Expert Systems With Applications 49 (2016) 112–122

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A linear model based on Kalman filter for improving neural network classification performance

Joko Siswantoro[a,b,*], Anton Satria Prabuwono[a,c], Azizi Abdullah[a], Bahari Idrus[a]

[a] Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM, Bangi, Selangor D. E., Malaysia
[b] Faculty of Engineering, University of Surabaya, Jl. Kali Rungkut, Surabaya 60293, Indonesia
[c] Faculty of Computing and Information Technology, King Abdulaziz University, Rabigh 21911, Saudi Arabia

ARTICLE INFO

Keywords:
Neural network
Linear model
Kalman filter
Classification performance

ABSTRACT

Neural network has been applied in several classification problems such as in medical diagnosis, handwriting recognition, and product inspection, with a good classification performance. The performance of a neural network is characterized by the neural network's structure, transfer function, and learning algorithm. However, a neural network classifier tends to be weak if it uses an inappropriate structure. The neural network's structure depends on the complexity of the relationship between the input and the output. There are no exact rules that can be used to determine the neural network's structure. Therefore, studies in improving neural network classification performance without changing the neural network's structure is a challenging issue. This paper proposes a method to improve neural network classification performance by constructing a linear model based on the Kalman filter as a post processing. The linear model transforms the predicted output of the neural network to a value close to the desired output by using the linear combination of the object features and the predicted output. This simple transformation will reduce the error of neural network and improve classification performance. The Kalman filter iteration is used to estimate the parameters of the linear model. Five datasets from various domains with various characteristics, such as attribute types, the number of attributes, the number of samples, and the number of classes, were used for empirical validation. The validation results show that the linear model based on the Kalman filter can improve the performance of the original neural network.

1. Introduction

The classification problem is the problem of assigning an object into one of predefined classes based on a number of features or attributes extracted from the object (Zhang, 2000). In machine learning, classification is categorized as a supervised learning method. A classifier is constructed based on a training set with known class labels (Alpaydin, 2010). Classification problems occur in various real world problems, including problems in character recognition (Gao & Liu, 2008), face recognition (Zhifeng, Dahua, & Xiaoou, 2009), speech recognition (Chandaka, Chatterjee, & Munshi, 2009), biometrics (Lyle, Miller, Pundlik, & Woodard, 2012), medical diagnosis (Akay, 2009; Mazurowski et al., 2008; Verma & Zhang, 2007), industry (Jamil, Mohamed, & Abdullah, 2009; Kılıç, Boyacı, Köksel, & Küsmenoğlu, 2007; Nashat, Abdullah, & Abdullah, 2014; Rocha, Hauagge, Wainer, & Goldenstein, 2010), business (Chen & Huang, 2003; Huang, Chen, & Wang, 2007; Min & Lee, 2005), and science (Evett & Spiehler, 1987; Sigillito, Wing, Hutton, & Baker., 1989). Several classification algorithms have been proposed to solve classification problems, namely decision tree (Quinlan, 1986), linear discriminant analysis (Li & Yuan, 2005), Bayesian classifier (Domingos & Pazzani, 1997), rule-based classifier (Clark & Niblett, 1989), neural network (Lippmann, 1987), *k*-nearest neighbor (Cover & Hart, 1967), and support vector machine (Cortes & Vapnik, 1995).

Artificial neural network or simply neural network is a computational model inspired by the biological nervous system. Neural network is a nonlinear model, which is very simple in computation and has the capability to solve complex real problems including prediction and classification. Neural network has appears to be a significant classification method and an alternative to

* Corresponding author at: Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM, Bangi, Selangor D. E., Malaysia. Tel.: +601121181700, +6283849409952.
E-mail addresses: joko_siswantoro@ubaya.ac.id, joko.siswantoro@gmail.com (J. Siswantoro), antonsatria@eu4m.eu (A.S. Prabuwono), azizi@ukm.edu.my (A. Abdullah), bahari@ukm.edu.my (B. Idrus).

http://dx.doi.org/10.1016/j.eswa.2015.12.012
0957-4174/

conventional classification methods (Zhang, 2000). It has been applied in various prediction and classification problems, such as bankruptcy prediction (Tsai & Wu, 2008), handwriting recognition (Goh, Mital, & Babri, 1997), product inspection (Kılıç et al., 2007), medical diagnosis (Mazurowski et al., 2008), and transportation (Garrido, de Oña, & de Oña, 2014).

The performance of a neural network is characterized by its structure, transfer function, and learning algorithm (Lippmann, 1987). The structure of a neural network depends on the number of hidden layers and the number of neurons in each hidden layer. However, there is no exact rule to determine the structure of a neural network. Generally, the more complex the relationship between the input data and the desired output, the more complex the structure of the neural network used in classification (Du & Sun, 2008). Therefore, a neural network classifier tends to be a weak classifier if it uses a structure that has an inappropriate number of hidden layers or an inappropriate number of neurons in its hidden layers. Although research on neural network classifiers has been widely conducted with significant results, it is still a challenging task, especially in research related to improving classification performance.

The ensemble method is a well-known method to improve the classification performance of a neural network by combining a series of trained neural networks (Giacinto & Roli, 2001; Glodek, Reuter, Schels, Dietmayer, & Schwenker, 2013; Zaamout & Zhang, 2012). However, if the outputs of each neural network are biased or correlated, then there is no guarantee that ensemble can improve the classification performance of the neural network (Zhang, 2000). Feature selection is another issue in improving classification performance. Feature selection aims to find a subset of features that achieves maximum classification performance and reduces computation effort. Various feature selection methods have been developed for neural network classifiers. One such method has used a genetic algorithm to select salient features (Li, 2006; Verma & Zhang, 2007). However, employing feature selection on a neural network classifier does not always improve classification performance, as reported in T.-S. Li (2006).

Improving classification performance is a promising issue, not only for neural network classifiers but also for other classifiers. Rocha et al. (2010) proposed classifier fusion for improving fruit and vegetable classification accuracy. They employed a combination of fusion of binary classifiers and a very long feature descriptor, including global color histogram (GCH), Unser's descriptors, color coherence vectors (CCVs), Border/Interior pixel Classification (BIC), and appearance descriptors. Although high classification accuracy is achieved, it takes significant time to perform the training stage. Mastrogiannis, Boutsinas, and Giannikos (2009) proposed the use of the ELECTRE methods concepts to improve the accuracy of data mining classification algorithms. Even if the proposed method can improve classification accuracy of several data mining algorithms, it can be applied to classify only categorical objects. Hacibeyoglu, Arslan, and Kahramanli (2011) analyzed the effect of discretization on classification. This method used entropy-based discretization to transform continuous-valued features into integer-valued features. Therefore, it cannot be applied to classify objects with only categorical- or integer-valued features.

