A multilayer perceptron-based medical decision support system for heart disease diagnosis

Hongmei Yan, Yingtao Jiang, Jun Zheng, Chenglin Peng, Qinghui Li

Abstract

The medical diagnosis by nature is a complex and fuzzy cognitive process, and soft computing methods, such as neural networks, have shown great potential to be applied in the development of medical decision support systems (MDSS). In this paper, a multi-layer perceptron-based decision support system is developed to support the diagnosis of heart diseases. The input layer of the system includes 40 input variables, categorized into four groups and then encoded using the proposed coding schemes. The number of nodes in the hidden layer is determined through a cascade learning process. Each of the 5 nodes in the output layer corresponds to one heart disease of interest. In the system, the missing data of a patient are handled using the substituting mean method. Furthermore, an improved back propagation algorithm is used to train the system. A total of 352 medical records collected from the patients suffering from five heart diseases have been used to train and test the system. In particular, three assessment methods, cross validation, holdout and bootstrapping, are applied to assess the generalization of the system. The results show that the proposed MLP-based decision support system can achieve very high diagnosis accuracy (>90%) and comparably small intervals (<5%), proving its usefulness in support of clinic decision process of heart diseases.

Keywords: Medical decision support system; Multilayer perceptron; Back propagation algorithm; Performance of classifier; Heart disease

1. Introduction

Statistics have consistently shown that heart disease is one of the leading causes of deaths in US and all over the world (CDC’s report). Significant life saving, however, can be achieved if an accurate diagnosis decision can be promptly made to patients suffering various types of heart diseases, after which an appropriate treatment can immediately follow. Unfortunately, accurate diagnosis of heart diseases has never been an easy task. As a matter of fact, many factors can complicate the diagnosis of heart diseases, often causing the delay of a correct diagnosis decision. For instance, the clinical symptoms, the functional and the pathologic manifestations of heart diseases are associated with many human organs other than the heart, and very often heart diseases may exhibit various syndromes. At the same time, different types of heart diseases may have similar symptoms.

To reduce the diagnosis time and improve the diagnosis accuracy, it has become more of a demanding issue to develop reliable and powerful medical decision support systems (MDSS) to support the yet and still increasingly complicated diagnosis decision process. The medical diagnosis by nature is a complex and fuzzy cognitive process, hence soft computing methods, such as neural networks, have shown great potential to be applied in the development of MDSS of heart diseases. In (Long, Naimi & Criscitello, 1992), a probability network-based heart failure program was developed to assist physicians in reasoning about patients, which produced appropriate diagnoses about 90% of the time on the training set. Azuaje et al. (Azuaje et al., 1999) employed artificial neural networks (ANN) to recognize Poincare-plot-encoded heart rate variability patterns related to the risk of the coronary heart disease. Tkacz et al. (Tkacz & Kostka, 2000) demonstrates how
wavelet neural networks (WNN) can be applied for disease classification useful to diagnose coronary artery disease at different levels. For the diagnosis of congenital heart diseases, Reategui et al. (Reategui et al., 1997) proposed a model by integrating case-based reasoning with neural networks. In (Tsai & Watanabe, 1998), fuzzy reasoning optimized by genetic algorithm was used for the classification of myocardial heart disease.

All the above studies, with different types of soft computing methods being applied, deal with only one specific type of heart disease individually. In fact, of all the known heart diseases, hypertension, coronary heart disease, rheumatic valvular heart disease, chronic cor pulmonale, and congenital heart disease have been identified as the five most common ones (http://www.ifcc.org/ejifcc/vol14-no2/140206200308n.htm; Fuster et al., 2001). No study, however, has yet attempted to consider a system to differentiate the diagnosis of all these five major heart diseases simultaneously, which is essential to assist physicians for their diagnosis.

