Knowl Inf Syst (2012) 30:135–154
DOI 10.1007/s10115-011-0392-6

REGULAR PAPER

A general insight into the effect of neuron structure on classification

Hadi Sadoghi Yazdi · Alireza Rowhanimanesh · Hamidreza Modares

Received: 29 April 2010 / Revised: 12 February 2011 / Accepted: 19 March 2011 /
Published online: 31 March 2011

Abstract This paper gives a general insight into how the neuron structure in a multilayer perceptron (MLP) can affect the ability of neurons to deal with classification. Most of the common neuron structures are based on monotonic activation functions and linear input mappings. In comparison, the proposed neuron structure utilizes a nonmonotonic activation function and/or a nonlinear input mapping to increase the power of a neuron. An MLP of these high power neurons usually requires a less number of hidden nodes than conventional MLP for solving classification problems. The fewer number of neurons is equivalent to the smaller number of network weights that must be optimally determined by a learning algorithm. The performance of learning algorithm is usually improved by reducing the number of weights, i.e., the dimension of the search space. This usually helps the learning algorithm to escape local optimums, and also, the convergence speed of the algorithm is increased regardless of which algorithm is used for learning. Several 2-dimensional examples are provided manually to visualize how the number of neurons can be reduced by choosing an appropriate neuron structure. Moreover, to show the efficiency of the proposed scheme in solving real-world classification problems, the Iris data classification problem is solved using an MLP whose neurons are equipped by nonmonotonic activation functions, and the result is compared with two well-known monotonic activation functions.

Keywords Neuron structure · Nonmonotonic activation function · Nonlinear input mapping · Classification · Multilayer perceptron (MLP) · Iris data classification

H. Sadoghi Yazdi (✉)
Department of Computer Engineering, Ferdowsi University
of Mashhad, 91775-1111 Mashhad, Iran
e-mail: h-sadoghi@um.ac.ir

A. Rowhanimanesh · H. Modares
Department of Electrical Engineering, Ferdowsi University
of Mashhad, 91775-1111 Mashhad, Iran

1 Introduction

Neural networks have emerged as an important tool for classification. Recent research activities identified neural networks (NNs) as a promising alternative to other classification methods ([25]). Many different types of NNs are now being used for classification purposes ([25]), including, among others, multilayer perceptron (MLP) NNs (Hagan et al.; Lin et al.), radial basis function (RBF) NNs ([1,30]), recurrent NNs ([9,28]), general regression neural networks ([31]), and product unit neural networks ([5]) as a multiplicative NNs.

The design of NNs requires several decisions, including the following: (1) the number of hidden units that resides in the hidden layer, (2) the neuron structure including the neuron input mapping and type of neuron activation function that used at the hidden and output units, and (3) learning algorithm that determines what weights will be applied to the network. Also, these three factors are not independent of each other. The number of hidden units influences the speed and performance of learning algorithm. It is a natural hope to employ as fewer as possible the number of hidden neurons under the conditions that they meet the performance requirements. In fact, a small network is able to have faster learning speeds. On the other hand, the type of neuron activation functions and the way of aggregation are the important factors that influence the number of hidden units required for a satisfying performance. Also, the types of activation functions have very important influences on the network learning speed, classification correction rates, and nonlinear mapping precision ([6,26]). So, the type of activation function is the first important factor that must be considered, because it influences the two other factors.

Several neural networks with different activation functions are introduced in the literature. The most frequently used activation functions are monotonic activation functions, such as sigmoid function and hard-limiter. A very good comparison between these two activation functions is made in Daqi and Yan [4]. MLP neural networks with sigmoid activation functions usually are the most widely used NNs. The classification mechanisms of MLPs with sigmoid activation functions have been paid attention to in the literature ([2,8,10,18,19,21,32,36]). However, when the networks with sigmoid activation functions are used in pattern recognition, there are some difficulties including convergence to local minima, a slow learning speed, and a proper selection rule of the number of hidden units. Various activation functions have been introduced to apply to neural networks in order to address the above-mentioned difficulties ([7,11,12,16,17,20,22,24,27,29,33,35,37]). Among them, nonmonotonic activation functions seem to be promising. Gaussian functions and sinusoidal activation functions are the most frequently used ones. Comparative results of this type of neurons with sigmoid functions were considered in the past. In Karaköse and Akin [20], a quadratic sigmoid function, which resembles radial basis functions, is utilized, and better performance indications than sigmoid based networks are emphasized. In a study by Sopena et al. [33], a sinusoidal activation function is used, and the authors show empirically that it has better performance in comparison with sigmoid ones. Also, in Gutierrez et al. [12], Hara and Nakayamma [14] sigmoid, sinusoidal, and Gaussian type activation functions are compared, and among those, the sinusoidal one is seen to display most desirable characteristics. The authors of these papers empirically show that there are a number of reasons that the choice of nonmonotonic functions, especially sinusoidal active functions, can produce a notable improvement in the performance of a neural network in classification. But they offer no analysis for their results.

This paper focuses on the analysis of performance of nonmonotonic neuronal activation function and answer to the question why nonmonotonic activation functions create better performance than monotonic activation functions. We will show that the use of nonmonotonic activation functions sometimes reduces the complexity of network by reducing the

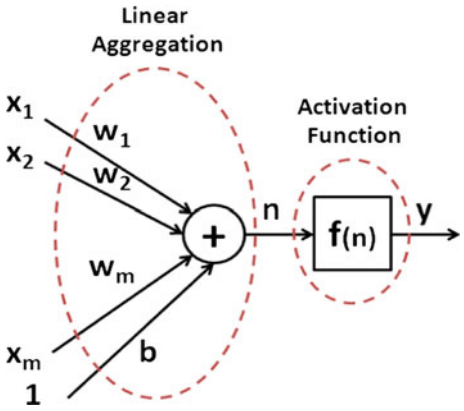


Fig. 1 Conventional neuron structure

number of hidden nodes needed. Moreover, nonlinear aggregation mapping is considered in this paper to design a new kind of neural structure that involves a more sophisticated architecture. The ability of the proposed neuron, which is called high power neuron (HPN), is shown through solving different synthetic 2-dimensional examples, which are solved manually. The motivation is to visualize what the proposed approach claims. Moreover, the effectiveness of the proposed HPN in solving the real-world classification problems is shown by solving the well-known Iris data classification problem. It should be mentioned that in this paper, we just focus on a particular class of neural networks, whose neuron structure consists of activation functions and aggregation mappings. This structure is discussed with details in Sect. 2.

The rest of the paper is organized as follows. In Sect. 2, the conventional neuron structure is described, and it is shown that how a conventional neuron solves a classification problem. In Sect. 3, the proposed general neuron structure is discussed based on nonmonotonic activation functions and nonlinear input mappings. Finally, conclusion is presented in Sect. 4.

2 Preliminaries

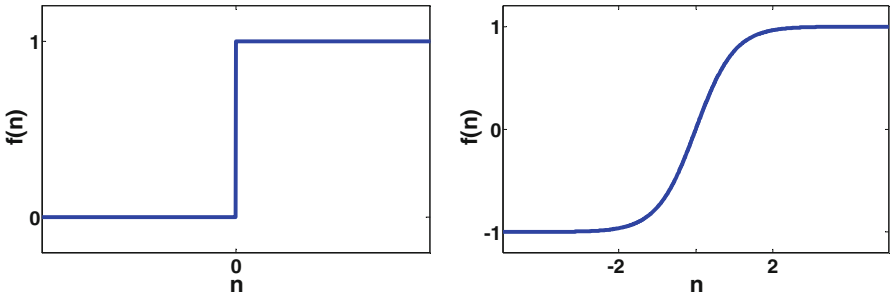
In this section, conventional neuron architecture is presented and the way that a network with combination of conventional neurons solves a classification problem is discussed. Figure 1 shows a conventional neuron structure, which is used in MLP neural networks. Generally, in MLP neural networks, a neuron can be decomposed into two fundamental parts: aggregation mapping and activation function.