In recent years, several authors tried to combine several techniques to improve classification performance. Farid, Zhang, Rahman, Hossain, and Strachan (2014) have proposed two hybrid algorithms of decision tree (DT) and naïve Bayes (NB) classifiers for multi-class classification. The first algorithm used NB to remove misclassified instances from training dataset before used to build DT. The second algorithm used DT to find a subset of attributes that play important roles in classification. Selected attributes by DT were then used for classification using NB. Seera and Lim (2014) have used Fuzzy Min–Max (FMM) neural network, classification and regression tree (CART), and random forest (RF) model to develop a hybrid intelligent system for medical data classification. FMM neural network was used to generate hyperbox fuzzy set. The generated hyperbox was then used to build CART. Finally, to increase classification performance an ensemble of CART was constructed using RF. Affonso, Sassi, and Barreiros (2015) have combined rough sets theory and fuzzy neural network for biological image classification. They used rough sets theory for feature selection. The selected features were used to train a multilayer perceptron neuro fuzzy network. Onan (2015) have proposed the combination of instance selection, feature selection, and fuzzy-rough nearest neighbor for automated diagnosis of breast cancer. Fuzzy-rough instance selection method was used to remove useless or erroneous instances from dataset, while consistency-based feature selection method and a re-ranking algorithm were used to select important feature. Pruengkarn, Chun Che, and Kok Wai (2015) have used clustering technique, feature selection, and ensemble of classifier to improve classification performance. Clustering technique was employed to separate dataset into misclassification dataset and clean dataset. The clean dataset was classified using a common classifier including DT, NB, ANN, and SVM. Whereas feature selection technique based on fuzzy C-means and ensemble of classifier using majority voting were used to classify misclassification dataset. Although all authors reported achieving high classification accuracy, they did not report the computing time for the proposed methods.

A neural network classifier achieves high classification accuracy when its predicted output is very close to its desired output. Therefore, to increase neural network classification accuracy, the use of a transformation that transforms the predicted output of a neural network to a value close to the desired output can be considered as a post processing. A linear model is a simple transformation that can be used to achieve such a purpose. The linear model consists of independent (input) variables, dependent (output) variables, and unknown parameters. The parameters of a linear model need to be estimated such that the error between the predicted output and the desired output is minimized. The Kalman filter (Kalman, 1960) is a method that can be used to estimate the parameters of a linear model. The Kalman filter is a recursive method for fitting a linear model to a given dataset such that the sum of square error is minimized without performing matrix inversion as in ordinary least square. Even if the model has a number of variables greater than the number of dataset elements, the Kalman filter can still calculate the estimate (Wu, Rutan, Baldovin, & Massart, 1996).

This paper proposes a method to improve neural network classification performance by constructing a linear model based on the Kalman filter. The proposed method uses the Kalman filter iteration to estimate the parameters of a linear model. The model uses the linear combination of object features and predicted outputs of a neural network as input variables to predict class labels. As in a neural network, the model can use any type of variables as input. Therefore, the model would improve neural network classification performance without considering the types of object features.

The rest of the paper is organized as follows. Sections 2 and 3 provide a brief explanation about the structure of neural network and Kalman filter, respectively. Section 4 explains the proposed method. Section 5 describes datasets and method used for validation. Section 6 presents experimental results and discussion. And finally, conclusion and future work are provided in Section 7.

2. Neural network

The neural network model consists of interconnected neurons with weights, arranged in layers. The structure of a neuron consists of inputs $p_1, p_2, \ldots, p_n$, weights $w_1, w_2, \ldots, w_n$, bias b,

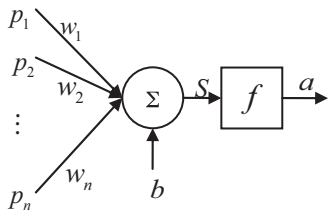

Fig. 1. The structure of a neuron: p_1, p_2, … p_n are inputs, w_1, w_2, … w_n are weights, b is bias, S is the sum of weighted inputs and bias, f is transfer function, and a is output.

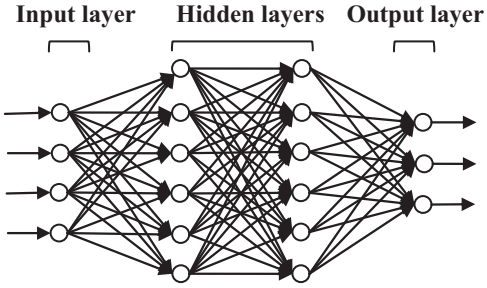

Fig. 2. The general topology of a neural network consists of an input layer, two hidden layers, and an output layer.

transfer function f, and output a as shown in Fig. 1. The neuron inputs come from the environment or other neurons in the previous layer. All weighted inputs and biases are summed and the result is passed through the transfer function to generate output, as in Eqs. (1) and (2). The output is then sent to other neurons in the next layer as input. The transfer functions commonly used in neural network are the linear function, the step function, and the sigmoid function (Demuth, Beale, & Hagan, 2006).

$$S = \sum_{i=1}^{n} w_i p_i + b \tag{1}$$

$$a = f(S) \tag{2}$$

The general topology of a neural network consists of an input layer, one or more hidden layers, and an output layer, as shown in Fig. 2. The input layer corresponds to object features that are used to classify objects. The output layer corresponds to an object class in the case of classification or a prediction value in the case of prediction. The hidden layers are located between the input layer and the output layer. A neural network can have one or more hidden layers. The number of hidden layers and their neurons depend on the complexity of the relationship between the input and the output. The hidden layers are constructed for the learning process by computation on neurons and weights. The weights and biases are adaptively adjusted during neural network training using a learning algorithm and training data until the weights converge (Du & Sun, 2008). The weights converge if the error between the predicted output and the desired output for all elements of training data reaches a minimum value. The common criteria used to measure the error between the predicted output and the desired output is the mean square error (MSE) (Zhang, 2000). The MSE of estimator $\hat{\mathbf{z}}$ is defined as the average of square difference between $\hat{\mathbf{z}}$ and desired output $\mathbf{z}$ as in Eq. (3),

$$\mathrm{MSE} = \frac{1}{K \times M} \sum_{i=1}^{K} \left\| \hat{\mathbf{z}}_i - \mathbf{z}_i \right\|^2 \tag{3}$$

where K is the number of sample in training data, M is the number of neural network output, $\|.\|$ is Euclidean norm, $\hat{\mathbf{z}}$ is predicted output and $\mathbf{z}$ is desired output.

3. Kalman filter

The Kalman filter was proposed by Kalman (1960) to solve the Wiener problem from the system state point of view. It has been applied in many areas including control systems, tracking, navigation, and estimation (Yeh & Huang, 2005). Generally, the Kalman filter is used to estimate the state of a linear dynamical system based on information from a measurement, which is linearly related to the state (Grewal & Andrews, 2008). Suppose $\mathbf{x} \in R^n$ is the state of a discrete controlled linear system and $\mathbf{z} \in R^l$ is the measurement. At time k the state and the measurement satisfy the process equation as in Eq. (4) and the measurement equation as in Eq. (5), respectively.

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \tag{4}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{5}$$

Where $\mathbf{x}_k$ is the state at time k, $\mathbf{u}_k \in R^m$ is the control input at time k, $\mathbf{z}_k$ is the measurement at time k. $\mathbf{A}_k$ is $n \times n$ matrix that relates the state at time $k-1$ and the state at time k, $\mathbf{B}_k$ is $n \times m$ matrix that relates the control input at time k and the state at time k, $\mathbf{H}_k$ is $l \times n$ matrix that relates the state and the measurement at time k; $\mathbf{w}_k$ and $\mathbf{v}_k$ are process and measurement noise, respectively, that are assumed to be normal random variables with a mean of $\mathbf{0}$ and covariance matrices of $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively.

In the state estimation, the Kalman filter consists of two phases, which are the predict phase and the update phase. In the predict phase, the Kalman filter uses information from the previous state to estimate the *a priori* current state. The *a priori* current state estimation is then updated using information from the measurement to produce the *a posteriori* current state estimation in the update phase. The predict phase and the update phase are performed using Eqs. (6)–(10), respectively (Welch & Bishop, 2006).
Predict phase:

$$\hat{\mathbf{x}}_k^- = \mathbf{A}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \mathbf{u}_k \tag{6}$$

$$\mathbf{P}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{Q}_k \tag{7}$$

Update phase:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k \right)^{-1} \tag{8}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^- \right) \tag{9}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \tag{10}$$

Where $\hat{\mathbf{x}}_k^- \in R^n$ is the *a priori* state estimation at time k, $n \times n$ matrices $\mathbf{P}_k^-$ and $\mathbf{P}_k$ are the *a priori* and the *a posteriori* estimation error covariance, respectively, and the $n \times l$ matrix $\mathbf{K}_k$ is the Kalman gain.

