Neural Comput & Applic (2016) 27:537–548
DOI 10.1007/s00521-015-1875-2

ORIGINAL ARTICLE

Artificial neural network training utilizing the smooth variable structure filter estimation strategy

Ryan Ahmed[1] · Mohammed El Sayed[1] · S. Andrew Gadsden[2] · Jimi Tjong[3] · Saeid Habibi[1]

Received: 20 June 2013 / Accepted: 13 March 2015 / Published online: 27 March 2015

Abstract A multilayered neural network is a multi-input, multi-output nonlinear system in which network weights can be trained by using parameter estimation algorithms. In this paper, a novel training method is proposed. This method is based on the relatively new smooth variable structure filter (SVSF) and is formulated for feed-forward multilayer perceptron training. The SVSF is a state and parameter estimation that is based on the sliding mode concept and works in a predictor–corrector fashion. The SVSF training performance is tested on three benchmark pattern classification problems. Furthermore, a study is presented comparing the popular back-propagation method, the extended Kalman filter, and the SVSF.

Keywords Neural networks · Extended Kalman filter · Smooth variable structure filter · Estimation

1 Introduction

Artificial neural networks (ANNs) are mathematical models that process information and adapt in a fashion inspired by the human brain. ANNs are capable of modeling relationships between a set of inputs and outputs. ANNs show enhanced generalization capability, adaptation competency, and potent nonlinear input–output mapping [1]. In fact, a neural network with a sufficient number of neurons can approximately model any continuous function with an acceptable degree of accuracy [2, 3]. ANNs are applied in numerous applications such as pattern classification [4], pattern recognition [2], function approximation, data processing, and robotics applications [5]. This section is divided into two subsections, namely traditional ANN training strategies and estimator-based ANN training techniques.

1.1 Traditional ANN training techniques

Since the 1980s, several ANN training techniques have been proposed. Backpropagation (BP) is one of the first used techniques in training of multilayer perceptrons [6]. It was reported by Rumelhart, Hinton, and Williams in 1986 [7]. BP is a first-order stochastic gradient descent method that iteratively searches for link weights that minimize the output error in a supervised manner. However, since early versions of BP involve a constant learning rate, a slow speed of convergence is attained. In fact, several enhanced training algorithms have been developed to improve training performance, mapping accuracy, and speed of convergence compared to the BP algorithm [8]. For instance, the nonlinear least-squares Gauss–Newton method is a good candidate to iteratively solve supervised neural network training problems [7]. Nevertheless, it has been shown in [9] that the Jacobian matrix may become rank deficient in some cases, thus resulting in the numerical instability of the Gauss–Newton algorithm [10]. The second-order Levenberg–Marquardt training algorithm [11] has shown to circumvent the previous problems. Watrous [12] verified the application of a quasi-Newton method to neural network training. Quasi-Newton method demonstrated better convergence performance than the standard

✉ S. Andrew Gadsden
gadsden@umbc.edu

[1] Department of Mechanical Engineering, McMaster University, Hamilton, ON, Canada

[2] Department of Mechanical Engineering, University of Maryland, Baltimore County, Baltimore, MD, USA

[3] Ford Canada, Windsor, ON, Canada

BP algorithm, but it requires large memory storage to store the Hessian matrix [13]. The quasi-Newton and Levenberg–Marquardt demonstrate better performance than BP as they involve second-order derivative information. In addition, these algorithms are implemented in a batch (multi-streaming) mode where weights are updated based on more than one training sample in the training set. This is in contrast to the conventional early versions of BP where weights are updated by involving only one training sample (a serial mode) [14].

1.2 State estimation-based ANN training

Even though second-order algorithms have proven to outperform the classical first-order BP, they may suffer from poor convergence properties due to problems with local minima [15]. Therefore, enhanced ANN training strategies based on state and parameter estimators such as Kalman filter have been extensively studied in the literature. The Kalman filter (KF) is the most popular state estimation tool. It provides a statistically optimal estimate for linear systems in the presence of Gaussian white noise. In the case of nonlinear systems, the extended Kalman filter is applied by linearizing the system or measurement matrices around the current state estimate at each time. An EKF-based neural network training technique was first introduced by Singhal and Wu [16]. The EKF provides a powerful neural network training capability compared to conventional first-order gradient-based algorithms, such as the BP [15]. In literature, the EKF has been widely applied for training of both feed-forward [17] and recurrent networks [18, 19] in both a global form (GEKF) and a decoupled form (DEKF). Although the EKF demonstrates a close performance compared to a second-order derivative, batch-based method, it can avoid local minima problems by encoding second-order information in terms of a state error covariance matrix [15]. Accordingly, the EKF represents an efficient and practical alternative to second-order training methods.