Aggregation mapping is a mathematical transformation that takes the input vector and maps it into the net input. Let $X = [x_1, x_2, \ldots, x_m]^{\mathrm{T}}$ is the input sample; the net input of hidden unit is given by:

$$n = \sum_{i=0}^{m} \omega_i x_i = WX + b \tag{1}$$

where $W = [W_1, W_2, \ldots, W_m]^{\mathrm{T}}$ is a constant weight matrix, $\omega_0 = b$ is called the bias, and $x_0 = 1$. n is called the hidden basis function. After the aggregation mapping, the neuron produces an output using an activation function. This activation function transforms the value produced by the aggregation mapping to the neuron output, as formulated in Eq. (2).

$$y = f(n) = f(WX + b) \tag{2}$$



$$f_{hard\,\lim}(n)=\begin{cases}0 & n<0\\ 1 & n\geq 0\end{cases} \qquad f_{\tan sig}(n)=\frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$$

Fig. 2 Conventional monotonic activation functions (*Left* hard-limiter as a nonsmooth case, *Right* Sigmoid as a smooth case)

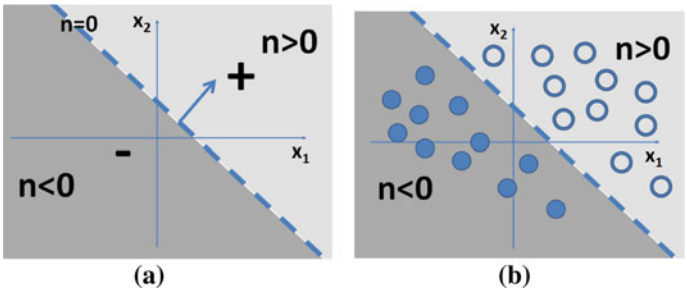


Fig. 3 Conventional neuron as a binary classifier

As mentioned in Sect. 1, in most of conventional applications, f is a monotonic function (or semi-monotonic for discontinuous functions) such as sigmoid and hard-limiter functions (Hagan et al.) as shown in Fig. 2.

Now, let us consider how a conventional neuron solves a classification problem. Equation $WX + b = 0$, which is obtained from setting basis function of hidden unit equal to zero, is a hyper plane over an m-dimensional input space. It is well known that a hidden neuron with monotonic activation functions generate a single hyperplane, which divides the input space into two regions. As an example, consider a hyperplane (line) shown in Fig. 3a over a 2-dimensional input space. As Fig. 3 indicates, the input samples are located either at the positive side of the hyperplane or at the negative side of the hyperplane. The vector of a hyperplane, W, points to the positive side. So, regarding Fig. 3a and Eq. (1), for the samples located at the positive side, we have $n > 0$, and for the samples located at the negative side, we have $n < 0$. Thus, given an input vector X a conventional neuron can easily classify it by considering the sign of basis function, n. When the net input of a pattern fell to the positive side of hyperplane, the output of neuron is positive, while patterns that fell to the negative side results in a negative output.

So, an activation function such as the ones of Fig. 2 is used to detect the sign of n. As a result, a conventional neuron can be used for binary classification problems if the given patterns can be classified by only one line (hyper plane) such as Fig. 3b.

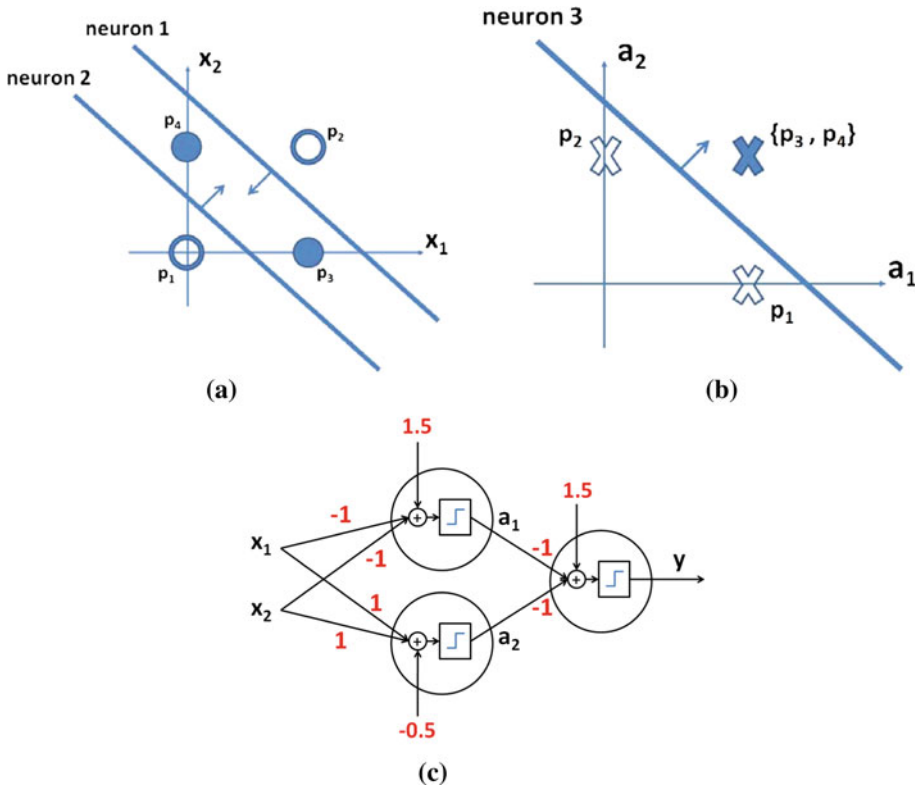


Fig. 4 Solving XOR problem using 3 conventional neurons as a 2-2-1 MLP network

The combination of several of these hyperplanes can give rise to complex classifications. For example, consider XOR problem that cannot be solved by a single conventional neuron. In other words, XOR problem is not separable by a single line, while a conventional neuron only provides a single line. Thus, more than one neuron must be used. Figure 4 shows a two-layer MLP with two neurons in the hidden layer and one neuron at the output layer. The hard-limit function as a monotonic activation function is used in this case. In fact, this is the smallest MLP neural network that can be constructed by conventional neurons for solving XOR problem. The two neurons of the hidden layer create two lines in the input space, which classify the input space into three parts. These three parts are characterized by vector $[a_1, a_2]^{\mathrm{T}}$ as $[0, 1]^{\mathrm{T}}$, $[0, 0]^{\mathrm{T}}$ and $[1, 0]^{\mathrm{T}}$. In other words, the hidden layer plays the role of a nonlinear mapping that maps $[x_1, x_2]^{\mathrm{T}}$ (input space) to $[a_1, a_2]^{\mathrm{T}}$ (new space). Now, in the new space, the set of patterns $\left\{\left([0, 1]^{\mathrm{T}}, 1\right), \left([1, 0]^{\mathrm{T}}, 1\right), \left([0, 0]^{\mathrm{T}}, 0\right)\right\}$ are linearly separable. Figure 4b indicates how the single conventional neuron (single line) of output layer can easily classify these patterns. We can see that three conventional neurons are required to solve XOR problems. This implies that 9 unknown parameters of the network must be determined through learning since each conventional neuron has 3 unknown parameters. In the next section, we discuss how a powerful neuron structure called high power neuron (HPN) is able to solve XOR problem singly.