4. Proposed method

Suppose a trained neural network is used to classify an object into one of M classes as in Fig. 3(a). The input layer of the neural network consists of N neurons that correspond to object features $f_1, f_2, \ldots, f_N$ and the output layer consists of M neurons that correspond to desired output (class label) $z_1, z_2, \ldots, z_M$, where

$$z_i = \begin{cases} 1, & \text{if the object belong to class } c_i \\ 0, & \text{otherwise} \end{cases}, \quad i = 1, 2, \ldots, M.$$

Let $\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_M$ be the predicted output of the neural network, the values of $\tilde{z}_i$ is expected close to z_i for all $i = 1, 2, \ldots, M$ to ensure the object is correctly classified. The proposed method, called

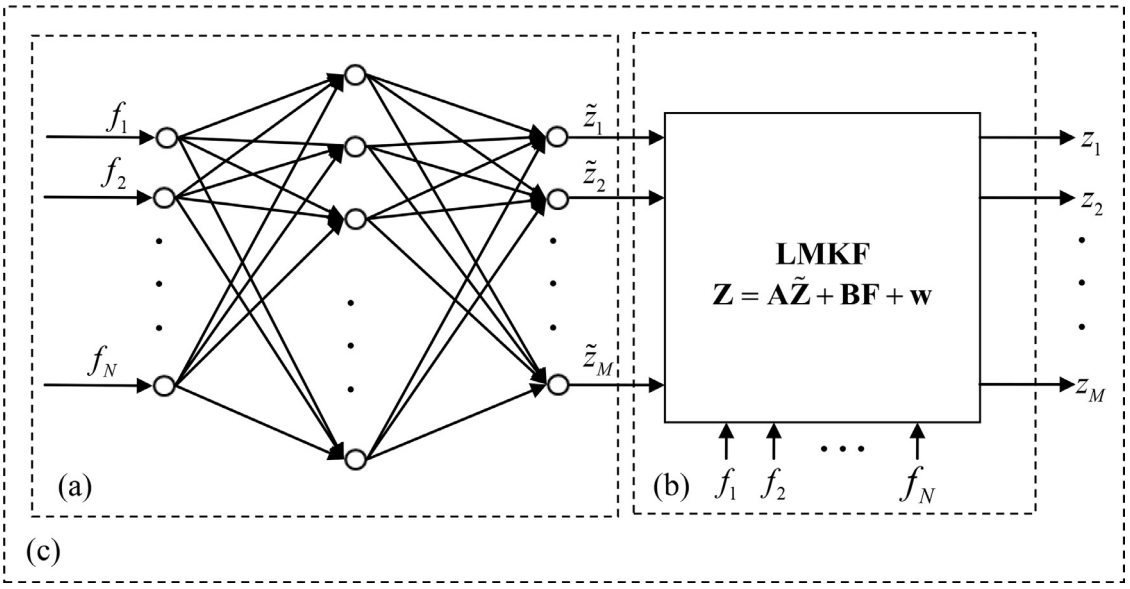

Fig. 3. The combinations (c) of neural network (a) and LMKF (b). LMKF uses the predicted output of neural network $\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_M$ and object features as input $f_1, f_2, \ldots, f_N$ to estimate class label.

neural network combined with linear model based on Kalman filter (NN-LMKF), is the combination of a neural network classifier (Fig. 3(a)) and a linear model (Fig. 3(b)) based on the Kalman filter (LMKF), as depicted in Fig. 3(c). The LMKF can be considered to be a post processing of the neural network classifier to increase classification performance. The parameters of LMKF are estimated using the Kalman filter iteration. The LMKF uses the predicted output of the neural network and the object features as input to estimate the class label.

The proposed method consists of two main phases, which are the training phase and the testing phase. The steps in the training phase are as follows: train the neural network, predict the output of the neural network for the objects in the training set, classify the objects in the training set using the neural network output, calculate the classification accuracy of the neural network classifier, construct the linear model, estimate the LMKF parameters using the Kalman filter, predict the output of the NN-LMKF, classify the objects in the training set using the NN-LMKF output, and calculate the classification accuracy of the NN-LMKF. For the testing phase, the steps consist of predicting the output of the NN-LMKF and classifying the objects using the NN-LMKF output. Fig. 4 shows the flowchart of the proposed method for the training phase and the testing phase.

4.1. Linear model construction

The LMKF is constructed to adjust the predicted output of the neural network such that the classification accuracy is increased by using the linear combination of object features. Therefore, the independent variables of the LMKF consist of object features $f_1, f_2, \ldots, f_N$ and predicted outputs of the neural network $\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_M$ while the dependent variables are the desired outputs$z_1, z_2, \ldots, z_M$. In addition, it is assumed that the model has a normally distributed error term with mean **0** and covariance matrix **R**, as formulated in Eq. (11):

$$\mathbf{z} = \mathbf{A}\tilde{\mathbf{z}} + \mathbf{Bf} + \mathbf{v}. \tag{11}$$

where $\mathbf{z} = [z_1\ z_2\ \ldots\ z_M]^T$, $\tilde{\mathbf{z}} = [\tilde{z}_1\ \tilde{z}_2\ \ldots\ \tilde{z}_M]^T$, $\mathbf{f} = [f_1\ f_2\ \ldots\ f_N]^T$, **A** is $M \times M$diagonal matrix as in Eq. (12), **B** is $M \times N$ matrix with element as in Eq. (13), and **v** is the error term. Matrices **A** and **B** are unknown parameters for the linear model.

$$\mathbf{A} = diag\begin{bmatrix} a_{11} & a_{22} & \cdots & a_{MM} \end{bmatrix} \tag{12}$$

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1N} \\ b_{21} & b_{22} & \cdots & b_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M1} & b_{M2} & \cdots & b_{MN} \end{bmatrix} \tag{13}$$

To estimate the parameters of the linear model in Eq. (11) using the Kalman filter, the process and the measurement equations for the system need to be defined first. The state of the system is defined as a vector **x** whose elements consist of the diagonal elements of matrix **A** and all elements of matrix **B**, as in Eq. (14):

$$\mathbf{x} = \begin{bmatrix} a_{11} & a_{22} & \cdots & a_{MM} & b_{11} & b_{12} & \cdots & b_{1N} & \cdots & b_{M1} & b_{M2} & \cdots & b_{MN} \end{bmatrix}^T. \tag{14}$$

Because the parameters of the linear model in Eq. (11) are constant values, the state does not change from time to time. It is assumed that the state is only perturbed by white noise. Furthermore, there is no control input for the system. Therefore, the dynamics of the system can be expressed as a stationary process perturbed by white noise and the state equation at time k is defined in Eq. (15):

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{w}_k, \tag{15}$$

where $\mathbf{w}_k$ is state noise that is assumed to be a normal random variable with mean **0** and covariance matrix **Q**.

