Artif Intell Rev (2013) 39:151–163
DOI 10.1007/s10462-011-9263-5

Intelligent system based on local linear wavelet neural network and recursive least square approach for breast cancer classification

M. R. Senapati · P. K. Dash

Published online: 3 June 2011

Abstract A new learning technique for local linear wavelet neural network (LLWNN) is presented in this paper. The difference of the network with conventional wavelet neural network (WNN) is that the connection weights between the hidden layer and output layer of conventional WNN are replaced by a local linear model. A hybrid training algorithm of Error Back propagation and Recursive Least Square (RLS) is introduced for training the parameters of LLWNN. The variance and centers of LLWNN are updated using back propagation and weights are updated using Recursive Least Square (RLS). Results on extracted breast cancer data from University of Wisconsin Hospital Madison show that the proposed approach is very robust, effective and gives better classification.

Keywords Local linear wavelet neural network · Recursive least square · Gradient descent algorithm · Wisconsin breast cancer (WBC)

1 Introduction

Recently, instead of using common sigmoid activation functions, the wavelet neural network (WNN) employing nonlinear wavelet basis functions (named wavelets), which are localized in both the time space and frequency space, has been developed as an alternative approach to nonlinear fitting problem (Wang et al. 2000; Zhang and Benveniste 1992). Two key problems in designing of WNN are how to determine WNN architecture and what learning algorithm can be effectively used for training the WNN (Chen et al. 2000). These problems are related to determine an optimal WNN architecture, to arrange the windows of wavelets, and to find

M. R. Senapati (✉)
Department of Computer Science and Engineering, Gandhi Engineering College, Biju Patnaik University of Technology, Rourkela, India
e-mail: manas_senapati@sify.com

P. K. Dash
S 'O' A University, Bhubaneswar, Orissa, India
e-mail: dashpk13@yahoo.com

the proper orthogonal or non orthogonal wavelet basis. Curse of dimensionality is a mainly unsolved problem in WNN theory which brings some difficulties in applying a WNN to high dimension problems. The basis function neural networks are a class of neural networks, in which the output of the network is a weighted sum of a number of basis functions. The usually used basis functions include Gaussian radial basis functions (V D Sanchez 1994, 1998; Karayiannis 1999; McGarry et al. 1999; Samantaray et al. 2006), wavelet basis functions Yang et al. (2004), neuro fuzzy basis functions (Chen et al. 2001; Kawaji and Chen 2001). Some have used nature inspired parameter identification techniques (Chen et al. 1999; Kermani et al. 1995; Kennedy et al. 1995; Araujo 2007; Al-Obaidy et al. 2008). The recursive least square (RLS) is a parameter identification technique. Some of the attractive features of the RLS include ease of implementation and the fact that no gradient information is required. It has a faster rate of convergence compared to gradient search and least mean square.

Pattern Classification is an important research and application area. Much effort has been devoted over the past several decades to the development and improvement of Pattern Classification models. This paper, a local linear wavelet neural network (LLWNN) extends the application of Chen et al. (2006, 2004) and is proposed for breast cancer detection, in which the connection weights between the hidden layer units and output units are replaced by a local linear model. The usually used learning algorithm for WNN is gradient descent method. But its disadvantages are slow convergence speed and easy stay at local minimum. A combination approach of RLS with adaptive diversity learning and gradient descent method is proposed for training the LLWNN. Simulation results for Pattern Classification problems show the effectiveness of the proposed method. The main contributions of this paper are (1) the LLWNN providing a more parsimonious interpolation in high-dimension spaces when modeling samples are sparse; (2) a novel hybrid training algorithm for LLWNN was proposed. The paper is organized as follows. The LLWNN is introduced in Sect. 2. The RLS learning algorithm for training LLWNN is described in Sect. 3. Learning Algorithm using Kalman Filter (http//ww.cs.unc.edu/~welch) (Sum et al. 1999) is explained in Sect. 4. The experimental results on Pattern Classification for Wisconsin Breast Cancer (WBC) problem is given in Sect. 5. Finally, concluding remarks are derived in the last section.