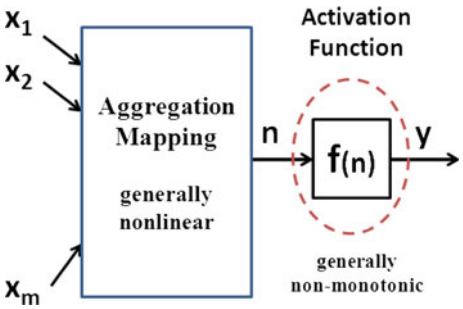


Fig. 5 HPN structure

3 The proposed high power neuron (HPN)

Figure 5 indicates HPN structure, where in contrast to conventional structure, the aggregation mapping and activation functions are allowed to be nonlinear and nonmonotonic, respectively. In the following, first, the importance of nonmonotonic activation functions and nonlinear aggregation is described, separately. Then, the combination of nonmonotonic activation function and nonlinear aggregation is described.

3.1 Nonmonotonic activation functions

Regarding Figs. 1 and 2, activation function of a conventional neuron is usually a monotonic function, such as hard-limit or sigmoid functions. As discussed in the former section, a hidden neuron with monotonic activation functions generates a single hyperplane that divides the input space into two regions discriminated by the sign of basis function of neuron, n. In conventional neuron, only this property of basis function is usually considered, while it is very important to note that for a given input sample, the value of basis function, n, also determines the distance of the given input sample from the hyperplane. Thus, how can we divide the space into more than two regions by exploit this significant property?

Suppose, we use $f_{\text{squ}}(n)$ defined by Eq. (3) instead of the monotonic functions of Fig. 2.

$$f_{\text{squ}}(n) = \begin{cases} 1 & n \in (0, a_1) \cup (a_2, a_3) \cup \left(\hat{a}_2, \hat{a}_1\right) \cup \ldots \\ 0 & n = \ldots, \hat{a}_1, 0, a_1, a_2, \ldots \\ -1 & n \in \left(\hat{a}_1, 0\right) \cup (a_1, a_2) \cup (a_3, a_4) \cup . \end{cases} \tag{3}$$

Figure 6a displays $f_{\text{squ}}(n)$. Regarding this figure, this nonmonotonic function $f_{\text{squ}}(n)$ is not only sensitive to the sign of basis function, while the value of basis function (distance from the line) is severely considered in this function. For example, $f_{\text{squ}}(n)$is $+1$ in intervals of $0 < 0 < a_1,\ a_2 < n < a_3,\ \hat{a}_2 < n < \hat{a}_1 \ldots$ and is -1 in $\hat{a}_1 < n < 0, a_1 < n < a_2$. If $f_{\text{squ}}(n)$ is used as the activation function of the neuron of Fig. 1, then using this new neuron, the input space is divided into more than two parts as shown in Fig. 6b. Regarding this figure, it seems that the HPN creates a family of parallel lines instead of a single line. This set of parallel lines is obtained from solving the following equation:

$$f_{\text{squ}}(n_0) = 0 \overset{more\ than\ one\ root}{\Rightarrow} n_0 \in \ldots, \widehat{a}_1, 0, a_1, a_2, \ldots \tag{4}$$

Since n_0 is not unique, according to $WX + b = n_0$, different parallel lines are created only by using a nonmonotonic activation function instead of monotonic ones. It should be noted that various nonmonotonic activation functions with interesting properties can be defined. The discontinuous behavior of f_{squ} versus n is similar to hard-limiter. In contrast,

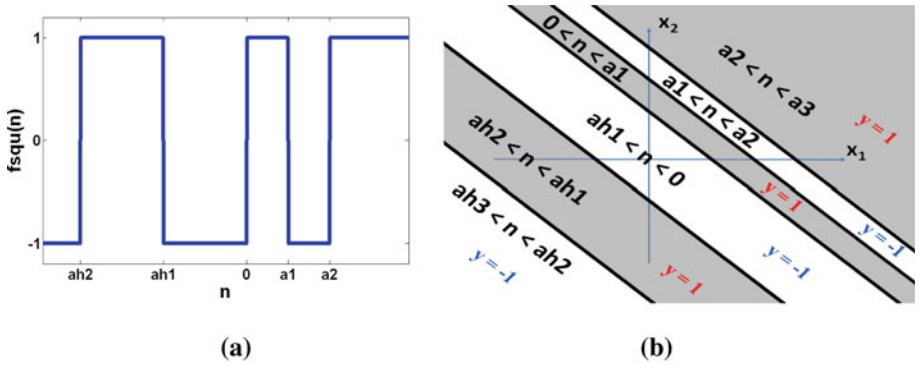


Fig. 6 $f_{\text{squ}}(n)$ as a nonmonotonic activation function defined by Eq. (3)

various nonmonotonic activation functions can be used such as sinusoidal functions (periodic nonmonotonic activation functions), Gaussian functions, polynomial functions, which have continuous behavior versus n as similar as sigmoidal activation functions.

Now, we return to XOR problem and solve it by a single neuron with nonmonotonic activation function. Consider a single neuron of Fig. 1 when the activation function is a special case of $f_{\text{squ}}(n)$ represented by $f_{\text{pulse}}(n)$ and defined as follows:

$$f_{\text{pulse}}(n) = \begin{cases} 1 & n < -1 \\ 0 & -1 \leq n \leq 1 \\ -1 & n > 1 \ldots \end{cases} \tag{5}$$

So, using the above discussions, two parallel lines can be achieved as later:

$$w_1 x_1 + w_2 x_2 = -1 \quad \text{and} \quad w_1 x_1 + w_2 x_2 = 1 \tag{6}$$

By choosing $W = [2, 2]$ and $b = -2$ and regarding Fig. 7, this neuron can singly solve XOR problem. Now, we can see that XOR problem can be solved by only one neuron instead of three conventional neurons. Note that many other functions can be used instead of $f_{\text{pulse}}(n)$ like periodic version of $f_{\text{squ}}(n)$ represented by $f_{\text{pulse}}(n)$ and defined as later:

$$f_{\text{psqu}}(n) = \sum_{k=-\infty}^{+\infty} f_{\text{pulse}}(n - 3k) \tag{7}$$

In the following, we consider three further simulation examples that are much more complex than XOR problem.

3.1.1 Example 1: two rings of points classification problem

Consider a data set composed of two rings of points as in Fig. 8. An MLP classifier must create separation hyperplanes among two classes to classify them. This example shows the importance of choosing the type of activation functions. First, we discuss how a neural network with hard-limit activation functions, as a monotonic activation function can solve this problem. Hence, assume that we use hard-limit activation functions in both hidden and output nodes. For the neurons with hard-limiter activation function, the hyperplanes given by the hidden basis functions with values 0 ($n = 0$) are the decision boundaries, and the decision regions have no change when all of the weights and biases change in same proportion ([4]).

