping.

• Piece-wise linear approximation is easier to interpret.

• Interpretability vs accuracy.
Rule extraction from a pruned network

REFANN

(Rule extraction from function approximating neural networks):

- For each hidden unit, approximate the hidden unit activation function by a 3-piece linear function.

- Replace the predicted output of the network by the linear combination of these piece-wise linear functions. These are the rule consequence.

- The rule conditions describe the subspace of the input space and are formed by the conjunction of domains of the 3-piece linear approximating functions of the hidden units.
Approximation of hyperbolic tangent function

- The function \( \tanh(x) \) is approximated by the piecewise linear function (dashed line).
- \( x_m \) is the largest activation value among the training samples.
- \( x_0 \) is computed such that the area between the curve and the dashed line is minimised.
Example 1: Predicting CPU’s relative performance

- Input attributes:
  * MYCT: machine cycle time (nanoseconds)
  * MMIN: minimum main memory (KB)
  * MMAX: maximum main memory (KB)
  * CACH: cache memory (KB)
  * CHMIN: minimum channels
  * CHMAX: maximum channels

- Output: CPU’s relative performance

- Samples: 167 training, 21 x-validation and 21 testing.

- Neural network: 8 hidden units, 1 hidden unit remained after pruning.
Example 1 (continued)

- The activation function of the remaining hidden unit is approximated by
  
  \[-0.5696 + 0.2256x, \text{ if } x < -0.7354\]

  \[x, \text{ if } x \geq 0.7354\] where \(x\) is the weighted input.

- The rules are:

  \[\text{if (weighted inputs } x < -0.7354), \text{ then}\]

  \[\text{predict } Y = (-0.5696 + 0.2256x) \times v\]

  \[\text{otherwise}\]

  \[\text{predict } Y = x \times v\]

  where \(v\) is the neural network weight from the hidden unit to the output unit.
Example 1 (continued)

• The weighted inputs:

\[
\begin{align*}
3.0 \text{ MMIN} + 2.7 \text{ MMAX} + 258.1 \text{ CACH} + 281.5 \text{ CHMAX} &= 111189
\end{align*}
\]

define a separating hyperplane in the input space.

• We replace this hyperplane by the decision boundaries that are axis parallel:

\[
\begin{align*}
\text{If } \text{MMAX} \leq 24000 \text{ and } \text{CACH} \leq 142, \text{ then } Y_1 \\
\text{If } \text{MMIN} \leq 2300 \text{ and } \text{CHMAX} \leq 38, \text{ then } Y_2 \\
\text{If } \text{MMAX} > 2300, \text{ then } Y_2 \\
\text{If } \text{CACH} > 142, \text{ then } Y_2 \\
\text{Default rule: } Y_1
\end{align*}
\]

\[
\begin{align*}
Y_1 &= 4.96 + 0.0036 \text{ MMIN} + 0.0032 \text{ MMAX} + 0.3086 \text{ CACH} + 0.3366 \text{ CHMAX} \\
Y_2 &= -453.02 + 0.0159 \text{ MMIN} + 0.0143 \text{ MMAX} + 1.3662 \text{ CACH} + 1.4903 \text{ CHMAX}
\end{align*}
\]
Predictive error rates:

RMSE: root mean squared errors
RRMSE: relative RMSE
MAE: mean absolute error
RMAE: relative MAE
Example 2: Auto-Mpg data set

- Input attributes:
  - Cylinders: 3, 4, 5, 6, 8
  - Model: 70, 71, 72, …., 82
  - Origin: 1, 2, 3
  - Displacement: continuous value
  - Horsepower: continuous value
  - Weight: continuous value
  - Acceleration: continuous value

- Predict the fuel consumption of different car models.
- 318 training samples, 40 cross-validation samples and 40 test samples.
Example 2 (continued)

Rules:

- If \((model \leq 78)\) and \((\text{horsepower} > 115)\) and \((\text{weight} > 3432)\), then \(Y_1\)
- If \((model \leq 76)\) and \((\text{weight} > 3574)\), then \(Y_1\)
- If \((model \leq 76)\) and \((\text{horsepower} > 130)\), then \(Y_1\)
- If \((\text{horsepower} \leq 98)\), then \(Y_2\)
- If \((\text{horsepower} \leq 130)\) and \((\text{weight} \leq 3432)\), then \(Y_2\)
- If \((\text{model} > 76)\) and \((\text{weight} \leq 3432)\), then \(Y_2\)
- If \((\text{model} > 76)\) and \((\text{horsepower} \leq 115)\), then \(Y_2\)
- If \((\text{model} > 78)\), then \(Y_2\)
- Default rule. \(Y_2\)
8. Hierarchical rules: The ReRX algorithm

- One of the key decisions financial institutions have to make is to decide whether or not to grant credit to a customer who applies for a loan.