2 Local linear wavelet neural network

In terms of wavelet transformation theory, wavelets in the following form:

$$\begin{aligned}
\psi &= \left\{ \psi i = |a_i|\, \psi\left(\frac{x - b_i}{a_i}\right) : a_i, b_i \in R^n, i \in Z \right\}, \\
X &= (x_1, x_2, \ldots x_n), \\
a_i &= (a_{i1}, a_{i2}, \ldots a_{in}), \\
b_i &= (b_{i1}, b_{i2}, \ldots b_{in}),
\end{aligned}$$

are a family of functions generated from one single function $\Psi(x)$ by the operation of dilation and translation. $\Psi(x)$, which is localized in both the time space and the frequency space, is called a mother wavelet and the parameters ai and bi are named the scale and translation parameters, respectively. The **x** represents inputs to the WNN model.

In the standard form of WNN, the output of a WNN is given by

$$f(x) = \sum_{i=1}^{M} \omega_i \psi i(x) = \sum_{i=1}^{M} \omega_i \, |a_i|^{-1/2} \, \psi\left(\frac{x - b_i}{a_i}\right), \tag{1}$$

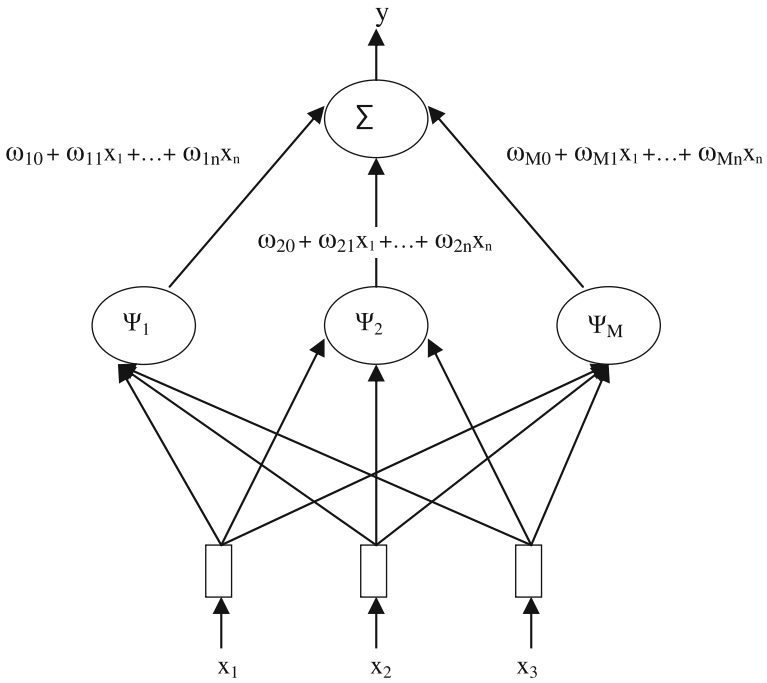

Fig. 1 A local linear wavelet neural network

where Ψi is the wavelet activation function of i^{th} unit of the hidden layer and ω_i is the weight connecting the i^{th} unit of the hidden layer to the output layer unit. Note that for the n-dimensional input space, the multivariate wavelet basis function can be calculated by the tensor product of n single wavelet basis functions as follows

$$\psi(x) = \prod_{i=1}^{n} \psi(x_i) \tag{2}$$

Obviously, the localization of the i^{th} units of the hidden layer is determined by the scale parameter ai and the translation parameter bi. According to the previous researches, the two parameters can either be predetermined based upon the wavelet transformation theory or be determined by a training algorithm. Note that the above WNN is a kind of basis function neural network in the sense of that the wavelets consists of the basis functions. Note that an intrinsic feature of the basis function networks is the localized activation of the hidden layer units, so that the connection weights associated with the units can be viewed as locally accurate piecewise constant models whose validity for a given input is indicated by the activation functions. Compared to the multilayer perceptron neural network, this local capacity provides some advantages such as the learning efficiency and the structure transparency. However, the problem of basis function networks is also led by it. Due to the crudeness of the local approximation, a large number of basis function units have to be employed to approximate a given system. A shortcoming of the WNN is that for higher dimensional problems many hidden layer units are needed.