Multilayer perceptron (MLP) is one of the most popular neural network models due to its clear architecture and comparably simple algorithm. In this paper, we demonstrate that such a complex system to support heart disease diagnosis can be developed following a computational model based on a multilayer perceptron network consisting of three layers: (i) one input layer, (ii) one hidden layer, and (iii) one output layer. The input layer of the system takes in 40 variables, extracted from an experimental study on a large number of patient cases. The number of nodes in the hidden layer is determined by a cascade learning process. The output layer has 5 nodes corresponding to the heart diseases of interest. The whole system is trained using an improved back-propagation (BP) algorithm. Moreover, we have applied cross validation, holdout and bootstrapping methods to assess the generalization of the system. The experimental results indicate that our proposed system has a strong capability to classify the five heart diseases with high accuracy (＞90%) and comparable small intervals (＜5%), proving its usefulness in support of clinic diagnosis decision process. As pointed out in (David & Vivian, 2000), one has to recognize that the purpose of our system, like many other MDSS systems currently employed or under development, is to augment, not to replace the human diagnosticians in the complex process of medical diagnosis.

In what follows, we first review the architecture of MLP network and the basic BP algorithm in Section 2. The improvement of the learning/training strategy is also discussed in that section. In Section 3, the methods of building the decision support system based on an MLP, are presented. The experimental results are reported in Section 4. Finally, the conclusions are summarized in Section 5.

2. The MLP architecture and the BP algorithm

In this section, the theoretical background of the neural network architecture and the learning algorithm pertaining to our study are reviewed. Improvements to these algorithms are also discussed.

2.1. The multilayer perceptron (MLP)

MLP is one of the most frequently used neural network architectures in MDSS (Bishop, 1995; Hand, 1997; Ripley, 1996), and it belongs to the class of supervised neural networks. The multilayer perceptron consists of a network of nodes (processing elements) arranged in layers. A typical MLP network consists of three or more layers of processing nodes: an input layer that receives external inputs, one or more hidden layers, and an output layer which produces the classification results (Fig. 1). Note that unlike other layers, no computation is involved in the input layer. The principle of the network is that when data are presented at the input layer, the network nodes perform calculations in the successive layers until an output value is obtained at each of the output nodes. This output signal should be able to indicate the appropriate class for the input data. That is, one can expect to have a high output value on the correct class node and low output values on all the rest.

A node in MLP can be modeled as an artificial neuron (Fig. 2), which computes the weighted sum of the inputs at the presence of the bias, and passes this sum through the activation function. The whole process is defined as follows:

\[ y_j = \sum_{i=1}^{p} w_{ji} x_i + \theta_j \]

(1)

where \( y_j \) is the linear combination of inputs \( x_1, x_2, ..., x_p \), \( \theta_j \) is the bias, \( w_{ji} \) is the connection weight between the input \( x_i \)

![Fig. 1. Architecture of a multilayer perceptron network.](image-url)
and the neuron \( j \), and \( f_j(\cdot) \) is the activation function of the \( j \)th neuron, and \( y_j \) is the output.

The sigmoid function is a common choice of the activation function, as defined in Eq. (2).

\[
f(a) = \frac{1}{1 + e^{-a}}
\]  

(2)

The bias term \( \theta_j \) contributes to the left or right shift of the sigmoid activation function, depending on whether \( \theta_j \) takes a positive or negative value.

2.2. Back propagation (BP) learning algorithm

Once the architecture of MLP has been determined, the connection weights of the network have to be computed through a training procedure based on the training patterns and the desired output. BP is one of the simplest and most general methods for the supervised training of MLP (Duda et al., 2001).

The basic BP algorithm (Bishop, 1995; Duda et al., 2001) works as follows:

1. Initialize all the connection weights \( W \) with small random values from a pseudorandom sequence generator.
2. Repeat until convergence (either when the error \( E \) is below a preset value or until the gradient \( \partial E(t)/\partial W \) is smaller than a preset value).
   2.1 Compute the update using
   \[
   \Delta W(t) = -\eta \frac{\partial E(t)}{\partial W}
   \]
   2.2 Update the weights with \( W(t+1) = W(t) + \Delta W(t) \)
   2.3 Compute the error \( E(t+1) \).

where \( t \) is the iteration number, \( W \) is the connection weight, and \( \eta \) is the learning rate. The error \( E \) can be chosen as the mean square error (MSE) function between the actual output \( y_j \) and the desired output \( d_j \):

\[
E = \frac{1}{2} \sum_{j=1}^{n} (d_j - y_j)^2
\]  

(3)

There are two common training strategies: the incremental training strategy (Shigetoshi et al., 1995) and the batch training strategy (Jang et al., 1997). Usually, an incremental strategy is more efficient and also faster for systems with large training samples, as random disturbances can be induced to help the system escape from a local minimum point.