The linear model in Eq. (11) is used as the measurement equation with a slight modification as in Eq. (16),

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{16}$$

where $\mathbf{H} = \partial\mathbf{z}/\partial\mathbf{x}$ is the Jacobian matrix of **z** whose elements are all first order partial derivatives of **z** with respect to all elements

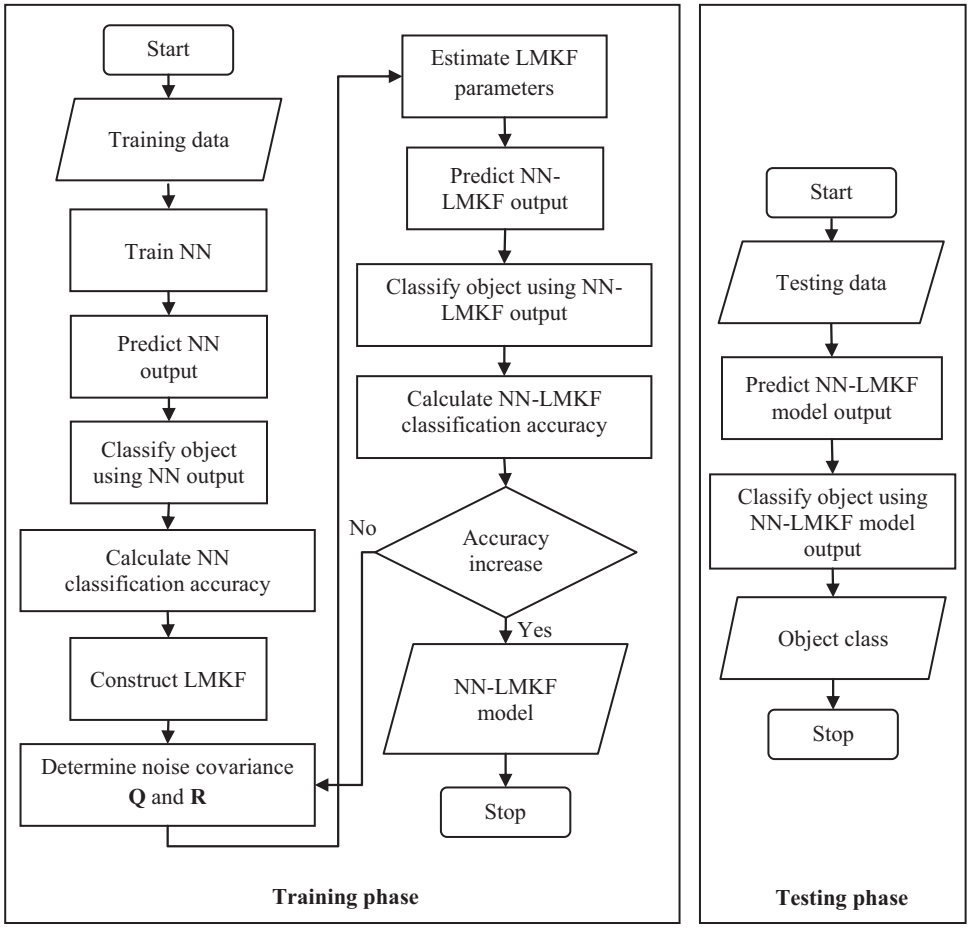

Fig. 4. The flowchart of proposed method for the training phase (left) and the testing phase (right).

of **x**. Therefore, the elements of **H** consist of the elements of $\tilde{\mathbf{z}}$ and **f**, and 0 s, as in Eq. (17):

$$\mathbf{H} = \begin{bmatrix} \tilde{z}_1 & 0 & \cdots & 0 & f_1 & f_2 & \cdots & f_{Nf} & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \tilde{z}_2 & \cdots & 0 & 0 & 0 & \cdots & 0 & f_1 & f_2 & \cdots & f_N & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \tilde{z}_M & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & f_1 & f_2 & \cdots & f_N \end{bmatrix}. \tag{17}$$

4.2. Parameter estimation using the Kalman filter

Based on the process equation in Eq.(15), the measurement equation in Eq. (16), and Eqs. (6)–(10), the predict and update phases in the Kalman filter iteration are performed using Eqs. (18)–(22):

Predict phase:

$$\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1} \tag{18}$$

$$\mathbf{P}_k^- = \mathbf{P}_{k-1} + \mathbf{Q} \tag{19}$$

Update phase:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R} \right)^{-1} \tag{20}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^- \right) \tag{21}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \tag{22}$$

Before performing the Kalman filter iteration, the values of **Q**, **R**, $\hat{\mathbf{x}}_0$ and $\mathbf{P}_0$ should first be assigned. Because there is no information about the values of **Q** and **R**, the proposed method assumes that **Q** and **R** are scalar matrices in the form $\mathbf{Q} = q\mathbf{I}$ and $\mathbf{R} = r\mathbf{I}$, where q and r are positive real numbers. The values of q and r are chosen such that the classification accuracy is increased after applying the NN-LMKF to the training data. Furthermore, $\hat{\mathbf{x}}_0 = \mathbf{0}$ and $\mathbf{P}_0 = \mathbf{I}$ are chosen as the initial values of **x** and **P**, respectively. The iteration is performed using the entire training data set until the convergence criteria are satisfied. The convergence criterion for the Kalman filter iteration is chosen from one of the following criteria:

- Small mean square error (MSE): MSE $< \varepsilon_1$
- Stable state: $\|\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_{k-1}\| < \varepsilon_2$
- Exceeds the maximum epoch

where ε_1 and ε_2 are small positive real number, and MSE is calculated using Eq. (3).

Once the parameter estimation is completed, the NN-LMKF is then used to estimate the desired output (class label) **z** of the object in the testing set based on object features $f_1, f_2, \ldots, f_N$ and the predicted output of the neural network, $\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_M$. The estimator $\hat{\mathbf{z}}$ is used to classify the object, the object belongs to class c_i

Table 1
The summary of datasets used in validation.

Dataset name	Attribute type	Number of attribute	Number of sample	Number of class
Glass identification	Real	10	214	2
Iris	Real	4	150	3
Statlog (Vehicle Silhouettes)	Integer	18	846	4
Statlog (Australian Credit Approval)	Categorical, real	14	690	2
Statlog (Heart)	Categorical, integer, real	13	270	2

Algorithm 1 Algorithm to perform the parameter estimation of LMKF

INPUT: Training set $Tr = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K\}$, desired output$\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K\}$, convergence criteria ε and *maxEpoch*, covariance matrix $\mathbf{Q}$ and $\mathbf{R}$.
OUTPUT: LMKF parameters estimator $\hat{\mathbf{x}}$
Train neural network using *Tr*
FOR *j* **FROM** 1 **TO** *K*
  Predict output of neural network $\tilde{\mathbf{z}}_j = \text{PredictNN}(\mathbf{f}_j)$
END FOR
Classify every object in *Tr* using neural network output
Calculate classification accuracy of neural network Acc_{NN}
Set classification accuracy of NN-LNKF $Acc_{NN-LNKF} = 0$
WHILE $Acc_{NN} \leq Acc_{NN-LNKF}$
  INPUT covariance matrix of state $\mathbf{Q} = q\mathbf{I}$
  INPUT covariance matrix of measurement $\mathbf{R} = r\mathbf{I}$
  Set initial value of state $\hat{\mathbf{x}}_0 = \mathbf{0}$
  Set initial value of covariance matrix of state $\mathbf{P}_0 = \mathbf{I}$
  FOR *i* **FROM** 1 **TO** *maxEpoch*
    FOR *k* **FROM** 1 **TO** *K*
      Calculate $\mathbf{H}_k$ from $\tilde{\mathbf{z}}_k$ and $\mathbf{f}_k$
      Calculate $\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1}$
      Calculate $\mathbf{P}_k^- = \mathbf{P}_{k-1} + \mathbf{Q}$
      Calculate $\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R})^{-1}$
      Calculate $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-)$
      Calculate $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^-$
    END FOR
    FOR *k* **FROM** 1 **TO** *K*
      Predict output of NN-LNKF $\hat{\mathbf{z}}_k = \mathbf{H}_k \hat{\mathbf{x}}_k$
    END FOR
    Calculate MSE
    IF MSE $< \varepsilon$**THEN**
      Break
    END IF
  END FOR
  Classify every object in *Tr* using NN-LMKF output
  Calculate classification accuracy of NN-LMKF $Acc_{NN-LNKF}$
END WHILE

if $\hat{z}_i = \max\{\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_M\}$. The steps for the parameter estimation of LMKF are summarized in Algorithm 1.