In order to take advantage of the local capacity of the wavelet basis functions while not having too many hidden units, here we propose an alternative type of WNN. The

architecture of the proposed LLWNN is shown in Fig. 1. Its output in the output layer is given by

$$y = \sum_{i=1}^{M} (\omega_{i0} + \omega_{i1}x_1 + \cdots + \omega_{in}x_n)\psi i(x)$$
$$= \sum_{i=1}^{M} (\omega_{i0} + \omega_{i1}x_1 + \cdots + \omega_{in}x_n)\,|ai|^{-1/2}\,\psi\left(\frac{x - b_i}{a_i}\right), \tag{3}$$

where X = [x$_1$, x$_2$, . . . , x$_n$] Instead of the straightforward weight ω_i (piecewise constant model), a linear model is introduced.

$$\text{vi} = \omega_{i0} + \omega_{i1}x_1 + \cdots + \omega_{in}x_n \tag{4}$$

The activities of the linear models vi (I = 1, 2, . . . , M) are determined by the associated locally active wavelet functions $\psi i(x)$ (I = 1, 2, . . . , M) thus vi is only locally significant. The motivations for introducing the local linear models into a WNN are as follows: (1) local linear models have been studied in some neuro-fuzzy systems and shown good performances (Chen et al. 2001; Kawaji and Chen 2001); and (2) local linear models should provide a more parsimonious interpolation in high-dimension spaces when modelling samples are sparse. The scale and translation parameters and local linear model parameters are randomly initialized at the beginning and are optimized by recursive least square algorithm discussed in the following section.

3 Recursive least square

The recursive least square (RLS) is a parameter identification technique. In RLS algorithm there are two variables involved in the recursions (those with time index n- 1): $\hat{w}(i-1)$, Pi$-_1$. We must provide initial values for these variables in order to start the recursions:

$$\cdot w(0)$$

If we have some apriori information about the parameters $\hat{w}$this information will be used to initialize the algorithm. Otherwise, the typical initialization is

$$w(0) = 0$$
$$\cdot \text{P}(0)$$
$$P(i) = \left[\sum_{n=1}^{i} \lambda^{i-1}\psi(n)\psi(n)^T\right]^{-1}$$

the exact initialization of the recursions uses a small initial segment of the data ψ(i$_1$); $\psi(\text{i}_1 + 1) \ldots, \psi(0)$ to compute

$$P(0) = \left[\sum_{n=1}^{o} \lambda^{-1}\psi(n)\psi(n)^T\right]^{-1} \tag{5}$$

All the necessary equations to form the RLS algorithm are

$$\text{k(i)} = \frac{\text{P (i} - 1)\,\phi\text{T(i)}}{\lambda + \text{P (i} - 1)\,\phi\text{T(i)}} \tag{6}$$

$$\text{w(j)} = \text{wj (i} - 1) + \text{k(i)}\left[\text{dj(i)} - \text{wj (i} - 1)\,\phi\text{T(i)}\right] \tag{7}$$

$$P(i) = \frac{1}{\lambda}\left[P(i-1) - k(i)\phi(i)P(i-1)\right] \tag{8}$$

where λ is real number between 0 and 1, $P(0) = a^{-1}I$, and a is a small positive number and $wj(0) = 0$.

Particle swarm optimization was used to find the initial centers of the clusters. The centers as well as spreads were updated using Error back propagation. Recursive Least Square approach was used to update the weights associated between the hidden layer and the output layer.

4 The discrete kalman filter

The Kalman filter addresses the general problem of trying to estimate the state $x_k^- \in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_k = Ax_{k\_l} + Bu_{k\_l} + w_{k\_l}, \tag{9}$$

with a measurement $z \in \Re^m$ that is

$$z_k = Hx_k + \nu_k \tag{10}$$

The random variables w_k and ν_k represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distributions

$$p(w) \sim N(0, Q), \tag{11}$$

$$p(v) \sim N(0, R). \tag{12}$$

In practice, the process noise covariance Q and measurement noise covariance R matrices might change with each time step or measurement, however here we assume they are constant.

The n x n matrix A in the difference Eq. (9) relates the state at the previous time step k-l to the state at the current step k, in the absence of either a driving function or process noise. Note that in practice A might change with each time step, but here we assume it is constant. The n x l matrix B relates the optional control input $u \in \Re^l$ to the state x. The m x n matrix H in the measurement Eq. (10) relates the state to the measurement z_k. In practice H might change with each time step or measurement, but here we assume it is constant.