The BP algorithm described above has some shortcomings. If the learning rate is set small enough to minimize the total error, the learning process will be slowed down. On the other hand, a larger learning rate may speed up learning process at the risk of potential oscillation. Another problem is that, partial minimal points or stable stages on error surface are often encountered during the learning process (Baba, 1989).

Using a momentum term is the simplest method to avoid oscillation problems during the search for the minimum value on the error surface. The weight update in BP algorithm with a momentum term \( \alpha \) is defined as follows:

\[
\Delta W(t) = -\eta \frac{\partial E(t)}{\partial W} + \alpha \Delta W(t-1)
\]  

(4)

where \( 0 < \alpha < 1 \).

The adaptive learning rate can also be adopted to speed up the convergence of the algorithm. For batch training strategy, the learning rate can be adjusted as follows

\[
\eta(t) = \begin{cases} 
\beta \eta(t-1) & \text{if } E(t) < E(t-1) \\
\theta \eta(t-1) & \text{if } E(t) > kE(t-1) \\
\eta(t-1) & \text{otherwise}
\end{cases}
\]  

(5)

where \( \eta(t) \) is the learning rate at the \( t \)th iteration, and \( \beta, \theta, \) and \( k \) are chosen as such that \( \beta > 1, 0 < \theta < 1, \) and \( k > 1 \). While for the incremental training strategy, learning rate can be updated using

\[
\eta(t) = \eta_0 + \lambda E(t-1)
\]  

(6)

where \( \eta_0 \) is a preset learning rate, and \( \lambda > 0 \).

The learning algorithm with forgetting mechanics is an algorithm that can ‘forget’ unused connections (Takeshi, 2001). With this forgetting mechanism, the weights that are not reinforced by learning will disappear. The obtained network, thus, has a skeletal structure that reflects the regularity contained in the data, useful to improve the convergence and the network accuracy. In general, the updating of connection weights with forgetting mechanics term is given by:

\[
\Delta W'(t) = \Delta W(t) - \varepsilon \text{sgn}(W(t))
\]  

(7)

where \( \varepsilon \) is the amount for the forgetting, and \( \text{sgn}(x) \) is the sign function (i.e., \( \text{sgn}(x) = 1 \) if \( x > 0 \), \( -1 \) if \( x < 0 \), and \( 0 \) if \( x = 0 \)). The absolute value of connection weight is set to decrease by \( \varepsilon \) due to the second term on the right-hand side of Eq. (7).

In practice, some optimization algorithms are often used to improve the network convergence (Gill et al., 1981), such as the steepest descent method, the Newton method,

![Fig. 2. One node of MLP: an artificial neuron.](image-url)
the Quasi–Newton method, and the conjugate gradients method. In this study, the conjugate gradients method is adopted, as it has a low computation cost and exhibits good results (Polak, 1971). The connection weights thus can be expressed by:

\[
W(t + 1) = W(t) + \eta(t)d(t)
\]  

(8)

\[
d(t) = -\nabla E[W(t)] + \beta(t)d(t-1)
\]  

(9)

\[
d(0) = -\nabla E[W(0)]
\]  

(10)

where \( \nabla E \) is the gradient, \( d(t) \) is conjugate gradient, \( \eta(t) \) is the step wide, \( \beta(t) \) is determined given by Polak–Ribiere function shown in Eq. (11).

\[
\beta(t) = \frac{[\nabla E(W(t)) - \nabla E(W(t-1))]^T \nabla E[W(t)]}{\nabla E[W(t-1)]^T \nabla E[W(t-1)]}
\]  

(11)

3. System architecture

The MLP used in this study consists of three layers including an input layer, a hidden layer and an output layer. In this section, we will present the details regarding each of these layers.

3.1. Input layer and signal encoding

When a patient reports to a physician, a large number of possibly relevant inputs must be considered during the diagnosis. A physician reaches a correct diagnosis or treatment decision based upon observations, the patient’s answers to questions, and physical examinations or lab results. A MDSS system simulates the diagnostic process by transforming the patient information into mathematical variables. These variables then can be processed using certain reasoning mechanisms.