5. Empirical validation

5.1. Datasets

The proposed method was validated using five datasets from the UCI Machine Learning Repository (Bache & Lichman, 2013). The datasets were chosen from various domains with various characteristics, such as attribute types, the number of attributes, the number of samples, and the number of classes, as summarized in Table 1. The datasets used for validation included Glass Identification dataset, Iris dataset, Statlog (Vehicle Silhouettes) dataset, Statlog (Australian Credit Approval) dataset, and Statlog (Heart) dataset with the following characteristics:

1. Glass Identification dataset belongs to the Forensic Science domain and is used to identify types of glass (Evett & Spiehler, 1987). This dataset consist of 214 samples (163 window glasses and 51 non-window glasses) with 10 attributes including object Id. All attributes are real valued.
2. Iris dataset is a famous dataset found in the pattern recognition literature (Duda & Hart, 1973). This dataset consists of 150 samples (50 Iris Setosa, 50 Iris Versicolour, 50 Iris Virginica) with four real valued attributes.
3. Statlog (Vehicle Silhouettes) dataset is used to recognize 3D objects (vehicle) using shape features extracted from the 2D silhouette of the object (Siebert, 1987). This dataset consists of 846 samples from four classes (212 opel, 217 saab, 218 bus, 199 van) with 18 integer attributes.
4. Statlog (Australian Credit Approval) dataset is used to classify credit card applicants (Bache & Lichman, 2013). This dataset consists of 690 samples (307 first class and 383 second class) with a good mix of attributes (14 attributes: real and categorical). There are a few missing values in this dataset. The missing values were replaced with the mean and mode of the attribute for real valued and categorical attribute, respectively.
5. Statlog (Heart) dataset is used in the diagnosis of heart disease (Bache & Lichman, 2013). This dataset consists of 270 samples (150 without heart disease and 120 with heart disease) with 13 attributes (real, integer, and categorical).

5.2. Validation method

Random subsampling was used to evaluate the classification performance of the proposed method. Each dataset was randomly portioned into two mutually exclusive sets, a training set, on which the training phase was performed, and a testing set, on which the testing phase was performed. Ten training sets and ten testing sets were made by randomly selecting objects from each original dataset. The training set and the testing set were constructed in the following manner:

- 50% of the objects were randomly selected from the original dataset as the training set and the rest were selected as the testing set.
- The proportions of classes in each training set were equal to the proportions of classes in each testing set.

Each training set was used to train the neural network and to estimate the parameters of the LMKF using the Kalman filter iteration, and the testing set was used to validate the proposed method.

The architecture of neural network used in this validation consisted of an input layer with N neurons, a hidden layer with $\lfloor (N+M)/2 \rfloor$ neurons, and an output layer with M neurons, where $\lfloor x \rfloor$ is the largest integer less than or equal to x. The tangent sigmoid function $\tanh(x)$, as in Eq. (23), was used as the transfer function both from the input layer to the hidden layer and from the hidden layer to the output layer. The neural network was trained using the Levenberg–Marquardt backpropagation algorithm.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{23}$$

The classification accuracy for a particular dataset was obtained by calculating the average of the classification accuracies on ten testing sets both for the original neural network and the proposed method. The classification accuracy was calculated using Eq. (24)

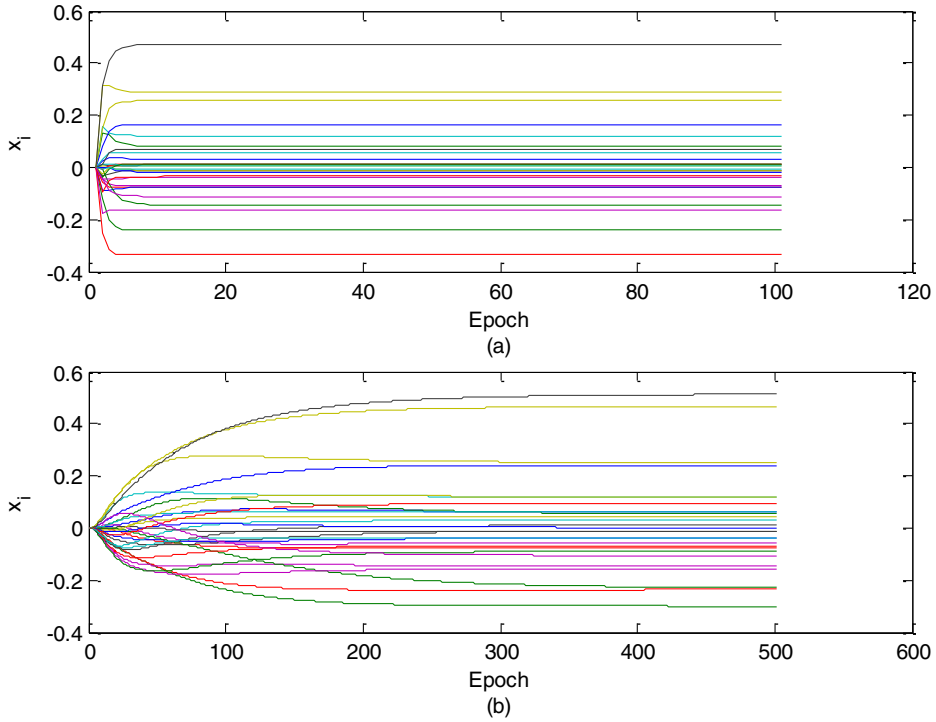

Fig. 5. The effect of choosing inappropriate matrix $\mathbf{R} = r\mathbf{I}$ to the dynamic of system: choosing too small an r value results in early achievement of a steady state (a), while choosing too large an r value results in slow movement of the state (b).

Table 2
The values of r used in estimating LMKF parameters for all datasets.

Dataset name	The values of r
Glass identification	10^5
Iris	10^3, 10^4, 10^5, 10^6
Statlog (Vehicle Silhouettes)	10^5, 10^6
Statlog (Australian Credit Approval)	8×10^5, 10^6, 2×10^6, 3×10^6, 10^7
Statlog (Heart)	3×10^6, 5×10^6, 6×10^6, 10^7, 1.2×10^7, 2×10^7, 4×10^7, 5×10^7

(Hacibeyoglu et al., 2011):

$$\text{Classification accuracy} = \frac{\text{the number of objectscorrectly classified}}{\text{total number of objects}} \times 100\%. \quad (24)$$

Finally, the improvement of the classification accuracy ψ for all datasets used for validation was calculated using Eq. (25):

$$\psi = \frac{\sum_{i=1}^{D} \varphi_i N_i}{\sum_{i=1}^{D} N_i}. \quad (25)$$

Where D is the number of dataset used for validation, φ_i is the average of increasing classification accuracy for the ith dataset, N_i is the number of objects in the testing set of the ith dataset.