- The aim of credit scoring is to develop classification models that are able to distinguish good from bad payers, based on the repayment behaviour of past applicants.

- These models usually summarize all available information of an applicant in a score:

  \[ P(\text{applicant is good payer} \mid \text{age, marital status, savings amount, \ldots}) \].

- Application scoring: if this score is above a predetermined threshold, credit is granted; otherwise credit is denied.
Results from “old” method for credit scoring:

If Term >12 months and Purpose = cash provisioning and Savings Account ≤ 12.40 € and Years Client ≤ 3 then Applicant = bad

If Term >12 months and Purpose = cash provisioning and Owns Property = no and Savings Account ≤ 12.40 € then Applicant = bad

If Purpose = cash provisioning and Income > 719 € and Owns Property = no and Savings Account ≤ 12.40 € and Years Client ≤ 3 then Applicant = bad

If Purpose = second-hand car and Income > 719 € and Owns Property = no and Savings Account ≤ 12.40 € and Years Client ≤ 3 then Applicant = bad

If Savings Account ≤ 12.40 € and Economical sector = Sector C then Applicant = bad

Default class: Applicant = good
Previous methods: Neurorule and Neurolinear

- Neurolinear: Assumes continuous variables (no prior discretization)
- Less comprehensible hyperplane rules, e.g.:

If $[-24.59(\text{Checking account}) + 29.66(\text{Term}) - 16.45(\text{Credit history}) - 3.66(\text{Purpose}) - 18.69(\text{Savings amount}) + 9.29(\text{Instalment rate}) - 18.74(\text{Personal status}) + 6.19(\text{Property}) - 10.03(\text{Age}) - 9.36(\text{Other instalment plans}) - 11.51(\text{Housing}) + 7.15(\text{Existing credits}) + 16.68(\text{Job}) + 2.046(\text{Number of dependents}) - 4.54(\text{Telephone}) - 8.29(\text{Foreign worker})] \leq 0.15$, then customer = good payer, else customer = defaulter
• Neurorule

  – Requires prior *discretization* of continuous variables

  – Simple propositional rules, e.g.:

    If Term > 12 months and Purpose = cash provisioning and Savings amount ≤ 12.40 Euro and Years client ≤ 3, then customer = defaulter

    Age from 0 to 80 years discretized into 4 subintervals
Re-RX Algorithm

- New rule extraction algorithm:
  \textit{Re-RX (Recursive Rule Extraction)}

- Handles mix of discrete/continuous variables without need for discretization of continuous variables
  - Discrete variables: propositional rule tree structure
  - Continuous variables: hyperplane rules at leaf nodes

- Example rule:

  \textbf{If} Years Clients < 5 and Purpose \neq Private Loan, \textbf{then}

  \textbf{If} Number of applicants \geq 2 and Owns real estate = yes, \textbf{then}

  \textbf{If} Savings amount + 1.11 \text{Income} - 38249 \text{Insurance} - 0.46 \text{Debt} > -1939300, \textbf{then}

  Customer = good payer

  Else ...

- Combines comprehensibility and accuracy
Re-RX Algorithm: Detailed Steps

Algorithm Re-RX(S, D, C):

Input: A set of samples S having discrete attributes D and continuous attributes C

Output: A set of classification rules

1. Train and prune a neural network using the data set S and all its attributes D and C.

2. Let D' and C' be the sets of discrete and continuous attributes still present in the network, respectively. Let S' be the set of data samples that are correctly classified by the pruned network.

3. If D' = ∅, then generate a hyperplane to split the samples in S' according to the values of their continuous attributes C' and stop. Otherwise, using only discrete attributes D', generate the set of classification rules R for the data set S'.

