Neural Process Lett (2011) 34:193–208
DOI 10.1007/s11063-011-9191-z

Supervised Learning Probabilistic Neural Networks

I-Cheng Yeh · Kuan-Cheng Lin

Published online: 22 July 2011

Abstract This study proposed supervised learning probabilistic neural networks (SLPNN) which have three kinds of network parameters: variable weights representing the importance of input variables, the reciprocal of kernel radius representing the effective range of data, and data weights representing the data reliability. These three kinds of parameters can be adjusted through training. We tested three artificial functions as well as 15 benchmark problems, and compared it with multi-layered perceptron (MLP) and probabilistic neural networks (PNN). The results showed that SLPNN is slightly more accurate than MLP, and much more accurate than PNN. Besides, the data weights can find the noise data in data set, and the variable weights can measure the importance of input variables and have the greatest contribution to accuracy of model among the three kinds of network parameters.

Keywords Supervised learning · Probabilistic neural network · Variable importance · Regression · Classification

List of Symbols

$f_A(X)$	The probabilistic density function of Category A at point X
f_p	The weight of the p-th sample in the sample base
h_p	The data weight of the p-th sample in the sample base
M	The number of output variables
m	The number of input variables
n	The number of samples in the sample base
n_A	The number of training vectors of Category A
t_{pq}	The known value of the q-th output variable of the p-th sample in the sample base
V_p	The reciprocal of kernel radius of the p-th sample in the sample base

I.-C. Yeh (✉) · K.-C. Lin
Department of Information Management, Chung Hua University, Hsinchu, Taiwan
e-mail: icyeh@chu.edu.tw

W_i	The i-th input variable weight
$\mathbf{X}$	The testing data vectors
$\mathbf{X}_{A\mathbf{p}}$	The p-th training data of Category A
x_i	The value of i-th input variable in the testing sample
x_i^p	The i-th input variable of the p-th sample in the sample base
y	The inference value of the output variable of the training data
σ	The smooth parameter
σ_p	The smooth parameter of the p-th sample in the sample base

1 Introduction

Probabilistic neural network (PNN) is a common neural network model, which is based on Bayesian classifier and the probabilistic density function [1–7]. PNN has a wide range of applications in model identification, time series prediction, as well as fault diagnosis and other fields [8–11]. Generally, the probabilistic density function is the normal probabilistic density function as follows.

$$f_A(X) = \frac{1}{(2\pi)^{\frac{m}{2}}\sigma^m}\left(\frac{1}{n_A}\right)\sum_{p=1}^{n_A}\exp\left(-\frac{(X - X_{Ap})'(X - X_{Ap})}{2\sigma^2}\right) \tag{1}$$

where $f_A(X)$ represents the value of probabilistic density function of Category A at point X; m represents the number of input variables; σ represents smooth parameter; n_A represents the number of training vectors in Category A; X represents the testing data vectors; X_{Ap} represents the p-th training data in Category A.

Because

$$\frac{1}{(2\pi)^{\frac{m}{2}}\sigma^m}\left(\frac{1}{n_A}\right) = \text{ Constant } = h \tag{2}$$

$$(\mathbf{X} - \mathbf{X}_{A\mathbf{p}})'(\mathbf{X} - \mathbf{X}_{A\mathbf{p}}) = \sum_{i=1}^{m}\left(x_i - x_i^{Ap}\right)^2 \tag{3}$$

the probabilistic density function can be simplified as follows

$$f_A(X) = h\sum_{p=1}^{n_A} f_{Ap} \tag{4}$$

where

$$f_{Ap} = \exp\left(-\frac{\sum_{i=1}^{m}(x_i - x_i^{Ap})^2}{2\sigma^2}\right) \tag{5}$$

where x_i represents the value of i-th input variable in the testing sample; x_i^{Ap} represents the i-th input variable of the p-th sample of Category A in the sample base.

PNN can deal with classification problems, but it cannot fulfill the regression. Hence, Specht [7] proposed general regression neural network (GRNN) to improve the network output with the weighted average method,

$$y = \frac{\sum_{p=1}^{n} f_p \cdot t_p}{\sum_{p=1}^{n} f_p} \tag{6}$$

where t_p = the known output variable of the p-th sample in the sample base; f_p = the weight of the p-th sample in the sample base; n = number of samples in the sample base.

The algorithm of PNN has a number of different variations [4,9,11–18]. For example, Specht [4] pointed out that PNN learn quickly from examples in one pass and asymptotically achieve the Bayes-optimal decision boundaries. The major disadvantage of a PNN is that it requires one node or neuron for each training pattern. Various clustering techniques have been proposed to reduce this requirement to one node per cluster center.

Song, et al. [9] proposed a modified probabilistic neural network (PNN) for brain tissue segmentation with magnetic resonance imaging (MRI). In this approach, covariance matrices are used to replace the singular smoothing factor in the PNN's kernel function, and weighting factors are added in the pattern of summation layer.

Rutkowski [11] proposed a new class of PNN working in nonstationary environment. He formulated the problem of pattern classification in nonstationary environment as the prediction problem and designed a probabilistic neural network to classify patterns having time-varying probability distributions.

Berthold and Diamond [12] proposed a constructive training algorithm for probabilistic neural networks, a special type of radial basis function networks. In contrast to other algorithms, predefinition of the network topology is not required. The proposed algorithm introduces hidden units whenever necessary and adjusts the shape of existing units individually to minimize the risk of misclassification. This leads to smaller networks compared to classical PNNs and therefore enables the use of large data sets.