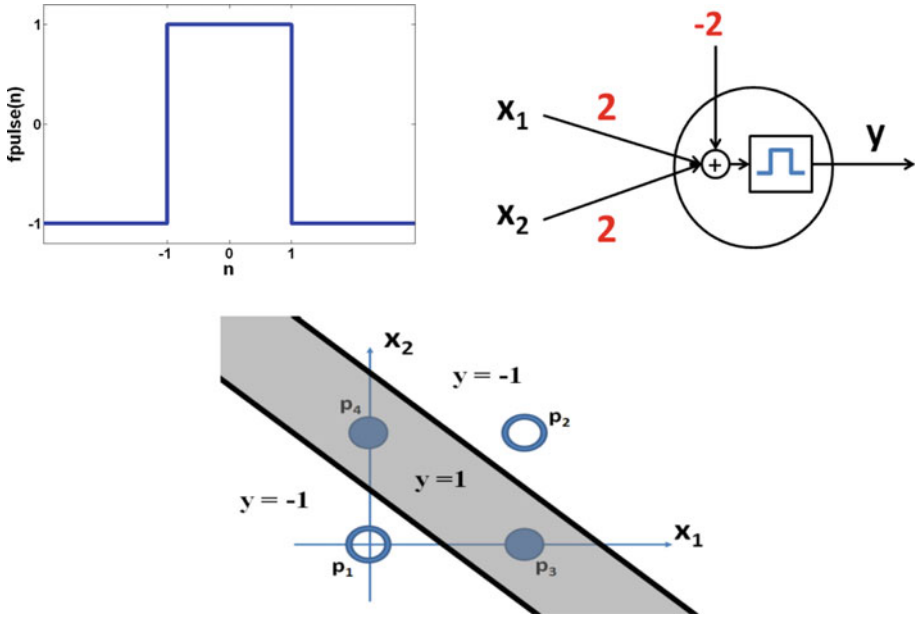


Fig. 7 Solving XOR problem by only one neuron with the nonmonotonic activation function of $f_{\text{pulse}}(n)$

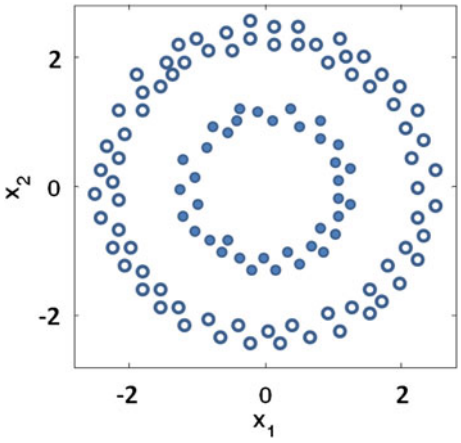


Fig. 8 Two rings of points classification problem

So, it may seem that we can separate two classes by using three hidden neurons to create three separation hyperplanes in the input space among two classes. We aim to recognize whether this problem can be solved by a 2-3-1 MLP with hard-limit activation functions or not. Figure 9 shows the best possible response obtained from this MLP.

It is obvious that the three hyperplanes shown in Fig. 9, which are the decision boundaries created by this activation function, cannot correctly classify these two classes. If we use a 2-4-1 perceptron, instead, this problem still cannot be solved. As we expect, the four hidden neuron create four decision hyperplanes in the input space that can separate two classes in the input space, but the two classes are still nonlinearly separable in the spaces spanned by the hidden responses. In fact, the number of neuron in hidden layer is not sufficient to make

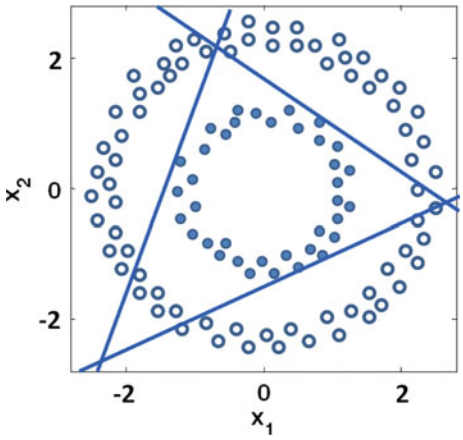


Fig. 9 This problem cannot be solved by any 2-3-1 MLP with hard-limit activation functions

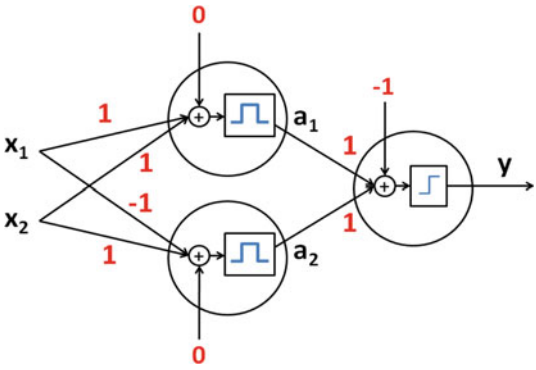


Fig. 10 MLP with $f_{\text{pulse}}(n)$, activation function to solve two rings of points classification problem

two classes linearly separable in the spaces spanned by the hidden responses. By increasing the number of hidden neurons into 6, these two classes separate linearly by output neuron and this classification problem can be solved.

Now, consider MLPs with nonmonotonic activation function, $f_{\text{pulse}}(n)$, described in Sect. 3, to solve this problem. Then, we will show that an MLP network with two hidden neurons and one output neuron can classify these two classes. Figure 10 shows a 2-2-1 MLP with $f_{\text{pulse}}(n)$ activation function, which is used to solve two rings of points classification problem.

Figure 11 shows how this network classifies them. Each activation functions create two hyperplane in the input space, and pattern whose net input is within two hyperplanes will give positive output, while all others give a negative output.

It is obvious that the input space transformed into a two-dimensional space where two classes are linearly separable in this space. The same results can be obtained by using sinusoidal activation functions. If the activation functions are sinusoidal, the number of regions is infinite, but there are only finite regions that patterns are located. In these regions, the outputs neurons are completely similar to the $f_{\text{pulse}}(n)$, activation functions described by Eq. (5).

Consequently, if nonmonotonic activation functions such as those described by Eq. (5) and sinusoidal activation functions are used, an MLP with two hidden nodes and one output node can easily solve this problem. On the other hands, if monotonic activation functions such as hard-limit and sigmoid one are used, we must use an MLP with a larger structure.

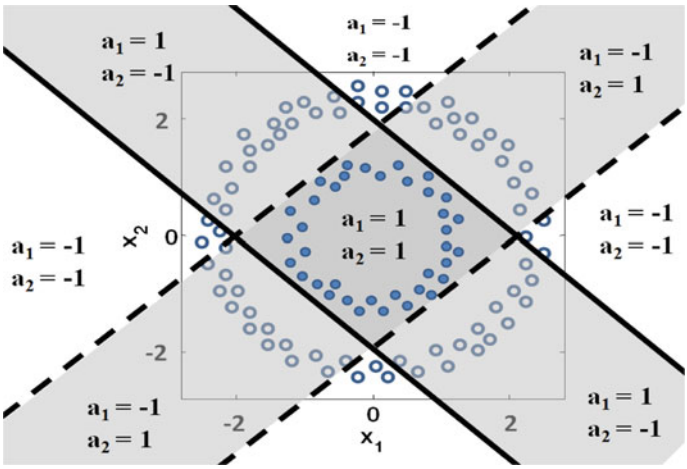


Fig. 11 Solving two rings of points classification problem by a 2-2-1 MLP with the nonmonotonic activation function of $f_{\text{pulse}}(n)$

Table 1 Comparing the architectural details of the networks needed for classifying two rings of points classification problem

	Hard-limit	$f_{\text{pulse}}(n)$	Sinusoidal
No. of hidden neurons	6	2	2
No. of adjustable parameters	25	9	9

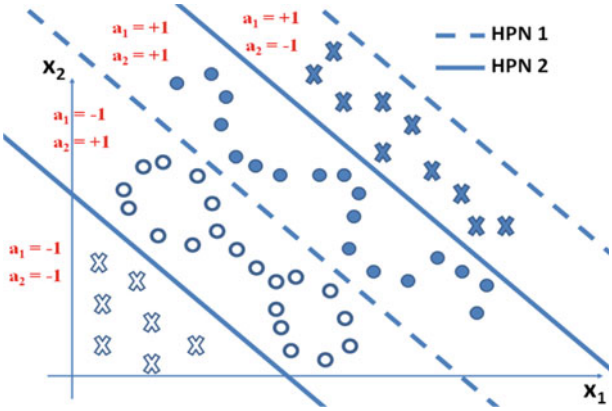


Fig. 12 Solving four classes classification problem by an MLP with two nonmonotonic activation functions of type $f_{\text{pulse}}(n)$

Table 1 compares the structure of MLPs for these activation functions in terms of number of hidden nodes and number of parameters needed to be updated.