4.1 The computational origins of the filter

We define $\hat{x}_k \in \Re^n$ to be our a priori state estimate at step k given knowledge of the process prior to step k, and $\hat{x}_k \in \Re^n$ to be our a posteriori state estimate at step k given measurement z_k, can then define a priori and a posteriori estimate errors as

$$\begin{aligned} e_k^- &\equiv x_k - \hat{x}_k^- \\ e_k &\equiv x_k - \hat{x}_k \end{aligned} \tag{13}$$

The a priori estimate error covariance is then

$$P_k = E\left[e_k - e_k^{-T}\right], \tag{14}$$

and the a posteriori estimate error covariance is

$$P_k = E\left[e_k e_K^{-T}\right]. \tag{15}$$

In deriving the equations for the Kalman filter, we begin with the goal of finding an equation that computes an a posteriori state estimate x_k as a linear combination of an a priori estimate x_k and a weighted difference between an actual measurement z_k and a measurement prediction Hx_k as shown below in (16). Some justification for (16) is given in "The Probabilistic Origins of the Filter" found below.

$$\hat{x}_k = \hat{x}_k^- + K\left(z_k - H\hat{x}_k^-\right) \tag{16}$$

The difference $(z_k - H\hat{x}_k^-)$ in (16) is called the measurement innovation, or the residual. The residual reflects the discrepancy between the predicted measurement $H\hat{x}_k^-$ and the actual measurement z_k. A residual of zero means that the two are in complete agreement. The n x m matrix K in (16) is chosen to be the gain or blending factor that minimizes the a posteriori error covariance (15). This minimization can be accomplished by first substituting (16) into the above definition for e_k substituting that into (15), performing the indicated expectations, taking the derivative of the trace of the result with respect to K, setting that result equal to zero, and then solving for K. One form of the resulting K that minimizes (15) is given by

$$\begin{aligned} K_k &= P_k^- H^T \left(HP_k^- H^T + R\right)^- \\ &= \frac{P_k^- H^T}{HP_k^- H^T + R} \end{aligned} \tag{17}$$

Looking at (17) we see that as the measurement error covariance R approaches zero, the gain K weights the residual more heavily. Specifically,

$$\lim_{R_k \to 0} K_k = H^{-1}$$

On the other hand, as the a priori estimate error covariance P_k^- approaches zero, the gain K weights the residual less heavily. Specifically,

$$\lim_{R_k \to 0} K_k = 0$$

Another way of thinking about the weighting by K is that as the measurement error covariance R approaches zero, the actual measurement z_k is "trusted" more and more, while the predicted measurement $H\hat{x}_k^-$ is trusted less and less. On the other hand, as the a priori estimate error covariance P_k^- approaches zero the actual measurement z_k is trusted less and less, while the predicted measurement $H\hat{x}_k^-$ is trusted more and more.

4.2 The probabilistic origin of the filter

The justification is rooted in the probability of the a priori estimate $\hat{x}_k^-$ conditioned on all prior measurements z_k (Bayes' rule). For now let it suffice to point out that the Kalman filter maintains the first two moments of the state distribution,

$$\begin{aligned} E[x_k] &= \hat{x}_k \\ E\left[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right] &= P_k. \end{aligned} \tag{18}$$

The a posteriori state estimate (16) reflects the mean (the first moment) of the state distribution—it is normally distributed if the conditions of (11) and (12) are met. The a posteriori

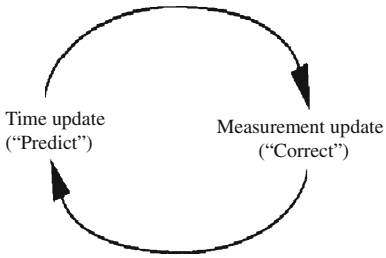

Fig. 2 The ongoing discrete Kalman filter cycle

estimate error covariance (15) reflects the variance of the state distribution (the second non-central moment). In other words,

$$p(x_k|z_k) \sim N(E[x_k], E\left[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right] = N(\hat{x}_k, P_k) \tag{19}$$

4.2.1 The discrete kalman filter algorithm

We will begin this section with a broad overview, covering the "high-level" operation of one form of the discrete Kalman filter. After presenting this high-level view, we will narrow the focus to the specific equations and their use in this version of the filter.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations. Indeed the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems as shown below in Fig. 2.

The time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement at that time.

The specific equations for the time and measurement updates are presented below

Discrete Kalman filter time update equations

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{20}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{21}$$

Again notice how the time update equations given below project the state and covariance estimates forward from time step k − 1 to step k. A and B are from (9), while Q is from (10). Initial conditions for the filter are discussed in the earlier references.