Galleske, et al. [17] pointed out that a convention PNN applies a same covariance matrix for all training examples. Although this conventional process can lead to fast training, but the major shortcoming is that the generalization ability is not optimized. For this reason, the authors not only introduced a covariance matrix to replace a smoothing factor, but also proposed a new recursive algorithm to calculate the elements of the covariance matrix for the PNN's Gaussian kernel function.

Montana [18] noted that a conventional PNN which uses an isotropic Gaussian kernel function is easily falling into nearest-neighbor-like algorithms, and it is not robust with respect to affine transformations of feature space. Therefore, the author proposed a weighted PNN which uses an anisotropic Gaussian kernel function to extend a conventional PNN. In addition, the covariance matrix of the Gaussian kernel function is optimized by a genetic algorithm.

PNN has only one smooth parameter that must be adjusted in the learning process; hence, it has the quick learning ability. However, it has two shortcomings: (1) it is relatively inaccurate when the amount of data is small; (2) the model is lack of capacity to assess the importance of input variables.

This study proposed supervised learning probabilistic neural networks (SLPNN), which have three kinds of network parameters:

(1) The variable weights decide the shape of probabilistic density function so that the contour lines are no longer round but elliptic. The larger the variable weight of the variable, the more important the variable, and the smaller the radius of the ellipse in the direction of the variable;
(2) The reciprocal of kernel radius is equivalent to the reciprocal of the smooth parameter in the traditional probabilistic density function. The larger the kernel radius (the smaller reciprocal of kernel radius), the larger the effective range of the sample;
(3) The data weight is equivalent to the height of probabilistic density function. The larger the height, the higher the credibility of the sample.

These three kinds of parameters can be adjusted through training, and can improve the accuracy of the model and estimate the importance of input variables.

The main contribution and innovation of this study is the unified mathematic theoretical framework for PNN. The traditional PNN often decides the smooth parameters (kernel radius parameters) of probabilistic density function by trial and error method, Although there are three kinds of network parameters in SLPNN. In this study, we used the principle of minimizing error sum of squares to derive the supervised learning rules for all the parameters, including kernel radius, shape, and height parameters of the transfer function, under a unified mathematic theoretical framework.

In Sect. 2, we will derive the learning rules of SLPNN. In Sects. 3 and 4, we will verify the performance of SLPNN and compare it with MLP and PNN, respectively with three artificial functions and 15 actual classification problems. In Sec. 5, we will explore the effects of each network parameters (variable weight, reciprocal of kernel radius, and data weight) on accuracy. In Sect. 6, we will make a summary of the testing results in the entire study.

2 Theory

In this study, the architecture of SLPNN is shown in Fig. 1, and the transfer function of the hidden neuron is as follows.

$$f_p = h_p^2 \exp\left(-\frac{\sum_{i=1}^{m} W_i^2 (x_i - x_i^p)^2}{2\sigma_p^2}\right) \tag{7}$$

where

x_i = the i-th input variable of the unknown samples;

x_i^p = the i-th input variable of the p-th sample in the training set;

W_i = the i-th input variable weight, which decides the shape of contour lines of the probabilistic density function;

h_p = the data weight of the p-th sample in the training set, which is equivalent to the height of the probabilistic density function;

σ_p = the smooth parameter of the p-th sample in the training set, which controls the width of the probabilistic density function.

The smooth parameter may be approaching to 0 in the learning process so as to make the denominator be zero. In order to avoid the problem, we used the reciprocal of kernel radius instead of smooth parameter.

Let

$$V_p = \frac{1}{\sqrt{2}\sigma_p} \tag{8}$$

where V_p = the reciprocal of kernel radius of the p-th sample in the training set.

Hence

$$f_p = h_p^2 \exp\left(-V_p^2 \sum_{i=1}^{m} W_i^2 \left(x_i - x_i^p\right)^2\right) \tag{9}$$

Let

$$D_p \equiv V_p^2 \sum_{i=1}^{m} W_i^2 \left(x_i - x_i^p\right)^2 \tag{10}$$

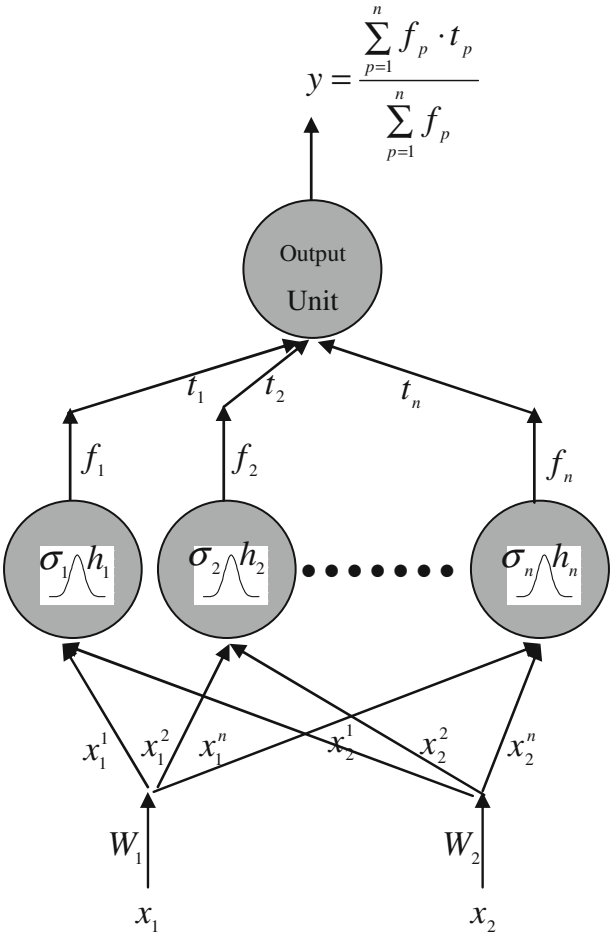

Fig. 1 Supervised learning probabilistic neural networks (SLPNN)