In this study, 40 input variables (Fig. 3) essential to the diagnosis of the heart diseases of interest are extracted from the heart diseases database. These variables can be divided into four categories: (i) the basic information of a patient, including the age and the sex, (ii) the symptoms (14 factors in total), (iii) the inducement and the history (5 factors in total), and (iv) the physical examinations and lab results (19 factors in total). Here, each input node corresponds to one input variable, and thus a total of 40 input nodes have been perceived in the input layer.

The 40 input variables (categorized into 4 groups) are encoded using the following schemes:

(i) Numerical variables such as age and blood pressure are normalized on to the interval \([0, 1]\). For instance, the patients’ ages may span from 0 to 100 years old, and thereby the age of a 56-year-old patient can be normalized to the value of \(56/100 = 0.56\).

(ii) Variables with two independent attributes, such as the sex and the neck venous return, are encoded with binary values \((0, 1)\). For instance, \(1\) represents male and \(0\) female; \(1\) is adopted when the neck venous return is positive, while \(0\) is used when it is negative.

(iii) Variables with three independent attributes, such as the factors of inducement, diastolic murmur, and

3.2. System architecture

In this section, we will present the details regarding each of these layers.
the Electrocardiogram (ECG) recordings, are encoded using the ternary values (−1, 0, 1). For example, factors of inducement will take the value of (i) −1, if the symptoms appear or worsen, when the patient is working, getting excited, or doing sports, etc; (ii) 0, if symptoms appear or worsen after hard work or intensive physical exercise; and (iii) 1, if it outbreaks chronically for a long time.

(iv) The rest variables are encoded using the three-value ordinal scales, with −1 representing the absence of the attribute, 1 representing the largest presence, and 0.5 representing intermediate level of the variables. For instance, headache can be encoded as 1 if a patient feels an acute headache, and −1 if he/she doesn’t feel headache at all. In between, the situation is coded as 0.5.

The complete encoding values of all 40 variables are listed in Fig. 3. Note that the encoded values of the input variables are mainly for numerically setting the connections of neurons in the network.

3.2. Hidden layer

Kavzoglu (Kavzoglu, 2001) has shown that the number of neurons in a hidden layer will significantly influence the network’s ability to generalize from the training data to the unknown examples. A network with small hidden nodes cannot fully identify the structures present in the training data (known as underfitting), while a network with large hidden nodes may determine decision boundaries in vector space that are unduly influenced by the specific properties of the training data. The latter phenomenon is known as overfitting. Some experimental formulas have been presented to decide the number of neurons needed in a hidden layer (Kavzoglu, 2001), but they can only provide very coarse estimation values. In fact, accuracy and convergence speed are the two main parameters of a network, and we thus determine the number of the hidden nodes through a cascade learning process to seek a balance between these two.

In the cascade learning process, a four-fold cross validation method (more details are provided in Section 4) is adopted. The whole process works as follows:

Step 1. Set the initial value of the number of the hidden nodes;
Step 2. Continue this step until a preset number is reached
   Step 2.1. Train the network with concurrent configuration of the hidden nodes, record the learning time for each group, and then compute the average time;
   Step 2.2. Test the accuracy with the corresponding test dataset, and compute the overall accuracy;
   Step 2.3. Increment the number of hidden nodes by 1.

Fig. 4 shows the result of the accuracy rate (curve 1) and the learning time (curve 2) with respect to the numbers of hidden nodes. Here one can see that, the network can achieve a high accuracy rate with a comparable learning time when the number of hidden nodes is between 10 and 15. When the number of hidden nodes reaches 15, the network achieves the highest accuracy ratio of 91.5%. As a result, in our system, the hidden layer is determined to consist of 15 nodes.

3.3. Output layer

The output layer comprises of 5 nodes and each node corresponds to one heart disease of interest. The architecture of the overall decision support system is illustrated in Fig. 5.

3.4. Summary about the system

In this section, we have shown that the proposed decision support system is a three-layer MLP with 40 input variables,
15 hidden nodes and 5 outputs. A back propagation algorithm augmented with the momentum term, the adaptive learning rate, the forgetting mechanics, and an optimized algorithm based on the conjugate gradients method is adopted. This improved learning algorithm works as follows:

Step 1. Initialize all the connection weights $W$ with small random values.