Discrete Kalman filter measurement updates equations.

$$K_k = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1} \tag{22}$$

$$\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H\hat{x}_k^-\right) \tag{23}$$

$$P_k = (I - K_k H)P^- \tag{24}$$

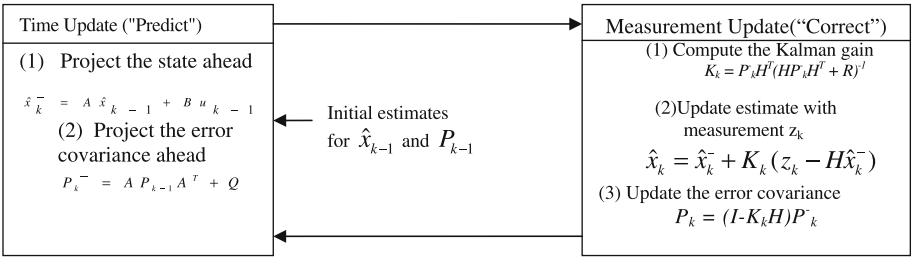

Fig. 3 A complete picture of the operation of the Kalman filter

The first task during the measurement update is to compute the Kalman gain, K_k. Notice that the equation given here as (22) is the same as (17). The next step is to actually measure the process to obtain z_k, and then to generate an a posteriori state estimate by incorporating the measurement as in (23). Again (23) is simply (16) repeated here for completeness. The final step is to obtain an a posteriori error covariance estimate via (24).

After each time and measurement update pair, the process is repeated with the previous a posteriori estimates used to project or predict the new a priori estimates. This recursive nature is one of the very appealing features of the Kalman filter—it makes practical implementations much more feasible. The Kalman filter instead recursively conditions the current estimate on all of the past measurements. Figure 3 below offers a complete picture of the operation of the filter, combining the high-level diagram of Fig. 2 along with the equations from Eq. 20 to Eq. 24.

4.3 Filter parameters and tuning

In the actual implementation of the filter, the measurement noise covariance R is usually measured prior to operation of the filter. Measuring the measurement error covariance R is generally practical (possible) because we need to be able to measure the process anyway (while operating the filter) so we should generally be able to take some off-line sample measurements in order to determine the variance of the measurement noise.

The determination of the process noise covariance Q is generally more difficult as we typically do not have the ability to directly observe the process we are estimating. Sometimes a relatively simple (poor) process model can produce acceptable results if one "injects" enough uncertainty into the process via the selection of Q. Certainly in this case one would hope that the process measurements are reliable.

In either case, whether or not we have a rational basis for choosing the parameters, often times superior filter performance (statistically speaking) can be obtained by tuning the filter parameters Q and R. The tuning is usually performed off-line, frequently with the help of another (distinct) Kalman filter in a process generally referred to system identification.

We note that under conditions where Q and R are in fact constant, both the estimation error covariance P_k and the Kalman gain K_k will stabilize quickly and then remain constant (see the filter update equations in Fig. 3). If this is the case, these parameters can be pre-computed by either running the filter off-line, or for example by determining the steady-state value of P_k.

It is frequently the case however that the measurement error (in particular) does not remain constant. For example, when sighting beacons in our optoelectronic tracker ceiling panels, the

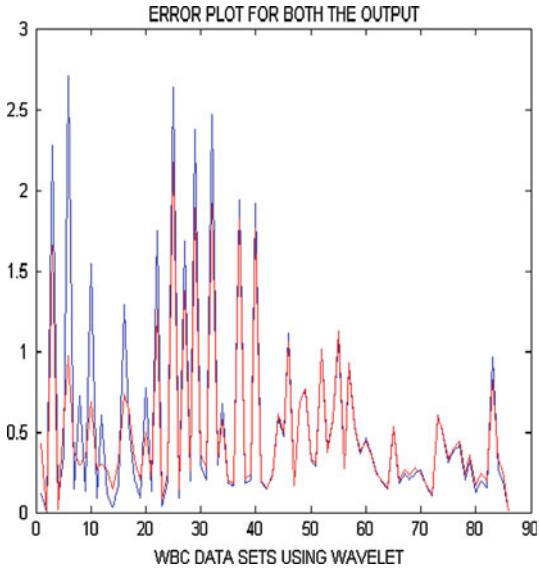

Fig. 4 Error plot of LLWNN

noise in measurements of nearby beacons will be smaller than that in far-away beacons. Also, the process noise Q is sometimes changed dynamically during filter operation—becoming Q_k—in order to adjust to different dynamics. For example, in the case of tracking the head of a user of a 3D virtual environment we might reduce the magnitude of Q_k if the user seems to be moving slowly, and increase the magnitude if the dynamics start changing rapidly. In such cases Q_k might be chosen to account for both uncertainties about the user's intentions and uncertainty in the model.