The next performance evaluation was performed based on the receiver operating characteristic (ROC). The ROC curve has been used to show the trade-off between hit rates and false alarm rates of classifiers in signal detecting theory. Currently, it is widely used in medical decision making to evaluate the performance of a diagnostic test, as in Mazurowski et al. (2008), Akay (2009), and Seera and Lim (2014). In machine learning, this curve is used to depict the performance of a binary classifier by calculating the true positive rate and the false positive rate in the several values of threshold. It is constructed by plotting a two-dimensional curve where the horizontal axis and the vertical axis are the false positive rate and the true positive rate, respectively. For a multiclass classifier with M classes, M different ROC curves are constructed, one for each class, by using a class reference formulation. The class reference ROC_i curve describes the classification performance using class c_i as the positive class and the others as the negative classes. In this study, the area under the ROC curve (AUC) was used to evaluate the performance of the proposed method. The AUC for a multiclass classifier is calculated from the AUC of each class reference ROC curve using Eq. (26):

$$\text{AUC}_{\text{Total}} = \sum_{i=1}^{M} p_i \, \text{AUC}_i. \quad (26)$$

Where AUC_i is area under the class reference ROC_i curve and p_i is the proportion of class c_i in the testing set. The higher the AUC value, the better classifier performance is. (Fawcett, 2006).

6. Result and discussion

For simplification, in this study the value of $\mathbf{Q}$ was set to the identity matrix $\mathbf{I}$ for all testing sets and only the value of $\mathbf{R} = r\mathbf{I}$ varied. Therefore, the dynamics of the state $\mathbf{x}$ are influenced by only training data and r. Too small an r value results in early achievement of a steady state, but the state may be steady with improper values and may produce reduced classification accuracy. Fig. 5(a) shows early achievement of a steady state as a result of choosing too small an r value. Conversely, too large an r value results in slow movement of the state, as shown in Fig. 5(b). As a

Table 3
The classification accuracy of NN and NN-LMKF for testing data in all datasets.

Dataset name	The average and std. dev. of classification accuracy (%)		The average and std. dev. of improvement (%)
	NN	NN-LMKF	
Glass identification	89.06 ± 5.17	93.30 ± 2.11	4.25 ± 4.22
Iris	79.20 ± 14.25	94.93 ± 1.76	15.73 ± 14.02
Statlog (Vehicle Silhouettes)	71.31 ± 6.50	77.49 ± 4.01	6.11 ± 4.16
Statlog (Australian Credit Approval)	82.18 ± 3.61	87.85 ± 0.76	5.67 ± 3.93
Statlog (Heart)	78.52 ± 3.81	85.04 ± 1.67	6.52 ± 3.98

Table 4
The Classification results for all dataset from other methods.

Dataset name	Method	Accuracy (%)	Ratio	Runs	Author(s)
Glass identification	BPNN-GA	92.77	9:1	10	Li (2006)
	RBFNN-GA	92.36	9:1	10	
	LVQNN-GA	92.71	9:1	10	
Iris	Clustering GP	97.9	9:1	10	Eggermont, Kok, and Kosters (2004)
	Refined FP (gain)	94.9	9:1	10	
	Refined FP (gain-ratio)	68.3	9:1	10	
	Logitboost NB	94.87	9:1	10	Kotsiantis and Pintelas (2005)
	G-FDT	98	9:1	10	Chandra and Paul Varghese (2009)
	Hybrid DT	98.66	9:1	10	Farid et al. (2014)
	Hybrid NB	98	9:1	10	
Statlog (Vehicle Silhouettes)	Logitboost NB	70.91	9:1	10	Kotsiantis and Pintelas (2005)
	G-FDT	70.12	9:1	10	Chandra and Paul Varghese (2009)
Statlog (Australian Credit Approval)	Clustering GP	86.3	9:1	10	Eggermont, Kok, and Kosters (2004)
	Refined FP (gain)	85.8	9:1	10	
	Refined FP (gain-ratio)	84.5	9:1	10	
Statlog (Heart)	Clustering GP	80.9	9:1	10	Eggermont, Kok, and Kosters (2004)
	Refined FP (gain)	80.4	9:1	10	
	Refined FP (gain-ratio)	81.3	9:1	10	
	Logitboost NB	79.3	9:1	10	Kotsiantis and Pintelas (2005)
	G-FDT	75.33	9:1	10	Chandra and Paul Varghese (2009)

Table 5
The area under the ROC curve (AUC) of NN and NN-LMKF on all datasets.

Dataset name	The average and std. dev. of AUC		The average and std. dev. of AUC increment
	NN	NN-LMKF	
Glass identification	0.887 ± 0.074	0.958 ± 0.045	0.071 ± 0.085
Iris	0.861 ± 0.141	0.979 ± 0.009	0.118 ± 0.137
Statlog (Vehicle Silhouettes)	0.866 ± 0.054	0.914 ± 0.018	0.048 ± 0.047
Statlog (Australian Credit Approval)	0.879 ± 0.033	0.925 ± 0.009	0.046 ± 0.029
Statlog (Heart)	0.823 ± 0.063	0.883 ± 0.031	0.060 ± 0.055

consequence, a large number of epochs are required to achieve stability. The value of r was chosen by trial and error such that the classification accuracy is increased after applying LMKF to the training data. The values of r used in validation of each dataset are tabulated in Table 2.

The classification accuracy results of the proposed method and the original neural network for each dataset are summarized in Table 3. It can be observed from Table 3 that the proposed method improved the classification accuracy of the original neural network for all datasets. The minimum improvement was 4.25% on average for Glass Identification dataset. The best improvement was 15.47% on average for Iris dataset. In most of the testing sets (92% testing set) the improvement of classification accuracy was greater than 1%. The proposed method also reduced the standard deviation of classification accuracy of the original neural network. This shows that the proposed method has classification accuracy with a smaller variation than the original neural network. Finally, the improvement of the classification accuracy for all of the datasets used for validation was 6.50%. This result shows that the proposed method is able to improve the classification performance of the original neural network regardless of the type of object attribute.

The value of covariance matrices **Q** and **R** play an important role in the estimation of LMKF parameters. In this study, the value of **Q** and **R** were determined using trial error. Choosing inappropriate values of **Q** and **R** may lead to increasing the MSE and decreasing the classification accuracy. This is a weakness of the proposed method. In this study, the proposed method was compared with other classification methods from previous researches. Table 4 shows the classification results for all datasets from other methods. As can be seen in Tables 3 and 4, the proposed method outperforms other methods for almost all datasets, except for Iris dataset. All classification methods in Table 4 used 10-folds cross validation. Therefore the ratio between training data and testing data is 9:1. On the other hand, the proposed method only used 50% of dataset for training data. This fact shows that the proposed method only requires smaller training data to achieve better performance, except for Iris dataset. For Iris dataset, Clustering GP, G-FDT, Hybrid DT, and Hybrid NB using the 9:1 training data to testing data ratio obtained higher classification accuracy than the proposed methods. Therefore, to achieve better classification accuracy on Iris dataset, the proposed method may need the number of object in training data greater than 50% of dataset.