then

$$f_p = h_p^2 \exp\left(-D_p\right) \tag{11}$$

The effects of variable weights, data weights, and smoothing parameters on the transfer function are shown in Figs. 2, 3, and 4. These figures show the three kinds of parameters control the shape, height, and radius of transfer function, respectively.

In order to derive the supervised learning rules to adjust the parameters in the equations above, let the error function be

$$E = \frac{1}{2}(t - y)^2 \tag{12}$$

where t = the known value of the output variable of the training data; y = the inference value of the output variable of the training data, and can be calculated by Eq. 6.

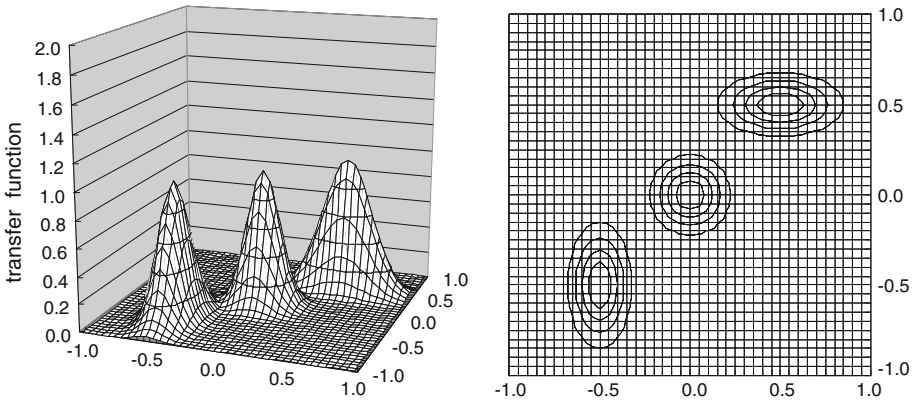

Fig. 2 The effects of variable weights W_i on the transfer functions (the *upper-right*, *middle*, and *lower-left* contour is the transfer function of the neuron whose $W_1 < W_2$, $W_1 = W_2$, and $W_1 > W_2$, respectively)

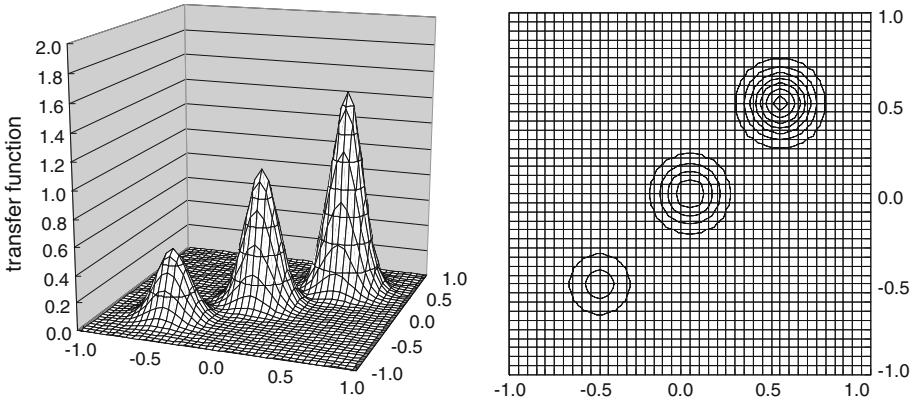

Fig. 3 The effects of data weights h_p on the transfer functions (the *upper-right*, *middle*, and *lower-left* contour is the transfer function of the neuron with large, middle, and small data weight, respectively)

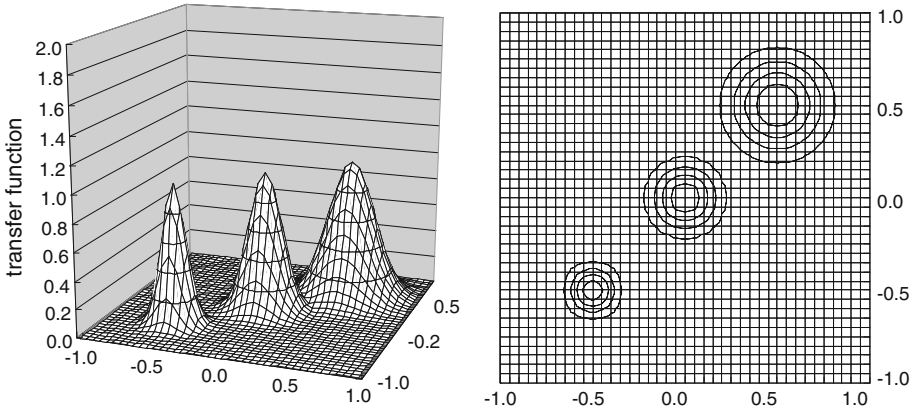

Fig. 4 The effects of smoothing parameters σ_p on the transfer functions (the *upper-right*, *middle*, and *lower-left* contour is the transfer function of the neuron with large, middle, and small smoothing parameter, respectively)

The goal to adjust the network parameters is to minimize the error function. Therefore, the supervised learning rules of network parameters can be derived by the steepest descent method as follows.