3.1.2 Example 2: a multi-class classification problem

Consider data set shown in Fig. 12. There are four Gaussian distributed classes of patterns. An MLP classifier must create separation hyperplanes among four classes to classify them.

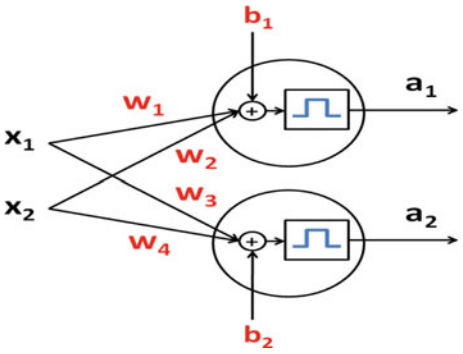


Fig. 13 MLP with nonmonotonic activation function of type $f_{\text{pulse}}(n)$ for solving a four-class classification problem

Table 2 Comparing the architectural details of the networks needed for four classes classification problem

	Hard-limit	$f_{\text{pulse}}(n)$	Sinusoidal
No. of neurons	6	2	2
No. of adjustable parameters	17	6	6

It may seem that we can separate two classes by using three hidden nodes to creating four separation hyperplanes in the input space between two classes. An MLP of 2-4-2 structure can successfully classify these four classes.

Now, we want to solve this problem using an MLP with nonmonotonic activation functions. Then, we will show that a network with only two neurons without any need to hidden neuron can classify these two classes. Figure 13 shows MLP with activation function of $f_{\text{squ}}(n)$, which is used to solve this problem.

Figure 12 shows how two neurons separate the input space into regions. Table 2 shows the output of each neuron in each of four regions where each region is candidate of a class. It confirms that these two neurons can easily solve this problem.

3.1.3 Example 3: HPN in the output layer

As discussed in Sect. 2, hidden layer(s) can be totally considered as a mapping that maps the input space to a new space called hidden space. In other words, by this mapping, new patterns called hidden patterns in the hidden space are achieved from the original patters in the input space. Then, the output layer must solve a new classification problem including classifying of hidden patterns. In most previous researches that the authors use nonmonotonic activation functions, they only use nonmonotonic functions in the hidden neuron. But in some classification problems, using a nonmonotonic activation function in the output layer reduces the required number of hidden neuron to solve the problem and consequently decreases the complexity of network. Suppose the binary classification problem depicted in Fig. 14. Suppose, we use a neural network with two hidden neurons with hard-limit activation function and one output neuron. Figure 15a shows that the patterns in the hidden space, which obtained from hidden layer mapping, are not linearly separable. If we use a neuron with hard-limiter activation function at the output layer, hidden patterns must be separable by only one line. Because of this limitation, we need to increase the number of neurons of the hidden layer until the hidden patterns can be separated by only one line. This limitation is not desirable since the complexity of training process significantly increases by adding neurons to hidden

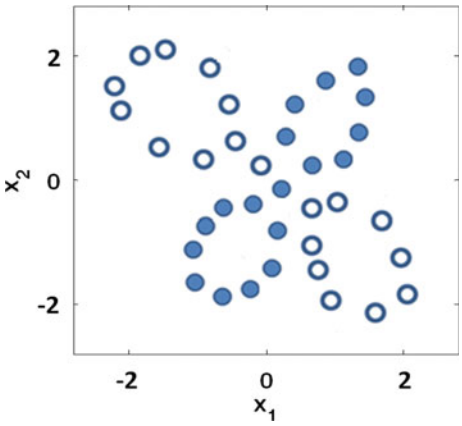


Fig. 14 A binary classification problem

layer. In contrast to conventional neuron, if the proposed neuron is used in the output layer, then the hidden patterns do not need to be separable by only one line since some nonlinear classification problems can be solved by only one HPN. For the problem in hand, it is clear that the patterns in the hidden space are like an XOR problem that can be easily separable by one HPN, and there is no need to increase the number of neurons in the hidden layer. Figure 15b shows the architecture of this network.

3.2 Nonlinear aggregation

In the previous section, we considered the positive role of using nonmonotonic activation functions in comparison with monotonic activation functions. Now, we aim to improve the structure of conventional neuron more by analyzing the effect of nonlinear aggregation mapping that can be compared with that of linear aggregation. As already discussed, aggregation mapping is a transformation over input vector and net input n. In the conventional neuron, aggregation is the linear mapping of Eq. (1). Consider the classification problems of Fig. 16. It is obvious that these problems cannot be classified using a single line. In other words, the conventional neuron of Fig. 1 cannot individually solve this problem, and a network of conventional neuron is required. In the following, a special nonlinear aggregation mapping is proposed to solve a special class of classification problems. Note that several nonlinear aggregation mapping can be presented, but we consider only one of them for analyzing the effect of nonlinear input mapping in forming the decision boundaries in classification problems.

Now, consider the neuron of Fig. 17 with the following nonlinear aggregation mapping (Eq. (8)) and hard-limiter activation function:

$$n = a_1 x_1^{n_1} + a_2 x_2^{n_2} + b \tag{8}$$

Where n_1, n_2, a_1, a_2 and b are adjustable parameters of mapping and can be real numbers. By changing the values of parameters of this mapping, various interesting and applied decision boundaries can be created that some of them are displayed in Fig. 18. Figure 16 shows some examples of nonlinear classification problems, which can be individually solved by neuron of Fig. 17.

It should be noted that, in addition to the values of parameters, the type of atomic functions, e.g., $\sin(.)$, $\exp(.)$, $(.)^r$, ..., and composite operators (e.g., $+, -, \times, \div$) which form the

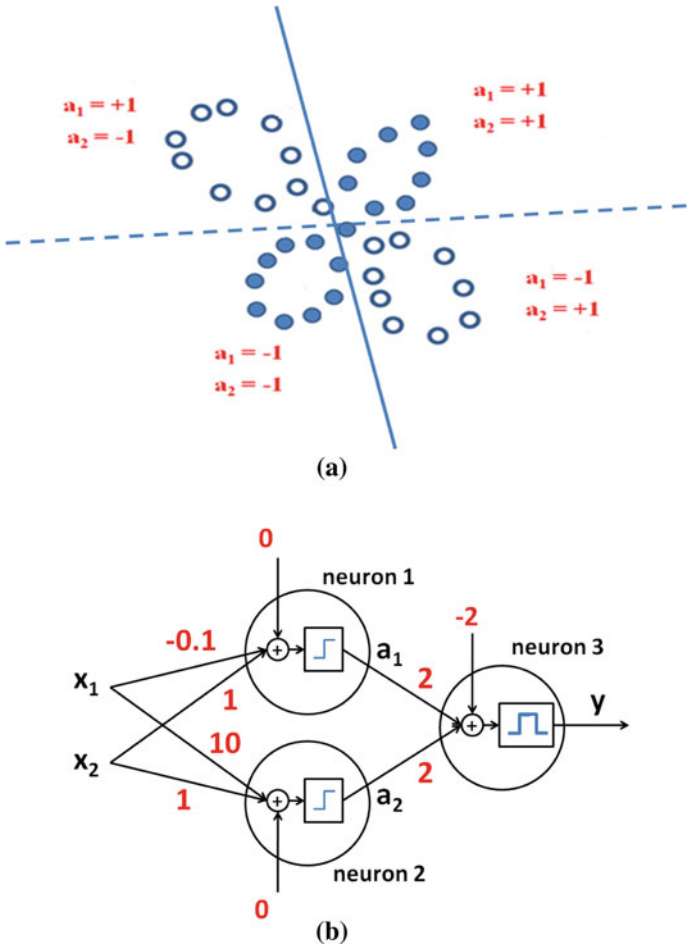


Fig. 15 **a** The patterns in the hidden space which obtained from hidden layer mapping are not linearly separable (similar to XOR problem). **b** HPN in the output layer

aggregation mapping can also be considered as decision variables and optimally determined through the training process.