Step 2. Input a sample, and repeat until the convergence. Compute the update: $\Delta W(t) = \eta(t)d(t) + \alpha \Delta W(t-1) - \epsilon \text{sgn}(W(t))$

Update the weights: $W(t+1) = W(t) + \Delta W(t)$

Compute the error: $E(t+1)$

Step 3. Go back to Step 2 until all the training samples have been exhausted.

The training parameters adopted in the system are presented in Table 1.

<table>
<thead>
<tr>
<th>The training parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Momentum constant</td>
<td>0.04</td>
</tr>
<tr>
<td>Preset learning rate</td>
<td>0.08</td>
</tr>
<tr>
<td>Learning rate adjusting constant</td>
<td>0.1</td>
</tr>
<tr>
<td>Forgetting constant</td>
<td>0.0003</td>
</tr>
<tr>
<td>Error precision target</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 1

15 hidden nodes and 5 outputs. A back propagation algorithm augmented with the momentum term, the adaptive learning rate, the forgetting mechanics, and an optimized algorithm based on the conjugate gradients method is adopted. This improved learning algorithm works as follows:

Step 1. Initialize all the connection weights $W$ with small random values.

Step 2. Input a sample, and repeat until the convergence. Compute the update: $\Delta W(t) = \eta(t)d(t) + \alpha \Delta W(t-1) - \epsilon \text{sgn}(W(t))$

Update the weights: $W(t+1) = W(t) + \Delta W(t)$

Compute the error: $E(t+1)$

Step 3. Go back to Step 2 until all the training samples have been exhausted.

The training parameters adopted in the system are presented in Table 1.

Table 1

<table>
<thead>
<tr>
<th>The training parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Momentum constant</td>
<td>0.04</td>
</tr>
<tr>
<td>Preset learning rate</td>
<td>0.08</td>
</tr>
<tr>
<td>Learning rate adjusting constant</td>
<td>0.1</td>
</tr>
<tr>
<td>Forgetting constant</td>
<td>0.0003</td>
</tr>
<tr>
<td>Error precision target</td>
<td>0.05</td>
</tr>
</tbody>
</table>

4. Experimental results and performance assessment

The heart disease database used for testing and tuning the system in this study consists of 352 cases gathered from the Southwest Hospital and the Dajiang Hospital, both located in Chongqing, P. R. China. We have trained and tested the system using these medical records and the results are reported in this section.

4.1. Dataset preparation and performance assessment strategy

Table 2 lists the distribution of the five classes of heart diseases in the database. These medical cases were obtained without prior screening.

Table 2

<table>
<thead>
<tr>
<th>Heart diseases diagnosis</th>
<th>Number of patients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypertension</td>
<td>86</td>
</tr>
<tr>
<td>Coronary heart disease</td>
<td>82</td>
</tr>
<tr>
<td>Rheumatic valvular heart disease</td>
<td>71</td>
</tr>
<tr>
<td>Chronic cor pulmonale</td>
<td>60</td>
</tr>
<tr>
<td>Congenital heart disease</td>
<td>53</td>
</tr>
<tr>
<td>Total</td>
<td>352</td>
</tr>
</tbody>
</table>

4.2. Cross-validation test

4.2.1. Method description

The cross-validation estimate of accuracy, sometimes referred as rotation estimation, is determined by the overall number of correct classifications divided by the total number of instances in the dataset (Kohavi, 1995):

$$Acc_{cv} = \frac{1}{n} \sum_{(x_i,y_i) \in D} \delta(I(D_i, x_i), y_i)$$

where $n$ is the size of the dataset $D$, $x_i$ is the instance of $D$, $y_i$ is the label of $x_i$, and $D_i$ is the possible label of $x_i$ by the classifier. Here,

$$\delta(i,j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

In this study, we have also applied a four-fold cross-validation method for the performance assessment. The data listed in Table 2 are thus equally divided into four subsets.
and four different sets of experiments have been performed:

(i) Training: $F_1 + F_2 + F_3$; Testing: $F_4$
(ii) Training: $F_1 + F_2 + F_4$; Testing: $F_3$
(iii) Training: $F_1 + F_3 + F_4$; Testing: $F_2$
(iv) Training: $F_2 + F_3 + F_4$; Testing: $F_1$

4.2.2. Results

The cross-validation estimated accuracy of each test run, the interval of estimated accuracy and the mean accuracy are listed in Table 4. Here the first column CV1 refers to the first test based on the cross validation method.