4. For each rule Ri generated:
   
   If support(Ri) > δ1 and error(Ri) > δ2, then:
   
   – Let Si be the set of data samples that satisfy the condition of rule Ri, and Di be the set of discrete attributes that do not appear in the rule condition of Ri.
   
   – If Di = ∅, then generate a hyperplane to split the samples in Si according to the values of their continuous attributes Ci and stop.

Otherwise, call Re-RX(Si, Di, Ci)
A Hierarchical Rule Extracted by Re-RX for Bene2

Rule R: If Years Clients < 5 and Purpose ≠ Private Loan, then

Rule R₁: If Number of applicants ≥ 2 and Owns real estate = yes, then

Rule R₁a: If $\text{Savings amount} + 1.11 \times \text{Income} - 38249 \times \text{Insurance} - 0.46 \times \text{Debt} > -1939300$, then customer = good payer

Rule R₁b: Else customer = defaulter

Rule R₂: Else if number of applicants ≥ 2 and Owns real estate = no, then

Rule R₂a: If $\text{Savings amount} + 1.11 \times \text{Income} - 38249 \times \text{Insurance} - 0.46 \times \text{Debt} > -1638720$, then customer = good payer

Rule R₂b: Else customer = defaulter

Rule R₃: Else ...
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>$\text{PCC}_{\text{train}}$</th>
<th>$\text{PCC}_{\text{test}}$</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>German</td>
<td>C5.0 - decision tree</td>
<td>81.98</td>
<td>71.26</td>
<td>27 leaves</td>
</tr>
<tr>
<td></td>
<td>C5.0 - rules</td>
<td>80.78</td>
<td>70.06</td>
<td>14 propositional rules</td>
</tr>
<tr>
<td></td>
<td>Neurolinear</td>
<td>80.93</td>
<td>77.25</td>
<td>2 oblique rules</td>
</tr>
<tr>
<td></td>
<td>Neurorule</td>
<td>75.83</td>
<td>77.25</td>
<td>4 propositional rules</td>
</tr>
<tr>
<td></td>
<td>Re-RX</td>
<td>80.48</td>
<td>80.54</td>
<td>41 propositional rules</td>
</tr>
<tr>
<td>Bnc1</td>
<td>C5.0 - decision tree</td>
<td>78.91</td>
<td>71.06</td>
<td>35 leaves</td>
</tr>
<tr>
<td></td>
<td>C5.0 - rules</td>
<td>78.43</td>
<td>71.37</td>
<td>15 propositional rules</td>
</tr>
<tr>
<td></td>
<td>Neurolinear</td>
<td>77.43</td>
<td>72.72</td>
<td>3 oblique rules</td>
</tr>
<tr>
<td></td>
<td>Neurorule</td>
<td>73.05</td>
<td>71.85</td>
<td>6 propositional rules</td>
</tr>
<tr>
<td></td>
<td>Re-RX</td>
<td>75.07</td>
<td>73.10</td>
<td>39 propositional rules</td>
</tr>
<tr>
<td>Bnc2</td>
<td>C5.0 - decision tree</td>
<td>81.80</td>
<td>71.63</td>
<td>162 leaves</td>
</tr>
<tr>
<td></td>
<td>C5.0 - rules</td>
<td>78.70</td>
<td>73.43</td>
<td>48 propositional rules</td>
</tr>
<tr>
<td></td>
<td>Neurolinear</td>
<td>76.05</td>
<td>75.31</td>
<td>2 oblique rules</td>
</tr>
<tr>
<td></td>
<td>Neurorule</td>
<td>74.27</td>
<td>74.13</td>
<td>7 propositional rules</td>
</tr>
<tr>
<td></td>
<td>Re-RX</td>
<td>76.65</td>
<td>75.26</td>
<td>67 propositional rules</td>
</tr>
</tbody>
</table>
9. Conclusions

Neural networks can be used as data mining tools if effective rule extraction algorithms are available:

- Provision of a user explanation capability.
- Neural network prediction can be explained in terms of: symbolic rules (breast cancer data set), oblique decision rules (iris data set), MofN rules (if M of the following N conditions are satisfied, then .....), piecewise linear, or hierarchical.
- Knowledge verification by experts.
- Data exploration, discovery of previously unknown relationships in data sets.
- Extracted rules preserve the high accuracy of the networks.
References