5 Discussion

In order to evaluate the performance of the rule extraction algorithm, we carried out a two fold experiment with WBC data set (http://ailab.si/orange/doc/datasets/breast-cancer-wisconsin-cont.htm). The results show that the percentage of classification, Table 11 is better if LLWNN is being trained by RLS as compared to other training methods. The algorithms associated to the optimization and rule extraction method were simulated using MATLAB v6.5.

WBC database: The WBC training data contains 400 exemplars and the test set containing 299 exemplars for a total of 699 exemplars. The input data were normalized by replacing each feature value x by $x = (x - \mu x)/\sigma x$ where μx and σx denote the sample mean and standard deviation of this feature over the entire data set. The networks are trained to respond with the target value $y_{ik} = 1$, and $y_{jk} = 0 \forall j \neq i$, when presented with an input vector x_k from the i^{th} category.

The MATLAB m-files were used to generate the simulation results presented in this section. The training algorithms were initialized with prototype vectors randomly selected from the input data on a two fold basis and with the weight matrix W set to 1 and σ initialized to 1. For the sake of simplicity and space we have drawn four figures (Figs. 4, 5, 6, 7) and eleven Tables to verify the results.

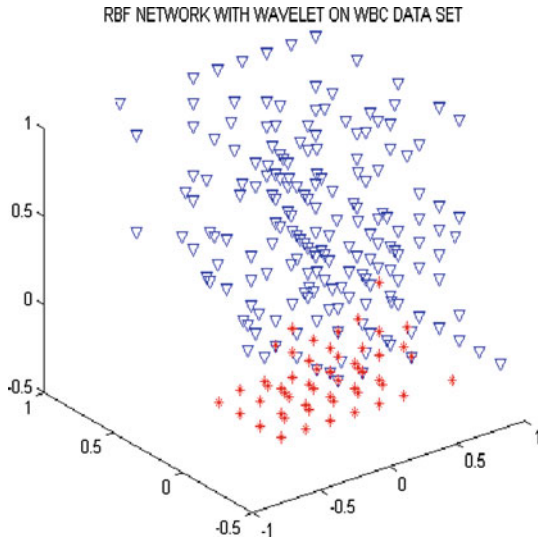

Fig. 5 Classification using LLWNN

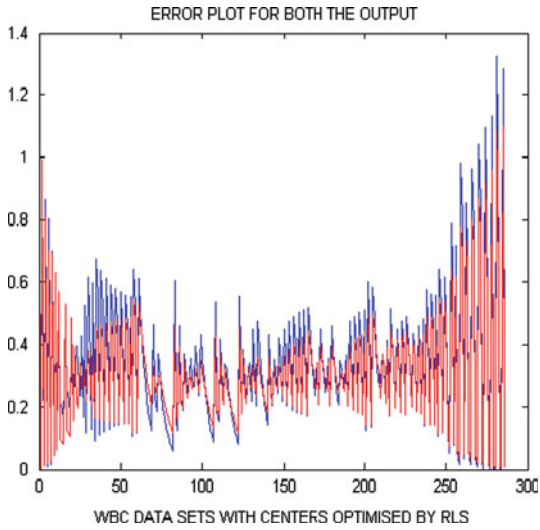

Fig. 6 Error plot of RBFNN and RLS

5.1 Simulation results

Tabular data Figures 4 and 6 shows the error plot of LLWNN and RBFNN. Figures 5 and 7 shows the classification obtained from LLWNN and RBFN. Tables 1, 2, 3, and 4, shows the weights, centers, spreads, scale parameter and translation parameter obtained from LLWNN.