- **variable weights**

$$\Delta W_i = -\eta \frac{\partial E}{\partial W_i} \tag{13}$$

where $\eta =$ the learning rate

According to the chain rule in the partial differential

$$\frac{\partial E}{\partial W_i} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial W_i} \tag{14}$$

According to the chain rule in the partial differential

$$\frac{\partial E}{\partial W_i} = \frac{\partial E}{\partial y} \sum_p \frac{\partial y}{\partial f_p} \frac{\partial f_p}{\partial D_p} \frac{\partial D_p}{\partial W_i} \tag{15}$$

According to Eq. 12

$$\frac{\partial E}{\partial y} = -(t - y) \tag{16}$$

According to Eq. 6

$$\frac{\partial y}{\partial f_p} = \frac{\left(\sum_p f_p\right) t_p - \left(\sum_p f_p t_p\right) 1}{\left(\sum_p f_p\right)^2} = \frac{t_p - \left(\sum_p f_p t_p / \sum_p f_p\right)}{\sum_p f_p} = \frac{t_p - y}{\sum_p f_p} \tag{17}$$

According to Eq. 11

$$\frac{\partial f_p}{\partial D_p} = -h_p^2 \exp\left(-D_p\right) \tag{18}$$

According to Eq. 10

$$\frac{\partial D_p}{\partial W_i} = V_p^2 \left(2W_i \left(x_i - x_i^p\right)^2\right) \tag{19}$$

Substitute Eqs. 16–19 into Eq. 15, and then substitute it into Eq. 13.

$$\Delta W_i = -\eta \frac{\partial E}{\partial W_i} = \eta (t - y) \sum_p \frac{t_p - y}{\sum_p f_p} \left(-h_p^2 \exp\left(-D_p\right)\right) \cdot V_p^2 \left(2W_i \left(x_i - x_i^p\right)^2\right) \tag{20}$$

Let δ_P be the gap between the p-th predictive output value and its actual output value

$$\delta_p \equiv (t - y)\left(t_p - y\right) h_p \exp\left(-D_p\right) \tag{21}$$

Eq. 20 can be simplified as follows.

$$\Delta W_i = -2\eta \frac{1}{\sum_p f_p} \sum_p \delta_p h_p \cdot V_p^2 W_i (x_i - x_i^p)^2 \tag{22}$$

- **Data weights**

$$\Delta h_p = -\eta \frac{\partial E}{\partial h_p} \tag{23}$$

According to the chain rules in the partial differential, we can get

$$\frac{\partial E}{\partial h_p} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial f_p}\frac{\partial f_p}{\partial h_p} \tag{24}$$

According to Eq. 11

$$\frac{\partial f_p}{\partial h_p} = 2h_p \cdot \exp\left(-D_p\right) \tag{25}$$

Substitute Eqs. 16, 17, and 25 into Eq. 24, and then substitute it into Eq. 23.

$$\Delta h_p = -\eta \frac{\partial E}{\partial h_p} = -\eta(-(t-y))\frac{t_p - y}{\sum_p f_p} 2h_p \exp\left(-D_p\right) \tag{26}$$

The equation above can be simplified by Eq. 21.

$$\Delta h_p = 2\eta \frac{1}{\sum_p f_p}\delta_p \tag{27}$$

- **Reciprocal of kernel radius**

$$\Delta V_p = -\eta \frac{\partial E}{\partial V_p} \tag{28}$$

According to the chain rules in the partial differential, we can get

$$\frac{\partial E}{\partial V_p} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial f_p}\frac{\partial f_p}{\partial D_p}\frac{\partial D_p}{\partial V_p} \tag{29}$$

According to Eq. 10

$$\frac{\partial D_p}{\partial V_p} = 2V_p \sum_i W_i^2 \left(x_i - x_i^p\right)^2 \tag{30}$$

Substitute Eqs. 16–18, and 30 into Eq. 29, and then substitute it into Eq. 28.

$$\Delta V_p = -\eta \frac{\partial E}{\partial V_p} = \eta(t-y)\frac{t_p - y}{\sum_p f_p}(-h_p^2 \exp(-D_p))2V_p \sum_i W_i^2 \left(x_i - x_i^p\right)^2 \tag{31}$$

The equation above can be simplified by Eq. 21.

$$\Delta V_p = -2\eta \frac{1}{\sum_p f_p}\delta_p h_p \cdot V_p \sum_i W_i^2 \left(x_i - x_i^p\right)^2 \tag{32}$$

These algorithms can gradually optimize the kernel shape, height, and radius parameter of the transfer function in the learning process, so that each input variable has a various variable weight, making the transfer function in an appropriate ellipse according to the training data, and thus reduce the error of the model.