3.3 HPN with nonlinear aggregation and nonmonotonic activation functions

In the previous two sections, we separately discussed the advantages of nonlinear aggregation mappings and nonmonotonic activation functions. The examples of previous sections demonstrate that if a neuron is equipped by each of these components, a powerful neuron is created that can individually solve many nonlinear classification problems that cannot be solved even by a network of conventional neurons if the number of neurons is not large enough. For more complex problems, combining nonlinear aggregation and nonmonotonic activation functions leads to creation of a more powerful neuron that benefits the advantages of both of these components, and thus, more complex classification problems can be solved. An example of this neuron is shown in Fig. 19. Using a nonlinear aggregation mapping and

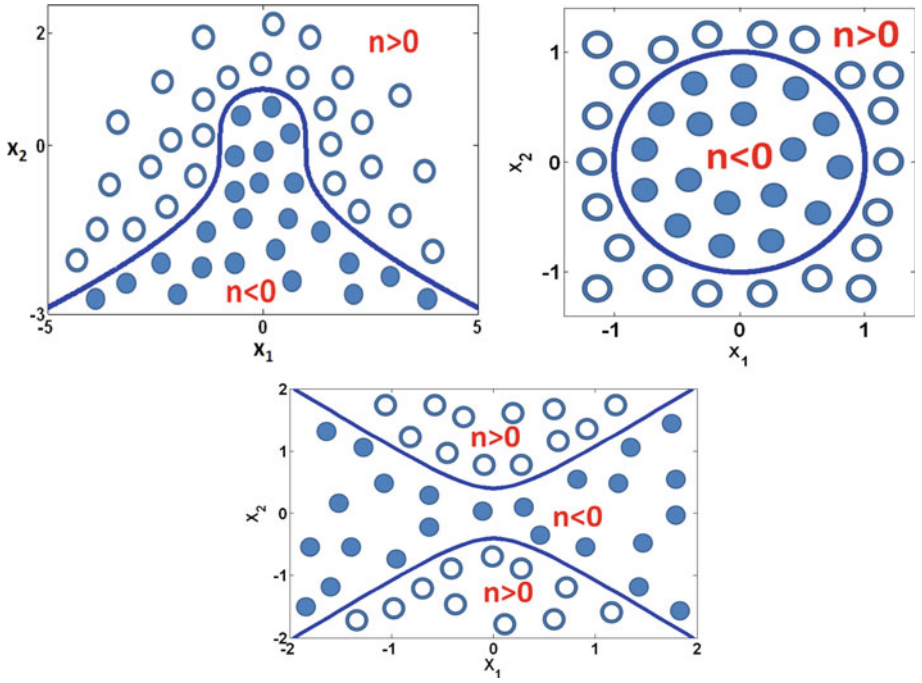


Fig. 16 The neuron of Fig. 17 can singly solve many nonlinear classification problems

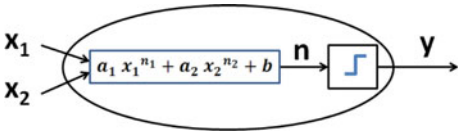


Fig. 17 Neuron with nonlinear aggregation mapping of Eq. (8) and hard-limiter activation function

a nonmonotonic activation function, a set of open or closed curves can be created. Figure 20 shows sample nonlinear classification problems that can be individually solved by the neuron of Fig. 19.

3.4 Evolutionary learning rule

In previous sections, we discuss the superiority of the proposed HPN, without any explanation of how we can find the weights of HPN. Generally, we can use any learning rule used for conventional neural network. Gradient descents are the most elegant and precise methods to learn neural network. However, gradient descent methods have the possibility of getting trapped at local optimum. Population-based search algorithms such as genetic algorithms (GA) ([15]) are found to have a better global perspective than the traditional methods in complex optimization problems. Among the advantages of GA in learning of neural networks are: (1) the objective function's gradient is not required. So, it can be used for nonderivative activation function, such as hard-limit and those introduced by Eq. (5); (2) it is not sensitive to starting point, and (3) it usually does not get stuck into so-called local optima. Based on these advantages, one can use GA to learn HPN neural networks. Implementation of the GA to determine the weights of the neural network has three basic stages: fitness evaluation, selection, and

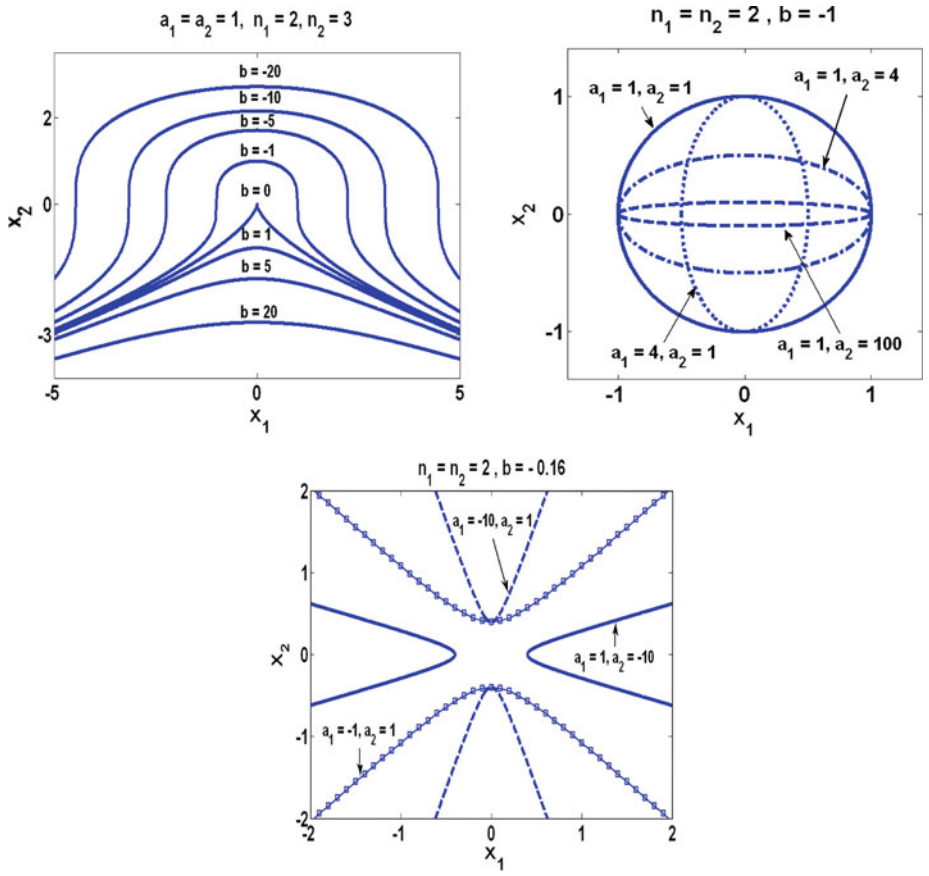


Fig. 18 By changing the values of parameters of nonlinear aggregation mapping of Eq. (8), various interesting and applied decision boundaries can be created