The experimental results of the four test sets have also been presented as a confusion matrix (Table 5). Generally, a confusion matrix contains information about the actual and the predicted classes. In the confusion matrix, the columns represent the test data, while the rows represent the labels assigned by the classifier. Several indices of classification accuracy can be derived from the confusion matrix.

The cross-validation classification accuracy thus far can be determined as

$$\frac{75 + 66 + 77 + 56 + 48}{352} = \frac{322}{352} = 91.5\% \quad (14)$$

From the confusion matrix shown in Table 5, we can readily calculate the procedure accuracy and the user

Table 3
Number of cases with missing data values and the mean value of each diagnostic variable

<table>
<thead>
<tr>
<th>Diagnostic variables</th>
<th>Number of cases with missing data</th>
<th>Mean of each variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Information Age</td>
<td>0</td>
<td>0.47</td>
</tr>
<tr>
<td>Sex</td>
<td>0</td>
<td>0.43</td>
</tr>
<tr>
<td>Symptoms Cough</td>
<td>0</td>
<td>0.18</td>
</tr>
<tr>
<td>Emptysis/expectoration</td>
<td>0</td>
<td>-0.59</td>
</tr>
<tr>
<td>Dyspnea</td>
<td>0</td>
<td>-0.36</td>
</tr>
<tr>
<td>Gasping/panting</td>
<td>0</td>
<td>0.18</td>
</tr>
<tr>
<td>Nausea/vomiting</td>
<td>0</td>
<td>-0.79</td>
</tr>
<tr>
<td>Dizziness</td>
<td>0</td>
<td>-0.19</td>
</tr>
<tr>
<td>Headache</td>
<td>0</td>
<td>-0.55</td>
</tr>
<tr>
<td>Fever</td>
<td>0</td>
<td>-0.90</td>
</tr>
<tr>
<td>Cyanosis</td>
<td>5</td>
<td>-0.79</td>
</tr>
<tr>
<td>Palpitations</td>
<td>0</td>
<td>0.17</td>
</tr>
<tr>
<td>Weakness or fatigue</td>
<td>2</td>
<td>-0.6</td>
</tr>
<tr>
<td>Discomfort, pressure, heaviness in the chest</td>
<td>0</td>
<td>-0.37</td>
</tr>
<tr>
<td>Chest pain (precordial region, posterior sternal)</td>
<td>0</td>
<td>-0.50</td>
</tr>
<tr>
<td>Edema of lower limb</td>
<td>12</td>
<td>-0.57</td>
</tr>
<tr>
<td>Known factors of inducement</td>
<td>0</td>
<td>-0.05</td>
</tr>
<tr>
<td>Upper respiratory infection</td>
<td>146</td>
<td>-0.69</td>
</tr>
<tr>
<td>Chronic pulmonary disease history</td>
<td>0</td>
<td>-0.67</td>
</tr>
<tr>
<td>Hypertension family history</td>
<td>43</td>
<td>-0.67</td>
</tr>
<tr>
<td>Coronary heart disease family history</td>
<td>56</td>
<td>-0.65</td>
</tr>
<tr>
<td>Inducement and history Blood pressure</td>
<td>0</td>
<td>0.45</td>
</tr>
<tr>
<td>Cervical venous engorgement</td>
<td>0</td>
<td>-0.66</td>
</tr>
<tr>
<td>Barrel chest</td>
<td>0</td>
<td>-0.65</td>
</tr>
<tr>
<td>Sound of breath in pulmonary</td>
<td>0</td>
<td>-0.43</td>
</tr>
<tr>
<td>Moist rales</td>
<td>0</td>
<td>-0.71</td>
</tr>
<tr>
<td>Boundary of heart sonant</td>
<td>0</td>
<td>-0.20</td>
</tr>
<tr>
<td>Systolic murmur</td>
<td>0</td>
<td>-0.13</td>
</tr>
<tr>
<td>Diastolic murmur</td>
<td>0</td>
<td>0.43</td>
</tr>
<tr>
<td>Systolic thrill</td>
<td>41</td>
<td>-0.59</td>
</tr>
<tr>
<td>Second heart sound</td>
<td>87</td>
<td>-0.57</td>
</tr>
<tr>
<td>Liver and kidney tenderness</td>
<td>0</td>
<td>-0.51</td>
</tr>
<tr>
<td>Hyperresonant note</td>
<td>0</td>
<td>-0.65</td>
</tr>
<tr>
<td>Cardiac rhythm in good order or not</td>
<td>0</td>
<td>-0.46</td>
</tr>
<tr>
<td>Eminence in precordial region</td>
<td>41</td>
<td>-0.66</td>
</tr>
<tr>
<td>Neck venous return</td>
<td>0</td>
<td>0.83</td>
</tr>
<tr>
<td>Pulmonic P-wave</td>
<td>74</td>
<td>-0.79</td>
</tr>
<tr>
<td>Cardiac enlargement</td>
<td>0</td>
<td>-0.07</td>
</tr>
<tr>
<td>Electrocardiogram</td>
<td>15</td>
<td>0.17</td>
</tr>
<tr>
<td>ST-T alteration</td>
<td>53</td>
<td>-0.55</td>
</tr>
</tbody>
</table>
The accuracy, which shows the proportion of the diseases in the test dataset that have been correctly recognized by the classifier, is presented as follows:

<table>
<thead>
<tr>
<th>Disease</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coronary heart disease</td>
<td>75/83 = 90.4%</td>
</tr>
<tr>
<td>Rheumatic valvular heart disease</td>
<td>66/73 = 90.4%</td>
</tr>
<tr>
<td>Hypertension</td>
<td>77/85 = 90.6%</td>
</tr>
<tr>
<td>Chronic cor pulmonale</td>
<td>56/59 = 94.9%</td>
</tr>
<tr>
<td>Congenital heart disease</td>
<td>48/52 = 92.3%</td>
</tr>
</tbody>
</table>

The user accuracy, which measures the proportion of diseases identified by the classifier as belonging to class $i$ that agrees with the test data, is given as follows:

<table>
<thead>
<tr>
<th>Disease</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coronary heart disease</td>
<td>75/82 = 91.5%</td>
</tr>
<tr>
<td>Rheumatic valvular heart disease</td>
<td>66/71 = 93.0%</td>
</tr>
<tr>
<td>Hypertension</td>
<td>77/86 = 89.5%</td>
</tr>
<tr>
<td>Chronic cor pulmonale</td>
<td>56/60 = 93.3%</td>
</tr>
<tr>
<td>Congenital heart disease</td>
<td>48/53 = 90.6%</td>
</tr>
</tbody>
</table>

4.4. Bootstrapping test

Bootstrapping method, similar to cross-validation, is also a method for estimating generalization error based on ‘resampling’ (Embrechta et al., 2001). Given a dataset of size $n$, a bootstrapping sample is created by sampling $n$ instances uniformly from the data set (with replacement). For estimating generalization error in classification problems, the 0.632 bootstrapping (Jain et al., 1987) is one of the currently favored methods, as it can perform well even when there is severe overfitting. The 0.632 bootstrapping estimate $\text{Acc}_{\text{boot}}$ is defined as (Efron & Tibshirani, 1993; Jain et al., 1987):

$$\text{Acc}_{\text{boot}} = \frac{1}{b} \sum_{i=1}^{b} \delta (I(D_i, x_i), y_i)$$

where $h$ is the size of the holdout set $D_h$.

In our study, we randomly designate $3/4$ of the data listed in Table 2 to form the training set and the remaining $1/4$ go to the holdout set. In total, four holdout runs have been performed. The holdout estimate accuracy of each run, the interval of estimated accuracy and the mean accuracy are listed in Table 6.

The holdout method tends to be a pessimistic estimator because only a portion of the data is given to the inducer for training. The more instances leave for the test set, the higher the bias of estimate; however, fewer test set instances means that the confidence interval for accuracy will be wider (Kohari, 1995).

4.5. Holdout test

The holdout method, sometimes referred as test sample estimation, partitions all the data into two disjoint subsets called the training set and the holdout set (test set). The holdout estimated accuracy is defined as (Penrod and Wagner, 1979)

$$\text{Acc}_h = \frac{1}{h} \sum_{(x_i, y_i) \in D_h} \delta (I(D_i, x_i), y_i)$$

where $h$ is the size of the holdout set $D_h$.