The above network architecture only has one output neuron. To deal with the multi-classes problem, Fig. 1 can be change into multi-output neurons, and the formula of the q-th output neuron is as follows:

$$y_q = \frac{\sum_{p=1}^{n} f_p t_{pq}}{\sum_{p=1}^{n} f_p} \tag{33}$$

where t_{pq} = the q-th known output variable of the p-th sample in the sample base.

After using the steepest descent method to derive the supervised learning rules of network with multi-output neurons, it is found that its rules are the same as Eqs. 22, 27, and 32. The only difference is that the gap of the p-th hidden neuron is changed into

$$\delta_p = \sum_{q=1}^{M} \left(t_q - y_q\right)\left(t_{pq} - y_q\right) h_p \exp\left(-D_p\right) \tag{34}$$

where M = the number of output variables; t_q = the known value of the q-th output variable of the training data; y_q = the inference value of the q-th output variable of the training data.

3 Application to Synthetic Data Sets

3.1 Evaluation of Accuracy

In this study, we designed three different types of function to test and investigate the performance and characteristics of SLPNN. In these functions, there are ten input variables ($X_1,\ X_2, \ldots\ X_{10}$) and an output variable. The input variables of each function are in the range from 0 to 1, producing 1000 training data and 100 testing data. The three data sets will be used to compare the accuracy of multi-layer perceptron (MLP), probabilistic neural network (PNN), and SLPNN.

In order to get the best results, we established the optimal model with different network frameworks and learning parameters.

- MLP: with different numbers of hidden layer units (2, 4, 8, 16, and 32) and learning rates (0.1, 0.3, 1.0, 3.0, and 10.0).
- PNN: with different smooth parameters (0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, and 10.0).
- SLPNN: with different learning rates (0.1, 0.3, 1.0, 3.0, and 10.0) and the initial reciprocals of kernel radius (0.1, 0.3, 1.0, 3.0, and 10.0).

Example 1 Linear and quadratic function

Assuming that in the following function

$$\begin{aligned} f(X) = 4\big[&0(X_1 - 0.5)^2 - 1(X_2 - 0.5)^2 + 2(X_3 - 0.5)^2 - 4(X_4 - 0.5)^2 \\ &+8(X_5 - 0.5)^2\big] + 0X_6 + 1X_7 - 2X_8 + 4X_9 - 8X_{10} \end{aligned} \tag{35}$$

In Eq. 35, it is clear that the relationship between ($X_1,\ X_2, \ldots\ X_5$) and output variable is quadratic and increases gradually, and that the relationship between ($X_6,\ X_7, \ldots\ X_{10}$) and output variable is linear and increases gradually. The results showed that, in different network frameworks and learning parameters, the coefficients of determination of the testing data of optimal model in MLP, PNN, and SLPNN were 0.9989, 0.7424, and 0.9707, indicating that the accuracy of SLPNN was only slightly lower than that of MLP, while far higher than that of PNN.

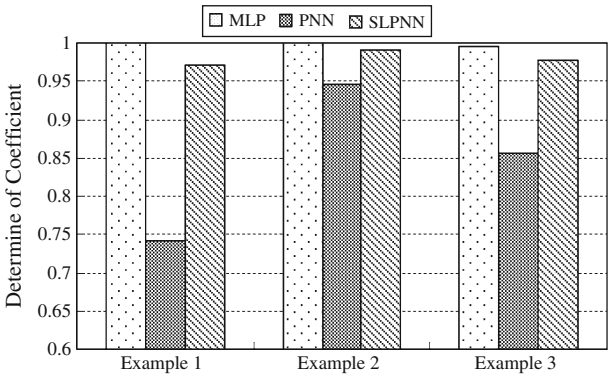

Fig. 5 The comparisons of the coefficients of determination in the three numerical examples

Example 2 Mixed function

Assuming that in the following function

$$f(X) = X_1 - X_2 + (X_5 - 0.25)^2 - (X_6 - 0.75)^2 + (X_9 - 0.25)(X_{10} - 0.75) \quad (36)$$

X_1 and X_2 are the linear effective variables; X_5 and X_6 are the curvature effective variables, X_9 and X_{10} are a group of variables with interaction, and the rest four variables are irrelevant variables. The results showed that the coefficients of determination in the testing data of optimal models in MLP, PNN and SLPNN were 0.9993, 0.9466, and 0.9917, indicating the accuracy of SLPNN was slightly lower than that of MLP, while far higher than that of PNN.

Example 3 Interactive and quadratic function

Assuming that in the following function,

$$f(X) = 1.5(X_1 - 0.5)(X_3 - 0.5) + 1.5(X_5 - 0.5)(X_7 - 0.5) + (X_9 - 0.5)^2 \quad (37)$$

In this function, $\{X_1,\ X_3\}$ and $\{X_5,\ X_7\}$ are the two groups of interactive variables; X_9 is the curvature effective variable; and the rest five variables are irrelevant variables. The results showed that the coefficients of determination in the testing data of the optimal models in MLP, PNN and SLPNN were 0.9953, 0.8551, and 0.9786, indicating the accuracy of SLPNN was slightly lower than MLP, while far higher than that of PNN.

Figure 5 shows the comparison of the coefficients of determination in the three numerical examples. It is clear that the coefficient of determination of SLPNN is close to that of MLP and much higher than that of PNN, indicating that the accuracy of SLPNN is far higher than that of PNN. This is probably because in the learning process, unlike PNN that uses the fixed variable weights, SLPNN can adjust the variable weights, the reciprocals of kernel radius, and the data weights at the same time so that the shapes of the transfer functions can be closer to the nature of the regression problem to reduce of the error of the model.