Various enhanced ANN training techniques have been proposed in several studies. A new hybrid learning algorithm that combines the EKF and particle filter has been presented in [20]. The new training scheme provides faster speed of convergence than the stand-alone EKF. An advanced EKF training technique has been proposed in [21]. The advanced form of Kalman filter-based parameter estimation method obtains a more accurate estimate of how a Gaussian distribution evolves under a nonlinear transformation. It has proven to offer performance advantages over standard EKF training. Reference [14] provides suggestions on how to initialize the EKF parameters in addition to presenting a new decoupling strategy that reduces the update rate of the error covariance matrix. Wan and van der Merwe [22] stated the effective use of the unscented Kalman filter (UKF) of Julier et al. [23] for feed-forward neural networks training.

1.3 The smooth variable structure filter

The smooth variable structure filter (SVSF) is a filtration strategy that is an extension to the sliding mode concept [24]. In the SVSF, an estimate of the state trajectory is generated and is forced toward the actual state trajectory. The estimates then remain within a neighborhood of the actual state trajectory, known as the existence subspace. The width of the existence subspace is a function of uncertainty in the filter model and varies with time. In the existence subspace, a switching gain is applied so that the state estimates switch back and forth along the true (desired) state trajectory. The period for the states to reach the existence subspace is known as the reachability phase [24].

The recently proposed SVSF provides a robust dynamic adaptation, high rate of convergence, and can guarantee estimation stability for bounded uncertainties and noise levels [24]. The SVSF has been successfully used for parameter and state estimation [25, 26]. The SVSF has demonstrated some advantages over the EKF in target tracking applications with respect to computational complexity, robustness, and tracking accuracy [27]. This is due to the sensitivity of the EKF to model uncertainties when used as a parameter estimator. The SVSF can be applied to both linear and nonlinear dynamic systems. Two versions of the SVSF have been developed in the literature: an original form of the SVSF without involving the state error covariance matrix $P_{k+1|k}$ and one that includes the covariance matrix [28]. In this research project, the original form of the SVSF will be discussed and applied to train feed-forward MLP networks. The proposed strategy has been applied to train ANN to detect engine fault conditions in [29].

In this paper, the effectiveness of the proposed strategy has been compared to well-known, widely used ANN training strategies on a benchmark, standard set of classification oriented problems that is used in assessing algorithms. The benchmark problem used is known as PROBEN1, which has been widely applied to assess ANN-based training strategies [30]. The SVSF is applied in a global (GSVSF), multi-streaming mode. The GSVSF's performance is compared against the standard first- and second-order derivative BP algorithms, as well as to the GEKF on a real-world benchmark problem.

2 Feed-forward multilayered neural network

A multilayer feed-forward network consists mainly of sensory units that constitute the input layer, one or more hidden layers, and an output layer.

As shown in Fig. 1, each node is connected to all nodes in the adjacent layer by links (weights) and computes a weighted sum of the inputs. An offset (bias) is added to the

resultant sum followed by a nonlinear activation function application. The input signal propagates through the network in a forward direction on a layer-by-layer basis. Consequently, the network represents a static mapping between inputs and outputs.

Let k denote the total number of layers, including the input and output layers. Node(n,i) denotes the ith node in the nth layer, and $N_n - 1$ is the total number of nodes in the nth layer. As shown in Fig. 2, the operation of node$(n+1,i)$ is described by the following equation [31]:

$$x_i^{n+1}(t) = \varphi\left(\sum_{j=1}^{N_n-1} w_{i,j}^n x_j^n(t) + b_i^{n+1}\right) \tag{1}$$

where $x_i^n(t)$ denotes the output of node(n,j) for the t training pattern, $w_{i,j}^n$ denotes the link weight from node(n,j) to the node$(n+1,i)$. b_i^n is the node offset (bias) for node(n,i) [31].