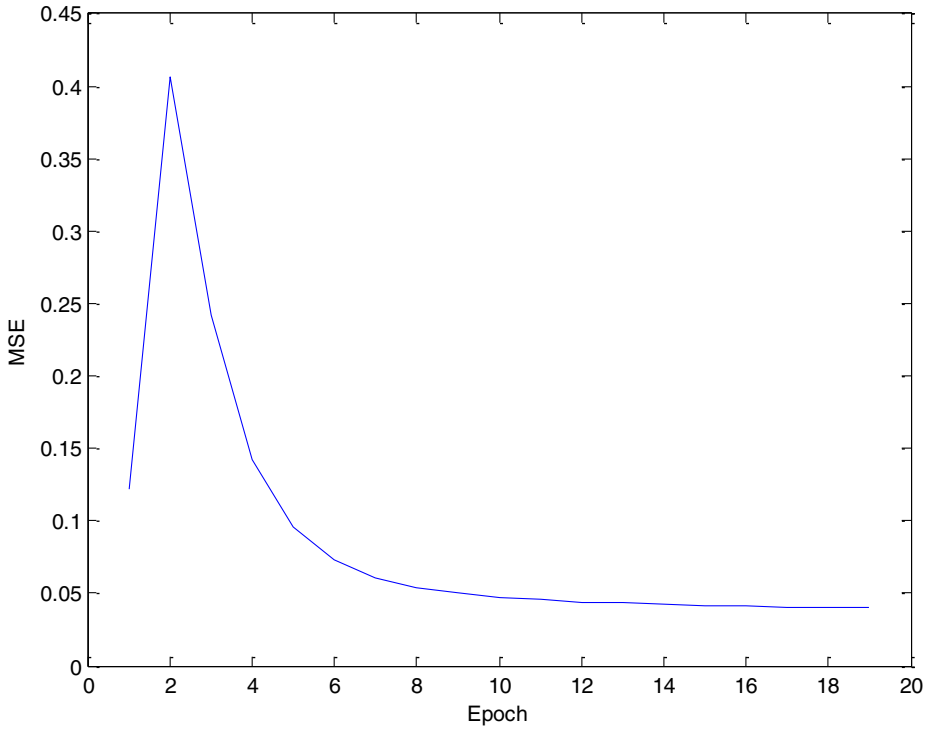

Fig. 6. The MSE of the NN-LMKF reduction for each epoch during Kalman filter iteration.

Table 6
The summary of the mean square error (MSE) of NN and NN-LMKF for training data in all datasets.

Dataset name	The average and std. dev. of MSE		The average and std. dev. of reduction
	NN	NN-LMKF	
Glass identification	0.062 ± 0.039	0.033 ± 0.015	0.029 ± 0.027
Iris	0.089 ± 0.081	0.042 ± 0.028	0.047 ± 0.057
Statlog (Vehicle Silhouettes)	0.078 ± 0.020	0.059 ± 0.009	0.019 ± 0.016
Statlog (Australian Credit Approval)	0.133 ± 0.036	0.106 ± 0.016	0.027 ± 0.027
Statlog (Heart)	0.137 ± 0.037	0.118 ± 0.023	0.019 ± 0.026

Table 7
The computing time and CPU usage for the training step in all datasets.

Dataset name	The average of computing time (s)	The average of CPU usage (%)
Glass identification	3.25	48
Iris	2.21	48
Statlog (Vehicle Silhouettes)	33.58	98
Statlog (Australian Credit Approval)	10.06	93
Statlog (Heart)	6.13	96

The AUCs for each classifier were calculated to extract information from the ROC curve to a single scalar value representing classifier performance. The trapezoidal rule (Kiusalaas, 2010) was used to calculate the AUC of each class reference ROC curve. The AUC totals for each classifier were then calculated using Eq. (26) and the results are summarized in Table 5. It can be observed in Table 5 that on average the AUC of the NN-LMKF is greater than the AUC of the original neural network for all datasets. The AUC increased for all datasets, and a minimum AUC increment of 0.046 on average was achieved for Statlog (Australian Credit Approval) dataset. The maximum AUC increment of 0.118 on average was achieved for Iris dataset. This result reinforces the previous classification accuracy analysis. Therefore, it can be ensured that the proposed method achieves better performance than the original neural network.

From the MSE point of view, the proposed method also reduced the MSE of the original neural network for all training data. Fig. 6 depicts the MSE of the NN-LMKF reduction for each epoch. As observed in Fig. 6, at the beginning of the Kalman filter iteration, the MSE of the NN-LMKF might be greater than the MSE of the original neural network. The MSE of the NN-LMKF then decreased asymptotically to a value below the MSE of the original neural network as the number of epochs increased. The MSEs of the NN-LMKF for each testing set are summarized in Table 6. As observed in Table 6, the averages of the MSE were reduced after applying the NN-LMKF to all datasets. On average, the minimum MSE reduction was 0.019 for Statlog (Heart) dataset and the maximum was 0.047 for Iris dataset. This result indicates that the predicted output of the NN-LMKF is closer to the desired output than is the predicted output of the original neural network.

The computing time and CPU usage for the training step depend on the number of features, classes, and samples in the training set. The more features, classes, and objects, the more computing time and the higher CPU usage are needed to perform the training phase. For example, to perform the training phase in a 3.00 GHz

Pentium(R) Dual-Core personal computer with 2.00 GB RAM and Windows 7 operating system, it took approximately 2.21 s and 48% of CPU usage in average for Iris dataset. For Statlog (Vehicle Silhouettes) dataset, the average of computing time and CPU usage were 33.58 s and 98% respectively. To perform classification in the testing phase, the computing time was less than 0.017 seconds per sample. The computing time and CPU usage for the training step in all datasets are summarized in Table 7.

7. Conclusion and future work

In this paper, a method for improving the classification accuracy of a neural network is proposed. The proposed method employed simple transformation model, called LMKF, as a post processing of the neural network. The model used the linear combination of the object features and the predicted output of the neural network to decrease the error of neural network. The parameters of the LMKF were estimated using the Kalman filter iteration during the training phase. The proposed method has been validated using five datasets from the UCI Machine Learning Repository that have different attribute types, numbers of attributes, numbers of objects, and numbers of classes. The validation results show that the proposed method has the ability to improve the classification accuracy of the original neural network regardless of the type of object attribute. In comparison with other classification methods, the proposed method achieved better performance for almost all dataset using smaller training data. Furthermore, the AUC analysis shows that the proposed method achieves better performance than the original neural network. In addition, the predicted output of the NN-LMKF is closer to the desired output than is the predicted output of the original neural network.

Although it has been validated that LMKF can increase the classification accuracy of neural network, there are some limitations on the proposed method. The first, if the object features, the predicted output of neural network, and the desired output do not have linear relationship then a linear model cannot be used to transform the predicted output of neural network to a value close to the desired output. Therefore, in this condition the proposed method may fail to increase the classification performance of neural network. The second, increasing the number of attributes and classes will result in increasing the computing time and CPU usage due to increasing the size of matrices involved in Kalman filter iteration.

For future research, the application of LMKF to improve the performance of other classifiers, such as the support vector machine (SVM), should be investigated. To obtain an optimal LMKF parameter estimator more accurately, an optimization method could be considered in determining the covariance matrices **Q** and **R**. Furthermore, the use of other methods, such as metaheuristic methods, could also be considered to replace Kalman filter iteration in estimating the parameters of the linear model. In addition, the application of a nonlinear model based on the extended Kalman filter for improving classifier performance should also be investigated.

Acknowledgments

The authors would like to thank the Ministry of Education Malaysia and Universiti Kebangsaan Malaysia for providing facilities and financial support under grant no. ERGS/1/2013/ICT07/UKM/02/2.

References

Affonso, C., Sassi, R. J., & Barreiros, R. M. (2015). Biological image classification using rough-fuzzy artificial neural network. *Expert Systems with Applications, 42*, 9482–9488.

Akay, M. F. (2009). Support vector machines combined with feature selection for breast cancer diagnosis. *Expert Systems with Applications, 36*, 3240–3247.

Alpaydin, E. (2010). *Introduction to machine learning* (2nd ed). Cambridge, Massachusetts: MIT Press.

Bache, K., & Lichman, M. (2013). *UCI machine learning repository*. Irvine, CA: University of California, School of Information and Computer Science.