Tables 5, 6 and 7 shows the centers, weights and spreads obtained by training RBFNN. Tables 8, 9 and 10 shows the centers, weights and spreads obtained by training RBFNN. The centers and spreads of all the technique were updated using error back propagation. The weights of LLWNN were updated using RLS. The weights of RBFNN were updated using RLS and Kalman Filter. In Table 11 the percentage of classification of each technique is shown.

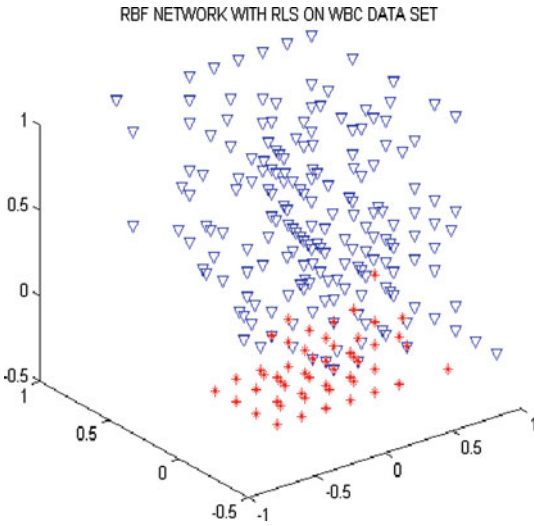

Fig. 7 Classification using RBFNN and RLS

Table 1 Centers obtained from LLWNN

		7.7888	7.7888	7.7888	7.7888	7.7888
WBC		7.7888	7.7888	7.7888	7.7888	
		13.0746	13.0746	13.0746	13.0746	13.0746
		13.0746	13.0746	13.0746	13.0746	

Table 2 Weights obtained from LLWNN

		3.2986	3.2986	3.2986	3.2986	3.2986
WBC		3.2986	3.2986	3.2986	3.2986	
		2.3088	2.3088	2.3088	2.3088	2.3088
		2.3088	2.3088	2.3088	2.3088	

Table 3 Spreads obtained from LLWNN

2.0997					
1.2149					

Table 4 Scale parameter a and b updated using RLS

		a	31.8562
WBC			74.6344
		b	31.8562
			77.2647

Rule for classification of WBC data sets using LLWNN

if (oo(r, 1) $\geq$ 0.06 & oo(r, 1) $\leq$ 0.4627) & (oo(r, 2) $\geq$ 0.0611 & oo(r, 2) $\leq$ 0.4197)then Benign;

if (oo(r, 1) ≥ -3.7718 & oo(r, 1) $\leq$ 0.0394) & (oo(r, 2) ≥ -3.2997 & oo(r, 2) $\leq$ 0.036) then Malignant;

Rule for classification of WBC data sets using RLS

Table 5 Centers obtained from RBFNN trained by RLS

	1.2352	1.2352	1.2352	1.2352	1.2352
	1.2352	1.2352	1.2352	1.2352	
WBC	1.6426	1.6426	1.6426	1.6426	1.6426
	1.6426	1.6426	1.6426	1.6426	

Table 6 Weights obtained from RBFNN trained by RLS

1.6426	1.6426
5.7481	5.7481

Table 7 Spreads obtained from RBFNN trained by RLS

7.8622
8.6432

Table 8 Centers obtained from RBFNN trained by Kalman filter

	1.1905	1.1905	1.1905	1.1905	1.1905
	1.1905	1.1905	1.1905	1.1905	
WBC	1.0019	1.0019	1.0019	1.0019	1.0019
	1.0019	1.0019	1.0019	1.0019	

Table 9 Weights obtained from RBFNN trained by Kalman filter

0.9849	0.9981
1.1905	1.0019

Table 10 Spreads obtained from RBFNN trained by Kalman filter

7.8622
8.6432

Table 11 Percentage of classification

Technique	% of classification
LLWNN with RLS	97.2818
RBFNN with Kalman filter	96.4235
RBFNN with RLS	97.1388

if (oo(r, 1) $\geq$ 8.5288 & oo(r, 1) $\leq$ 8.9768) & (oo(r, 2) $\geq$ 13.2243 & oo(r, 2) $\leq$ 13.8661) then Benign;

if (oo(r, 1) $\geq$ 8.9937 & oo(r, 1) $\leq$ 10.6680) & (oo(r, 2) $\geq$ 13.8850 & oo(r, 2) $\leq$ 16.2742) then Malignant;