3.2 Evaluation of Variable Weights

Because both MLP and PNN cannot present the importance of input variables, therefore this model is the black box model. However, the variable weight of SLPNN may show the importance of input variables, such a characteristic is one of the advantages of SLPNN superior to

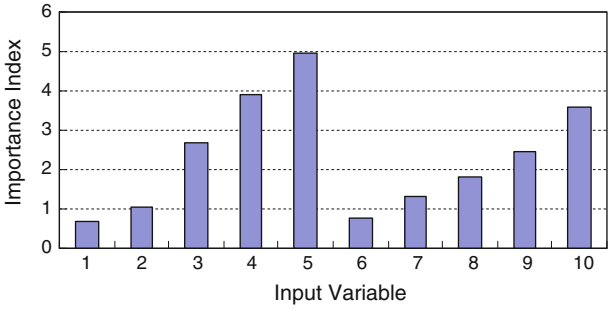

Fig. 6 Weight variables of SLPNN of Example 1

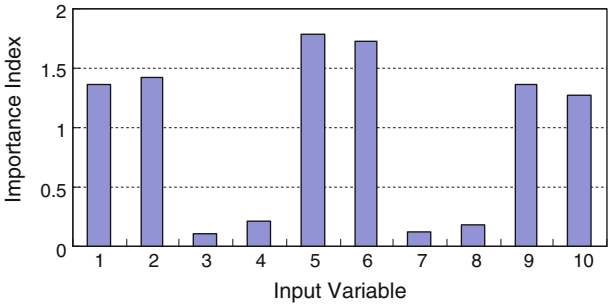

Fig. 7 Weight variables of SLPNN of Example 2

MLP and PNN. In this section, the three examples in the previous section were employed to verify such a characteristic.

(1) Figure 6 shows the variable weights of example 1. The variable weights of X_1, $X_2, \ldots X_5$ and X_6, $X_7, \ldots X_{10}$ on the output variable increase gradually.
(2) Figure 7 shows the results of the variable weights of example 2. In the chart, we can see that X_1, X_2, X_5, X_6, X_9 and X_{10} have much larger variable weights, indicating that they are important variables; the rest four variables have much smaller variable weights, indicating that they are irrelevant variables.
(3) Figure 8 shows the results of the variable weights in example 3. In the chart, we can see that X_1, X_3, X_5, X_7 and X_9 have much larger variable weights, indicating that they are important variables; the rest five variables have much smaller variable weights, indicating that they are irrelevant variables.

These results confirmed that the weight variables of SLPNN indeed can measure the importance of input variables on output variables for various functions.

3.3 Evaluation of Data Weights

In order to prove that the learn rule of data weight can make the noise data achieve to low data weight so as to reduce the adverse effects of noise data, and improve the accuracy of the model, we add a random value into four data in the data set of example 1. The random value is a normal distribution random variable with zero of average and its standard deviation is five times of the standard deviation of the output variable.

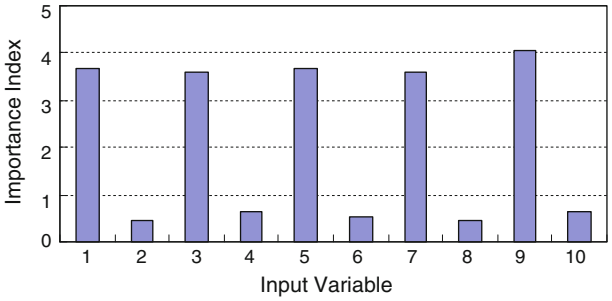

Fig. 8 Weight variables of SLPNN of Example 3

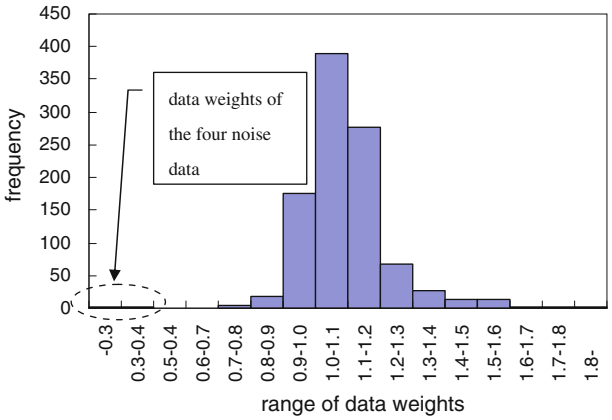

Fig. 9 Data weights of the training data with noise data

The results showed that, using different network architecture and learning parameters, the coefficients of determination of test examples of the optimal models established by MLP, PNN, SLPNN are 0.990, 0.718, and 0.961, respectively, indicating the accuracy of SLPNN only slightly lower than the MLP, while far better than PNN. The histogram of data weights of the 1000 training examples of the optimal SLPNN is shown in Fig. 9. The data weights of the four noise data in the training data are all smaller than 0.4, and they also are the data with the lowest data weights. Judged by the histogram, their data weights are the deviation from the normal range of exceptional values. We can see the learning rule can indeed make the noise data have the lowest data weights, reducing their interference to learning process. Hence, SLPNN has the ability of anti-noise, which is the one of the advantages superior to traditional MLP and PNN.