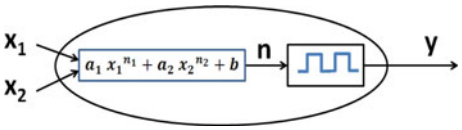


Fig. 19 Neuron with aggregation mapping of Eq. (8) and nonmonotonic activation function of Fig 6

breeding. Fitness evaluation needs the evaluating of the performance of all individuals in the population. Here, an individual is considered to be a separate set of network weights, with the fitness of the individual being a measure of the network's performance when these weights are being used for classification task. The population then consists of a collection of these individuals. Selection involves killing a given proportion of the population based on probabilistic "survival of the fittest". Killed individuals are replaced by children, who are created by breeding the remaining individuals in the population. For each child produced, breeding first requires probabilistic selection of two parent individuals, getting a more chance to fitter individuals to be chosen. Then, by applying of the crossover and mutation operators on the parent pair produces the new child. The crossover operator combines the information contained in two parent strings (i.e., two sets of network weights) by probabilistic copying

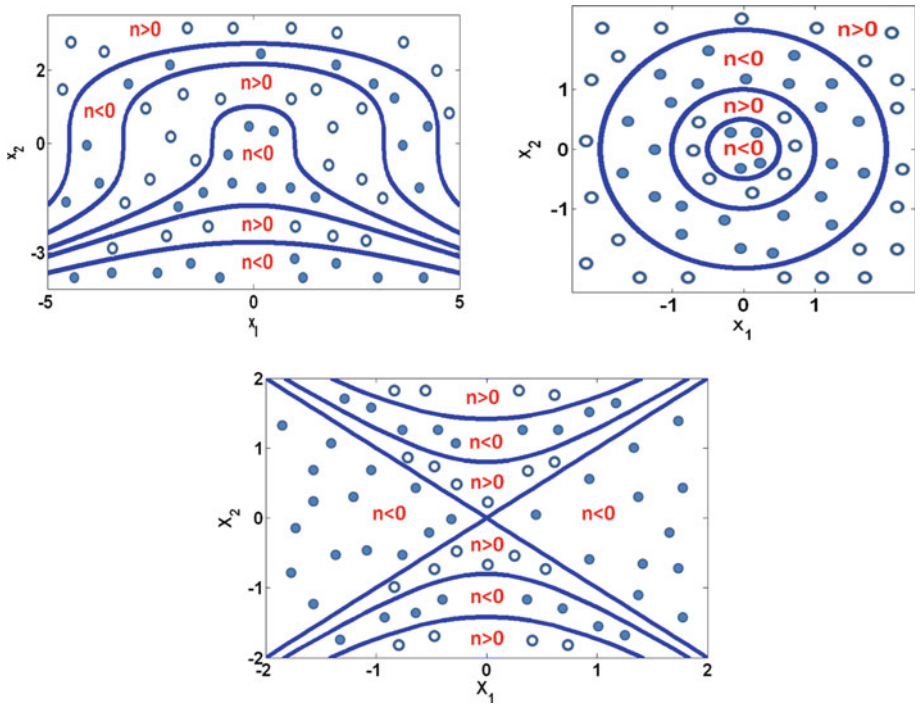


Fig. 20 The neuron of Fig. 19 can singly solve many nonlinear classification problems

of information from either parent to each corresponding string element of the child being produced. Mutation gives the algorithm a random search capability by random alterations of the network weights. GA iteratively improved the set of tentative solutions by applying the aforementioned stages to find a good solution.

3.5 Solving Iris data classification problem

While the main idea behind this paper is to show how the neuron structure in an MLP can affect the ability of neurons to deal with classification, it was shown in the previous sections by some 2-dimensional examples to visualize how the number of neurons can be reduced by choosing an appropriate neuron structure; however, in this section, in order to demonstrate the efficiency of the proposed approach in solving real-word problems, the well-known Iris data classification problem is considered. Iris is a benchmark data set first adopted by Fisher, a well-known statistics, to his discriminant experiment in 1936, and having been used by a lot of researchers for evaluating the performances of various classification methods. The Iris flower data consists of 50 samples from each of three species of Iris flowers as Iris setosa, Iris virginica, and Iris versicolor. Four features were measured from each sample including the length and the width of sepal and petal (UCI Machine Learning Repository). We take the first 25 patterns of every class (75 in all), to form the training set, and the others form the test set, in the same way as Daqia and Genxinga [3]. In the study by Daqia and Genxinga [3], the Iris classification problem is solved with an MLP network of size 4-3-3 with two well-known types of different activation functions, namely the hyperbolic tangent $\tanh(x)$ and the sigmoid $(1+\exp(-x))^{-1}$. The classification correction rates of both networks for the

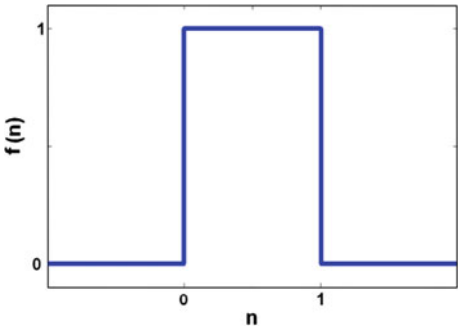


Fig. 21 Nonmonotonic activation function used in the MLP of Fig. 22

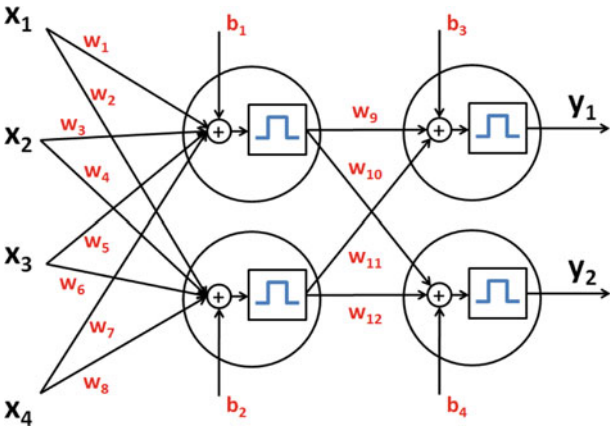


Fig. 22 The MLP which is used for solving Iris classification problem using the nonmonotonic activation function of Fig. 21

training set were 100% and for the testing set were 93.33. Now, we use an MLP network with nonmonotonic activation function presented in Fig. 21, to solve this classification problem. Figure 22 shows the MLP that can solve Iris classification problem. The network has only two hidden neurons. All activation functions are same and nonmonotonic as introduced in Fig. 21. GA algorithm described in the previous section is used to find the weights and biases of the MLP of Fig. 22. The optimal values of weights and biases found by GA are listed in Table 3. The result is shown in Table 3. Using these optimal values, the MLP of Fig. 22 can correctly recognize all of 75 samples of training set and 70 samples of 75 samples of testing set. This means a recognition rate of 100% for training set and 93.33% for testing set. Comparing the results obtained by HPN with that of the hyperbolic tangent and the sigmoid activation functions, this example demonstrates HPN can decrease the complexity of MLP while preserving the performance.