Rule for classification of WBC data sets using Kalman Filter

if (oo(r, 1) $\geq$ 1.6277 & oo(r, 1) $\leq$ 1.7129) & (oo(r, 2) $\geq$ 1.5870 & oo(r, 2) $\leq$ 1.6614) then Benign;

if (oo(r, 1) $\geq$ 1.7163 & oo(r, 1) $\leq$ 2.0338) & (oo(r, 2) $\geq$ 1.6651 & oo(r, 2) $\leq$ 1.9274) then Malignant;

6 Conclusion

The success of neural network architecture depends heavily on the availability of effective learning algorithms. The object of this study is to examine the effectiveness of RLS for training LLWNN. The theoretical strength of the LLWNN yet to be used in hundreds of technologies, and this paper demonstrates that training LLWNN using RLS is yet another fruitful application of LLWNN. Our simulation using MATLAB v6.5 verifies that weight optimization of LLWNN through RLS provides better performance as compared to optimizing the weights of RBFNN through RLS and Kalman Filter, Table 11. Further research could focus on the application of LLWNN trained with alternative forms of the generator function. This technique can be applied to large problems to obtain experimental verification of the computational saving of time can be included as a future work.

References

Al-Obaidy M, Ayesh A, Sheta AF (2008) Optimizing the communication distance of an ad hoc wireless sensor networks by genetic algorithms. Artif Intell Rev 29(3–4):183–194. doi:10.1007/s10462-009-9148-z

An Introduction to Kalman Filter by Welch and Bishop, Welch@cs.unc.edu, http://ww.cs.unc.edu/~welch

Araujo L (2007) How evolutionary algorithms are applied to statistical natural language processing. Artif Intell Rev 28(4):275–303. doi:10.1007/s10462-009-9104-y

Chen S, Wu Y, Luk B (1999) Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks. IEEE Trans Neural Netw 10(5):1239–1243

Chen YH et al (2000) Evolving wavelet neural networks for system identification. In: Proceeding of international conference on electrical engineering, pp 279–282

Chen YH et al (2001) Evolving the basis function neural networks for system identification. Int J Adv Comput Intell 5(4):229–238

Chen Y, Dong J, Yang B, Zhang Y (2004) Local linear wavelet neural network. Fifth world congress on intelligent control and automation (WCIA), Hangzhou, pp 1954–1957

Chen Y, Yang Bo, Dong J (2006) Time series prediction using a local linear wavelet neural network. Neuro Comput 69:449–465

Karayiannis N (1999) Reformulated radial basis neural networks trained by gradient descent. IEEE Trans Neural Netw 10(3):657–671

Kawaji S, Chen YH (2001) Evolving neuro fuzzy system by hybrid soft computing approaches for system identification. Int J Adv Comput Intel 5(4):229–238

Kennedy J, Eberhart RC, Shi Y (1995) Particle swarm optimization. Proc IEEE Int J Conf Neural Netw 4:1942–1948

Kermani BG, White MW, Nagle HT (1995) Feature extraction by genetic algorithm for neural networks in breast cancer classification. In: Engineering in Medicine and Biology Society, 17th Annual Conference, IEEE, vol 1, pp 831–832, doi:10.1109/IEMBS.1995.575385

McGarry K, Tait J, Wermter S (1999) Rule extraction from radial basis function networks. Proceedings of IEEE conference on Artificial Neural Networks, University of Edinburg, UK, September 7–10, pp 613–618

Samantaray SK, Dash PK, Panda G (2006) Fault classification and location using HS_transform and radial basis function neural network. Electrical power system research. Elsevier, pp 897–905

Sum J, Leung C-s, Young GH, Kan W-k (1999) Kalman filtering method in neural-network, training and pruning. IEEE Trans Neural Netw 10(1):161–166

V D Sanchez A (ed) (1994) Special issue on Backpropagation, Vol 6, Part II, Neurocomputing

V D Sanchez A (ed) (1998) Special issue on RBF network, Vol 19, Part I, Neurocomputing

Wang T et al (2000) A wavelet neural network for the approximation of nonlinear multivariable function. Trans Inst Electron Eng C 102-C:185–193

Yang X, Pang G, Yung N (2004) Discriminative training approaches to fabric defect classification based on wavelet transform. Pattern Recogn 37:889–899

Zhang Q, Benveniste A (1992) Wavelet Networks. IEEE Trans Neural Netw 3(6):889–898