4 Application examples

In this section, we tested the 15 benchmark problems in the UCI Machine Learning Repository [19] listed in Table 1 to compare the performances of SLPNN, MLP and PNN. In order to test the effectiveness of learning, we used the 10-fold cross-validation, that is, all the data were randomly divided into ten sets. The test was performed ten times, each time a set was held out as testing data while the remaining nine sets were used as the training data. The

Table 1 The descriptions of the data sets

Data set	Input	Output	Data
Iris	4	3	150
Insurance company	7	2	700
Statlog (shuttle)	9	3	5000
Glass identification	9	6	214
Vowel recognition	10	11	990
Wine	13	3	178
Forest cover type	14	7	4000
Letter recognition	16	26	2000
Image segmentation	18	7	2310
Statlog (vehicle silhouettes)	18	4	846
Statlog (German credit)	19	2	1000
Statlog (heart)	20	2	270
Thyroid disease	21	3	7200
Statlog (landsat satellite)	36	6	6000
SPAMBASE	57	2	4500

integrated performance of all the test data was used to assess the accuracy of each network model.

The setting principles of the network frameworks and learning parameters were the same as shown in the preceding section. Besides, we used sampling to reduce the number of data for some data sets with huge number of data, and used simple attributes selection algorithm to reduce the number of attributes for some data sets with huge number of attributes.

In addition, to avoid the influence of the initial connection weights, the error rates are the average of the results of 30 sets of various initial connection weights. To evaluate whether the performance differences between the three kinds of neural networks are significant or not, the t-test was employed. To evaluate the performance of SLPNN, we defined the following hypotheses:

H1: the error rate of SLPNN is smaller than MLP.

H2: the error rate of SLPNN is smaller than PNN.

The significance level is 1%. From the experimental results of SLPNN, MLP, and PNN listed in Table 2, we can see that SLPNN is significantly better than MLP in 11 out of 15 data sets, and is significantly better than PNN in all 15 data sets.

5 Properties of SLPNN

In order to confirm that in the three network parameter which one is the key parameter that makes SLPNN surpass PNN, we will employ the following ways to build models of the above three numerical examples and three application examples (Forest Cover Type, Landsat Satellite, and SPAMBASE). In the SLPNN,

(1) fix variable weight (W) to be 1.0.
(2) fix reciprocal of kernel radius (V) to be 1.0.
(3) fix data weight (h) to be 1.0.

Table 2 The comparisons of the error rates in the application examples

UCI data set	Error rate of neural networks (%)						Significance (1%)	
	MLP		SLPNN		PNN			
	Mean	STD	Mean	STD	Mean	STD	H1	H2
Iris	**2.7**	0.1	4.0	0.1	4.5	0.4		*
Insurance company	36.5	1.35	**33.62**	1.4	35.63	2.21	*	*
Statlog (shuttle)	0.35	0.02	**0.20**	0.02	0.33	0.03	*	*
Glass identification	26.6	0.38	**25.0**	0.41	29.69	1.63	*	*
Vowel recognition	41.1	1.02	**37.66**	0.94	48.48	1.59	*	*
Wine	1.72	0.03	**0.45**	0.03	3.45	0.05	*	*
Forest cover type	23.2	0.3	**15.6**	0.4	25.4	2.1	*	*
Letter recognition	34.6	0.89	**12.6**	0.67	20.0	1.02	*	*
Image segmentation	4.20	0.11	**3.33**	0.11	6.91	0.15	*	*
Statlog (vehicle silhouettes)	**12.8**	0.21	20.92	0.23	26.6	0.32		*
Statlog (German credit)	25.7	0.7	**23.33**	0.43	27.0	0.65	*	*
Statlog (heart)	15.7	0.21	**14.29**	0.19	15.71	0.2	*	*
Thyroid disease	1.95	0.03	**1.66**	0.08	6.48	0.15	*	*
Statlog (landsat satellite)	**5.09**	0.17	7.58	0.12	10.0	0.14		*
SPAMBASE	**3.63**	0.27	9.33	0.17	12.33	0.21		*

The results in Figs. 10 and 11 shows that fixing any network parameters will lower coefficient of determination or raise error rate, that is, reduce the accuracy of the model. Among them, the variable weight has the greatest contribution to model accuracy. The results show that during the process of learning, adjusting the variable weight to change the shapes of transfer functions, and make its contour lines no longer limit to round, but can be elliptic, could build a more accurate model. In addition, even fixing reciprocal of kernel radius to be 1.0 in the algorithm, from Eq. 9, we can see that adjusting the variable weights can make up the adverseness of fixing reciprocal of kernel radius, hence the effects of reciprocal of kernel radius should be small.

6 Conclusions

This study proposed supervised learning probabilistic neural networks (SLPNN) which has three kinds of network parameters: variable weights representing the importance of input variables, the reciprocal of kernel radius representing the effective range of data, and data weights representing the data reliability. These three kinds of parameters can be adjusted through training.

We tested three artificial functions as well as 15 real classification applications, and compared it with multi-layered perceptron (MLP) and probabilistic neural networks (PNN). The results are as follows.

(1) SLPNN is less accurate than MLP, but much more accurate than PNN for the artificial regression problems; SLPNN is more accurate than MLP and much more accurate than PNN for the actual classification applications.