In our study, we randomly designate $3/4$ of the data listed in Table 2 to form the training set and the remaining $1/4$ go to the holdout set. In total, four holdout runs have been performed. The holdout estimate accuracy of each run, the interval of estimated accuracy and the mean accuracy are listed in Table 6.

4.4. Bootstrapping test

Bootstrapping method, similar to cross-validation, is also a method for estimating generalization error based on ‘resampling’ (Embrechta et al., 2001). Given a dataset of size $n$, a bootstrapping sample is created by sampling $n$ instances uniformly from the data set (with replacement). For estimating generalization error in classification problems, the 0.632 bootstrapping (Jain et al., 1987) is one of the currently favored methods, as it can perform well even when there is severe overfitting. The 0.632 bootstrapping estimate $\text{Acc}_{\text{boot}}$ is defined as (Efron & Tibshirani, 1993; Jain et al., 1987):

$$\text{Acc}_{\text{boot}} = \frac{1}{b} \sum_{i=1}^{b} \delta (I(D_i, x_i), y_i)$$

where $h$ is the size of the holdout set $D_h$.

In our study, we randomly designate $3/4$ of the data listed in Table 2 to form the training set and the remaining $1/4$ go to the holdout set. In total, four holdout runs have been performed. The holdout estimate accuracy of each run, the interval of estimated accuracy and the mean accuracy are listed in Table 6.

The holdout method tends to be a pessimistic estimator because only a portion of the data is given to the inducer for training. The more instances leave for the test set, the higher the bias of estimate; however, fewer test set instances means that the confidence interval for accuracy will be wider (Kohari, 1995).
Table 6  
Accuracy result: holdout  
<table>
<thead>
<tr>
<th>Holdout</th>
<th>Holdout</th>
<th>Holdout</th>
<th>Holdout</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>[89.8, 93.2]</td>
</tr>
<tr>
<td>Acch (%)</td>
<td>92.0</td>
<td>93.2</td>
<td>92.0</td>
<td>93.2</td>
</tr>
<tr>
<td>Mean Acch (%)</td>
<td>90.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7  
Accuracy result: bootstrapping  
<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{\xi}_i$</td>
<td>0.921</td>
<td>[0.902, 0.946]</td>
</tr>
<tr>
<td>$\hat{\xi}_i$</td>
<td>0.906</td>
<td>[0.893, 0.927]</td>
</tr>
<tr>
<td>Accboot (%)</td>
<td>91.1</td>
<td></td>
</tr>
</tbody>
</table>

where b is the number of bootstrap samples, $\bar{\xi}_i$ is the accuracy estimate for bootstrap sample $i$, and $\hat{\xi}_i$ is the resubstitution accuracy estimate or apparent error rate.

In this study, we have generated 5 bootstrapping samples from the full samples summarized in Table 2. The network then has been trained with each bootstrapping sample and the accuracy estimate has been determined from the corresponding test set. The results are provided in Table 7.

4.5. Summary

Three different evaluation methods have been adopted to assess the performance of the proposed MLP-based decision support system. From the results reported in Tables 4–7, one can see that the system has a strong capability to accurately recognize all the five heart diseases (>90%) with comparable small intervals (5%).

5. Conclusion

In this paper, we have presented a medical decision support system based on the MLP neural network architecture for heart disease diagnosis. In particular, we have identified the 40 input variables critical to the diagnosis of the heart diseases of interest and encoded them accordingly. The system is trained by employing an improved BP algorithm. A heart diseases database consisting of 352 cases has been used in this study. Three assessment methods, cross validation, holdout and bootstrapping, have been applied to assess the generalization of the system. The results show that the proposed system can achieve very high diagnosis accuracy (>90%) and comparably small intervals (<5%), proving its usefulness in support of clinic diagnosis decision of heart diseases.

Acknowledgements

This work was supported by NSFC 30400105. The authors would like to thank Southwest Hospital and Dajiang Hospital, Chongqing, China, for their support in the heart disease database and valuable suggestions.

References

http://www.ifcc.org/eijifcc/vol14no2/140206200308n.htm