The function $\varphi(\cdot)$ is a nonlinear sigmoid activation function defined by [31]:

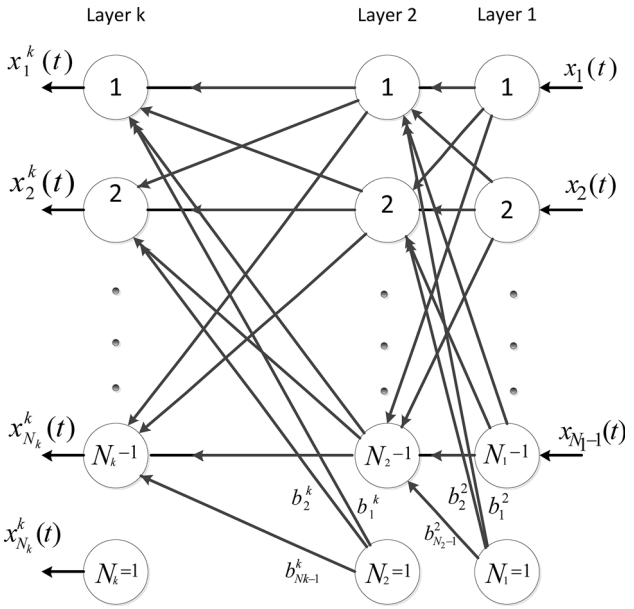

Fig. 1 Schematic of feed-forward multilayer perceptron network [31]

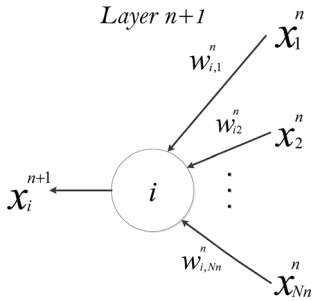

Fig. 2 Node $(n+1,i)$ representation [31]

$$\varphi(w) = \frac{1}{1+e^{-aw}} \quad a > 0 \text{ and } -\infty < w < \infty \tag{2}$$

For simplicity, the node bias is considered as a link weight by setting the last input N_n to node$(n+1,i)$ to the value of one as follows [31]:

$$x_{N_n}^n(t) = 1, \quad 1 \le n \le k$$
$$w_{i,N_n}^n = b_i^{n+1}, \quad 1 \le n \le k-1$$

Consequently, (1) can be rewritten in the following form [31]:

$$x_i^{n+1}(t) = \varphi\left(\sum_{j=1}^{N_n} w_{i,j}^n x_j^n(t)\right) \tag{3}$$

3 Global and decoupled EKF-based NN training

The EKF has been tailored to train feed-forward neural networks by formulating the network as a filtering problem [32]. Accordingly, feed-forward multilayer perceptron network behavior can be described by a nonlinear discrete-time state-space representation [33] such that:

$$w_{k+1} = w_k + \omega_k \tag{4}$$

$$y_k = C_k(w_k, u_k) + v_k \tag{5}$$

Equation (4) represents the system equation. It demonstrates the neural network as a stationary system with an additional zero-mean, white system noise ω_k with a covariance described by $\left[\omega_k \omega_l^T\right] = \delta_{k,l} Q_k$. Neural network weights and biases w_k are regarded as the system's state. Equation (5) is the measurement (observation) equation. It is a nonlinear equation relating network desired (target) response y_k to the network input u_k and weights w_k. The nonlinear function C_k represents the measurements function. A zero-mean, white measurement noise v_k is added with a covariance defined as $\left[v_k v_l^T\right] = \delta_{k,l} R_k$ [31].

Consider a feed-forward multilayer perceptron network with two hidden layers as shown in Fig. 3. All activation functions of the first, second, and output layers are nonlinear sigmoidal functions denoted as φ_I, φ_{II} and φ_o,

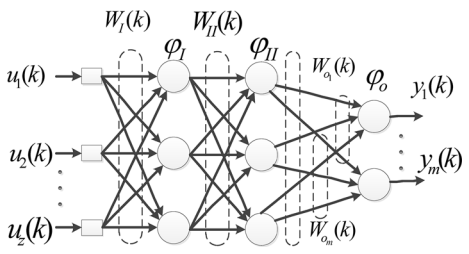

Fig. 3 Feed-forward multilayer perceptron with z inputs, two hidden, layers and m outputs [31]

respectively. The network transfer function in terms of network weights, inputs, and activation functions can be mathematically defined as [34]:

$$y_i(k) = \varphi_{\mathrm{o}}(w_{\mathrm{o}_i}(k)\varphi_{II}(w_{II}(k)\varphi_I(w_I(k)u(k)))) \tag{6}$$

where m denotes number of output neurons and $w_I, w_{II}, w_{\mathrm{o}}$ are group weight matrices for first hidden layer, second hidden layer, and output layer, respectively.