4 Conclusion

A general neuron structure that involves a more sophisticated architecture, including possible nonlinear aggregations in neuron input mapping and nonmonotonic activation functions, was proposed. The examples, to visualize how the number of neurons, can be reduced by

Table 3 Optimal values of weights and biases of the MLP of Fig. 22

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
0.1383	−0.4601	0.6460	−0.5833	0.3732	0.2545	−0.4780	−0.6181
w_9	w_{10}	w_{11}	w_{12}	b_1	b_2	b_3	b_4
−0.5522	0.7304	−0.1264	0.4139	0.5292	0.3940	0.4441	−0.1043

choosing an appropriate neuron structure, and some 2-dimensional examples were provided and solved manually. Moreover, to show the efficiency of the proposed scheme in solving real-world classification problems, the Iris data classification problem was solved using an MLP whose neurons are equipped by nonmonotonic activation functions. The presented simulation examples and our solution to Iris data classification problem demonstrated that how the proposed general neuron structure can often improve the ability of neurons to deal with classification problems.

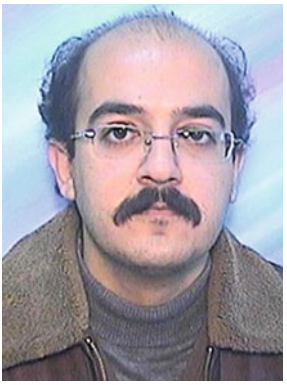
References

1. Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press, New york
2. Chandra P, Singh Y (2004) An activation function adapting training algorithm for sigmoidal feedforward networks. Neurocomputing 61:429–437
3. Daqia G, Genxinga Y (2003) Influences of variable scales and activation functions on the performances of multilayer feedforward neural networks. Pattern Recognit 36:869–878
4. Daqi G, Yan J (2005) Classification methodologies of multilayer perceptrons with sigmoid activation functions. Pattern Recognit 38:1469–1482
5. Durbin R, Rumelhart D (1989) Products units: a computationally powerful and biologically plausible extension to backpropagation networks. Neural Comput 1:133–142
6. Duch W, Janhowski N (1999) Survey of neural transfer functions. Neural Comput 2:163–212
7. Efe M (2008) Novel neuronal activation functions for feedforward neural networks. Neural Process Lett 8:63–79
8. Funahashi K (1998) Multilayer neural networks and Bayes decision theory. Neural Netw 11:209–213
9. Garfield S, Wermter S (2006) Call classification using recurrent neural networks, support vector machines and finite state automata. Knowl Inf Syst 9:131–156
10. Gibson GJ, Cowan CFN (1990) On the decision regions of multilayer perceptrons. In: Proceedings of IEEE 78(9):1590–1594
11. Guarnieri S, Piazza F, Uncini A (1999) Multilayer feedforward networks with adaptive spline activation function. IEEE Trans Neural Netw 10:672–684
12. Gutierrez PA, Hervas C, Carbonero M et al (2009) Combined projection and kernel basis functions for classification in evolutionary neural networks. Neurocomputing 72:2731–2742
13. Hagan MT, Demuth HB, Beale MH (1999) Neural network design. PWS Publishing Company, Boston
14. Hara K, Nakayamma K (1994) Comparison of activation functions in multilayer neural network for pattern classification. In: IEEE world congress on computational intelligence, IEEE in conference on neural networks, pp 2997–3002
15. Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor
16. Homik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4:251–257
17. Homik K (1993) Some new results on neural network approximation. Neural Netw 6:1069–1072
18. Huang SC, Huang YF (1991) Bounds on the number of hidden neurons in multilayer perceptrons. IEEE Trans Neural Netw 2:47–55
19. Huang GB, Chen YQ, Babri HA (2000) Classification ability of single hidden layer feedforward neural networks. IEEE Trans Neural Netw 11:799–801
20. Karaköse M, Akin E (2004) Type-2 fuzzy activation function for multilayer feedforward neural networks. In: IEEE international conference on systems, man and cybernetics, pp 3762–3767

21. Lapedes A, Farber R (1988) How neural nets work. In: Anderson D (ed) Neural information processing systems. American Institute of Physics, New York, pp 442–456
22. Leshno M, Lin VY, Pinkus A et al (1993) Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw 6:861–867
23. Lin SW, Chen SC, Wu WJ et al (2009) Parameter determination and feature selection for back-propagation network by particle swarm optimization. Knowl Inf Syst 21:249–266
24. Liu Q, Wang J (2008) Two *k*-winners-take-all networks with discontinuous activation functions. Neural Netw 21:406–413
25. Lippmann RP (1989) Pattern classification using neural networks. IEEE Commun Mag 27:47–64
26. Menon A, Mehrotra K, Mohan CK et al (1996) Characterization of a class of sigmoid functions with applications to neural networks. Neural Netw 9:819–835
27. Pao YH, Takefuji Y (1992) Functional-link net computing: theory, system architecture, and functionalities. IEEE Comput 25:76–79
28. Park DC (2009) Multiresolution-based bilinear recurrent neural network. Knowl Inf Syst 19:235–248
29. Piazza F, Uncini A, Zenobi M (1993) Artificial neural networks with adaptive polynomial activation function. In: Proceeding of the IJCNN, Bejing, pp 343–349
30. Reddy CK, Park JH (2010) Multi-resolution boosting for classification and regression problems. Knowl Inf Syst. doi:10.1007/s10115-010-0358-0
31. Specht DF (1991) A general regression neural network. IEEE Trans Neural Netw 2:568–576
32. Singh Y, Chandra P (2003) A class +1 sigmoidal activation functions for FANNs. J Econ Dyn Control 28:183–187
33. Sopena JM, Romero E, Alquezar R (1999) Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. In: International conference on artificial neural networks, pp. 323–328
34. UCI Machine Learning Repository: Iris Data Set. http://archive.ics.uci.edu/ml/datasets/Iris
35. Vecci L, Piazza F, Uncini A (1998) Learning and approximation capabilities of adaptive spline activation function neural networks. Neural Netw 1:259–270
36. Vaughn ML (1999) Derivation of the multilayer perceptron weight constraints for direct network interpretation and knowledge discovery. Neural Netw 12:1259–1271
37. Wu Y, Zhao M, Ding X (1997) Beyond weights adaptation: a new neuron model with trainable activation function and its supervised learning. In: International conference on neural networks, pp 1152–1157

Author Biographies

Hadi Sadoghi Yazdi is currently an Associate Professor of Computer Science and Engineering at Ferdowsi University of Mashhad (FUM). He received his B.S. degree in Electrical Engineering from FUM in 1994 and received his M.S. and Ph.D. degrees in Electrical Engineering from Tarbiat Modares University in 1996 and 2005, respectively. Dr. Sadoghi Yazdi has received several awards including Outstanding Faculty Award and Best System Design Award in 2007. His research interests are in the areas of Pattern Recognition, Machine Learning, Machine Vision, Signal Processing, Data Mining, and Optimization.



Alireza Rowhanimanesh is currently a Doctoral Student of Control Engineering at Ferdowsi University of Mashhad (FUM). His Ph.D. thesis is on Optimal Design of Swarms under the supervision of Prof. M.-R. Akbarzadeh-T. He received his B.S. and M.S. degrees in Control Engineering from FUM in 2007 and 2009, respectively. He has received several awards including Outstanding Graduate Student Award in 2009, Best Paper Award in ISFS2008, and Outstanding Student Awards in 2007 and 2003. His research interests are in the areas of Collective Intelligence, Nanomedicine, Perception-Based Computing, Soft Computing, and Robotics.

Hamidreza Modares is currently a Doctoral Student of Control Engineering at Ferdowsi University of Mashhad (FUM). His Ph.D. thesis is on Optimal Control of Unknown Nonlinear Systems under the supervision of Dr. M.-B. Naghibi-Sistani. He received his B.S. degree in Electrical Engineering from Tehran University in 2004 and then received his M.S. degree in Control Engineering from Shahrood University of Technology in 2006. His research interests are in the areas of Optimal Control, Machine Learning, Evolutionary Computing, Artificial Neural Networks, and Pattern Recognition.