Chandaka, S., Chatterjee, A., & Munshi, S. (2009). Support vector machines employing cross-correlation for emotional speech recognition. *Measurement, 42*, 611–618.

Chandra, B., & Paul Varghese, P. (2009). Fuzzifying Gini index based decision trees. *Expert Systems with Applications, 36*, 8549–8559.

Chen, M.-C., & Huang, S.-H. (2003). Credit scoring and rejected instances reassigning through evolutionary computation techniques. *Expert Systems with Applications, 24*, 433–441.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning, 3*, 261–283.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*, 273–297.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*, 21–27.

Demuth, H. B., Beale, M., & Hagan, M. (2006). *Neural network toolbox for use with MATLAB: User's guide*. MathWorks, Incorporated.

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning, 29*, 103–130.

Du, C.-J., & Sun, D.-W. (2008). 4- Object classification methods. In S. Da-Wen (Ed.), *Computer vision technology for food quality evaluation* (pp. 81–107). Amsterdam: Academic Press.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. Wiley.

Eggermont, J., Kok, J. N., & Kosters, W. A. (2004). Genetic programming for data classification: Partitioning the search space. In *Proceedings of the 2004 ACM symposium on applied computing* (pp. 1001–1005). ACM.

Evett, I.W., & Spiehler, E.J. (1987). Rule induction in forensic science. In: Central Research Establishment, Home Office Forensic Science Service.

Farid, D. M., Zhang, L., Rahman, C. M., Hossain, M. A., & Strachan, R. (2014). Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks. *Expert Systems with Applications, 41*, 1937–1946.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*, 861–874.

Gao, T.-F., & Liu, C.-L. (2008). High accuracy handwritten Chinese character recognition using LDA-based compound distances. *Pattern Recognition, 41*, 3442–3451.

Garrido, C., de Oña, R., & de Oña, J. (2014). Neural networks for analyzing service quality in public transportation. *Expert Systems with Applications, 41*, 6830–6838.

Giacinto, G., & Roli, F. (2001). Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing, 19*, 699–707.

Glodek, M., Reuter, S., Schels, M., Dietmayer, K., & Schwenker, F. (2013). Kalman filter based classifier fusion for affective state recognition. In Z.-H. Zhou, F. Roli, & J. Kittler (Eds.), *Multiple classifier systems: 7872* (pp. 85–94). Berlin Heidelberg: Springer.

Goh, W. L., Mital, D. P., & Babri, H. A. (1997). An artificial neural network approach to handwriting recognition. In Proceedings of 1997 first international conference on knowledge-based intelligent electronic systems, 1997. KES '97. (Vol. 1, pp. 132-136 vol.131).

Grewal, M. S., & Andrews, A. P. (2008). *Kalman filtering: Theory and practice using MATLAB* (3rd ed.). John Wiley & Sons, Inc.

Hacibeyoglu, M., Arslan, A., & Kahramanli, S. (2011). Improving classification accuracy with discretization on datasets including continuous valued features. *International Science Index, 5*, 497–500.

Huang, C.-L., Chen, M.-C., & Wang, C.-J. (2007). Credit scoring with a data mining approach based on support vector machines. *Expert Systems with Applications, 33*, 847–856.

Jamil, N., Mohamed, A., & Abdullah, S. (2009). Automated grading of palm oil fresh fruit bunches (FFB) using neuro-fuzzy technique. In International conference of soft computing and pattern recognition, 2009. SOCPAR '09. (pp. 245–249).

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering, 82*, 35–45.

Kılıç, K., Boyacı, İ. H., Köksel, H., & Küsmenoğlu, İ (2007). A classification system for beans using computer vision system and artificial neural networks. *Journal of Food Engineering, 78*, 897–904.

Kiusalaas, J. (2010). *Numerical methods in engineering with MATLAB* (2nd ed.). New York: Cambridge University Press.

Kotsiantis, S. B., & Pintelas, P. E. (2005). Logitboost of simple Bayesian classifier. *Informatica (Slovenia), 29*, 53–60.

Li, M., & Yuan, B. (2005). 2D-LDA: A statistical linear discriminant analysis for image matrix. *Pattern Recognition Letters, 26*, 527–532.

Li, T.-S. (2006). Feature selection for classification by using a ga-based neural network approach. *Journal of the Chinese Institute of Industrial Engineers, 23*, 55–64.

Lippmann, R. P. (1987). An introduction to computing with neural nets. *ASSP Magazine, IEEE, 4*, 4–22.

Lyle, J. R., Miller, P. E., Pundlik, S. J., & Woodard, D. L. (2012). Soft biometric classification using local appearance periocular region features. *Pattern Recognition, 45*, 3877–3885.

Mastrogiannis, N., Boutsinas, B., & Giannikos, I. (2009). A method for improving the accuracy of data mining classification algorithms. *Computers & Operations Research, 36*, 2829–2839.

Mazurowski, M. A., Habas, P. A., Zurada, J. M., Lo, J. Y., Baker, J. A., & Tourassi, G. D. (2008). Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural Networks, 21*, 427–436.

Min, J. H., & Lee, Y.-C. (2005). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert Systems with Applications, 28*, 603–614.

Nashat, S., Abdullah, A., & Abdullah, M. Z. (2014). Machine vision for crack inspection of biscuits featuring pyramid detection scheme. *Journal of Food Engineering, 120*, 233–247.

Onan, A. (2015). A fuzzy-rough nearest neighbor classifier combined with consistency-based subset evaluation and instance selection for automated diagnosis of breast cancer. *Expert Systems with Applications, 42*, 6844–6852.

Pruengkarn, R., Chun Che, F., & Kok Wai, W. (2015). Using misclassification data to improve classification performance. In 12th international conference on electrical engineering/electronics, computer, telecommunications and information technology (ECTI-CON), 2015 (pp. 1–5).

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106.

Rocha, A., Hauagge, D. C., Wainer, J., & Goldenstein, S. (2010). Automatic fruit and vegetable classification from images. *Computers and Electronics in Agriculture, 70*, 96–104.

Seera, M., & Lim, C. P. (2014). A hybrid intelligent system for medical data classification. *Expert Systems with Applications, 41*, 2239–2249.

Siebert, J. P. (1987). *Vehicle recognition using rule based methods*. Turing Institute.

Sigillito, V. G., Wing, S. P., Hutton, L. V., & Baker, K. B. (1989). Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest, 10*, 262–266.

Tsai, C.-F., & Wu, J.-W. (2008). Using neural network ensembles for bankruptcy prediction and credit scoring. *Expert Systems with Applications, 34*, 2639–2649.

Verma, B., & Zhang, P. (2007). A novel neural-genetic algorithm to find the most significant combination of features in digital mammograms. *Applied Soft Computing, 7*, 612–625.

Welch, G., & Bishop, G. (2006). *An introduction to the Kalman filter*. Chapel Hill, NC: University of North Carolina at Chapel Hil.

Wu, W., Rutan, S. C., Baldovin, A., & Massart, D.-L. (1996). Feature selection using the Kalman filter for classification of multivariate data. *Analytica Chimica Acta, 335*, 11–22.

Yeh, H. D., & Huang, Y. C. (2005). Parameter estimation for leaky aquifers using the extended Kalman filter, and considering model and data measurement uncertainties. *Journal of Hydrology, 302*, 28–45.

Zaamout, K., & Zhang, J. Z. (2012). Improving classification through ensemble neural networks. In Eighth international conference on natural computation (ICNC), 2012 (pp. 256–260).

Zhang, G. P. (2000). Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 30*, 451–462.

Zhifeng, L., Dahua, L., & Xiaoou, T. (2009). Nonparametric Discriminant Analysis for Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*, 755–761.