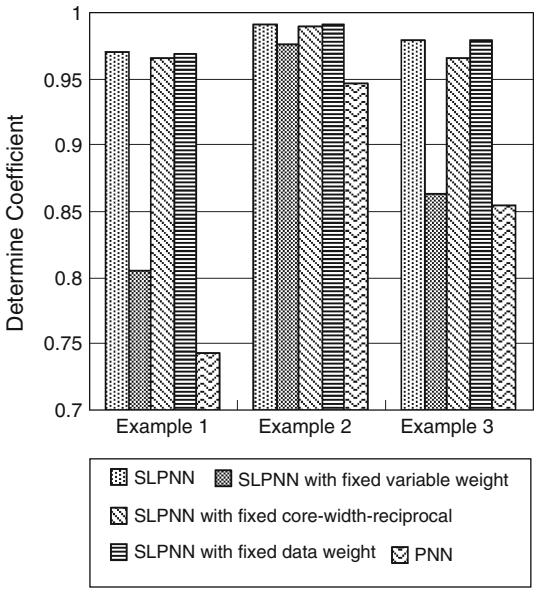

Fig. 10 Effects of network parameters: comparisons of the three numerical examples

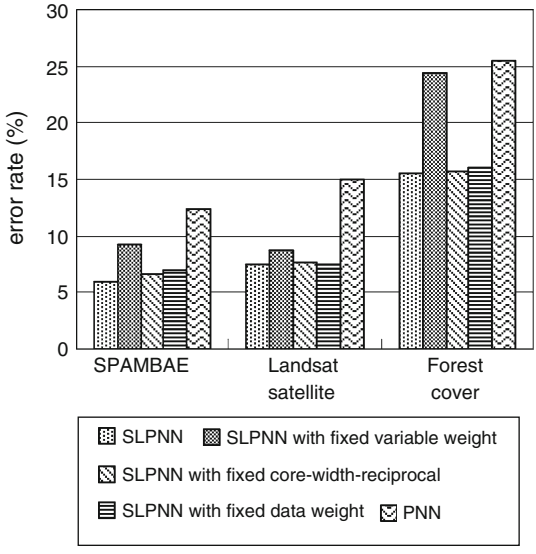

Fig. 11 Effects of network parameters: comparisons of the three application examples

(2) The variable weights can measure the importance of input variables.
(3) The data weights can find the noise data in data set.
(4) The variable weight has the greatest contribution to accuracy of model among the three kinds of network parameters.

References

1. Specht DF (1988) Probabilistic neural networks for classification, or associative memory. In: Proceedings of the 1988 IEEE international conference on neural networks, San Diego, vol 1, pp 525–535
2. Specht DF (1990) Probabilistic neural networks. Neural Netw 3(1):109–118
3. Specht DF, Shapiro PD (1991) Generalization accuracy of probabilistic neural networks compared with back-propagation networks. In: 1991 international joint conference on neural networks Seattle, vol 1, pp 887–892
4. Specht DF (1992) Enhancements to probabilistic neural networks. In: 1992 international joint conference on neural networks Baltimore, vol 1, pp 761–768
5. Patra PK, Nayak M, Nayak SK, Gobbak NK (2002) Probabilistic neural network for pattern classification. In: Proceedings of the 2002 international joint conference on neural networks Honolulu, vol 2, pp 1200–1205
6. Parzen E (1962) On estimation of a probability density function and mode. Annal Math Stat 33(3): 1065–1076
7. Specht DF (1991) A general regression neural network. IEEE Trans Neural Netw 2(6):568–576
8. Adeli H, Panakkat A (2009) A probabilistic neural network for earthquake magnitude prediction. Neural Netw 22(7):1018–1024
9. Song T, Jamshidi MM, Lee RR, Huang M (2007) A modified probabilistic neural network for partial volume segmentation in brain MR image. IEEE Trans Neural Netw 18(5):1424–1432
10. Yu SN, Chen YH (2007) Electrocardiogram beat classification based on wavelet transformation and probabilistic neural network. Pattern Recog Lett 28(10):1142–1150
11. Rutkowski L (2004) Adaptive probabilistic neural networks for pattern classification in time-varying environment. IEEE Trans Neural Netw 15(4):811–827
12. Berthold MR, Diamond J (1998) Constructive training of probabilistic neural networks. Neurocomputing 19(1):167–183
13. Burrascano P (1991) Learning vector quantization for the probabilistic neural network. IEEE Trans Neural Netw 2(4):458–461
14. Villmann T, Schleif FM, Hammer B (2006) Comparison of relevance learning vector quantization with other metric adaptive classification methods. Neural Netw 19(5):610–622
15. Schneider P, Biehl M, Hammer B (2009) Hyperparameter learning in robust soft LVQ. In: ESANN'2009 proceedings, European symposium on artificial neural networks—advances in computational intelligence and learning, Bruges (Belgium), 22–24 April 2009, pp 517–522
16. Seo S, Obermayer K (2003) Soft learning vector quantization. Neural Comput 15(7):1589–1604
17. Galleske I, Castellanos J (2002) Optimization of the kernel functions in a probabilistic neural network analyzing the local pattern distribution. Neural Comput 14(5):1183–1194
18. Montana D (1992) A weighted probabilistic neural network. Adv Neural Inf Process Syst 4:1110–1117
19. Frank A, Asuncion A (2010) UCI machine learning repository http://archive.ics.uci.edu/ml. University of California, School of Information and Computer Science, Irvine, CA