Linearization is performed by differentiating the network transfer function with respect to network synaptic weights (i.e., deriving the Jacobian). The Jacobian matrix $C_{k|\text{linearized}}$ can be mathematically expressed as follows [34]:

$$C_{k|\text{linearized}} = \begin{bmatrix} \frac{\partial y_1}{\partial w_1} & \frac{\partial y_1}{\partial w_2} & \cdots & \frac{\partial y_1}{\partial w_{N_T}} \\ \frac{\partial y_2}{\partial w_1} & \frac{\partial y_2}{\partial w_2} & & \frac{\partial y_2}{\partial w_{N_T}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial y_m}{\partial w_1} & \frac{\partial y_m}{\partial w_2} & \cdots & \frac{\partial y_m}{\partial w_{N_T}} \end{bmatrix} \tag{7}$$

where N_T denotes total number of synaptic weights (including bias) and z specifies number of input neurons. By differentiating (6) with respect to different weight groups $w_I, w_{II}, w_{\mathrm{o}}$, and for $i, l = 1, 2, \ldots, m$, the following is obtained [34]:

$$\frac{\partial y_i}{\partial w_{\mathrm{o}_l}} = \begin{cases} \acute{\varphi}_{\mathrm{o}}(w_{\mathrm{o}_i}\varphi_{II}(w_{II}\varphi_I(w_I u)))\varphi_{II}(w_{II}\varphi_I(w_I u)), & \text{if} \quad i = l \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$\frac{\partial y_i}{\partial w_{II}} = \acute{\varphi}_{\mathrm{o}}(w_{\mathrm{o}_i}\varphi_{II}(w_{II}\varphi_I(w_I u)))w_{\mathrm{o}_i}\acute{\varphi}_{II}(w_{II}\varphi_I(w_I u))\varphi_I(w_I u) \tag{9}$$

$$\frac{\partial y_i}{\partial w_I} = \acute{\varphi}_{\mathrm{o}}(w_{\mathrm{o}_i}\varphi_{II}(w_{II}\varphi_I(w_I u)))w_{\mathrm{o}_i}\acute{\varphi}_{II}(w_{II}\varphi_I(w_I u))w_{II}\acute{\varphi}_I(w_I u)u \tag{10}$$

By placing (8), (9), and (10) in one matrix [34]:

$$C_{k|\text{linearized}} = \begin{bmatrix} \frac{\partial y}{\partial w_{\mathrm{o}}} & \frac{\partial y}{\partial w_I} & \frac{\partial y}{\partial w_{II}} \end{bmatrix} \tag{11}$$

$C_{k|\text{linearized}}$ is the m-by-N_T measurement matrix of the linearized model around the current weight estimate.

The EKF-based neural network training introduced by Singhal and Wu [35] is known as the global extended Kalman filter (GEKF). In the GEKF algorithm, all network weights and biases are simultaneously processed and a second-order information matrix correlating each pair of network weights is obtained and updated [8]. Consequently, the GEKF computational complexity is $O(m\,N_T^2)$. A storage capacity of $O(N_T^2)$ is required, which is relatively high compared to the standard BP algorithm. The DEKF-based neural network training algorithm illustrated below represents the most general EKF-based neural network training method. The GEKF is a special form of the DEKF where the weight group number g is set to one. Neural network training using the DEKF algorithm can be expressed as follows [31]:

$$\Gamma_k = \left[\sum_{i=1}^{g}(C_k^i)^T P_k^i C_k^i + R_k\right]^{-1} \tag{12}$$

$$K_k^i = P_k^i C_k^i \mathbf{\Gamma_k} \tag{13}$$

$$\alpha_k = d_k - \hat{d}_k \tag{14}$$

$$\hat{w}_{k+1}^i = \hat{w}_k^i + K_k^i \alpha_k \tag{15}$$

$$P_{k+1}^i = P_k^i - K_k^i (C_k^i)^T P_k^i + Q_k^i \tag{16}$$

where the following nomenclature applies: $\mathbf{\Gamma}$, m-by-m matrix known as global scaling matrix (or global conversion factor); C, n_i-by-m gradient matrix, it involves weight gradient with respect to every output node; α, m-by-1 matrix representing innovation, which is the difference between desired (target) and actual network output; P, n_i-by-n_i error covariance matrix; Q, n_i-by-n_i process covariance matrix; K, n_i-by-m Kalman gain matrix; R, m-by-m measurement noise covariance matrix; $\hat{d}_k$, m-by-1 matrix representing actual network output; d_k, m-by-1 matrix representing target (desired) output.

The above DEKF training algorithm operates in a serial mode fashion. In serial mode, one training sample is involved in error calculation, gradients computation, and synaptic weight update. A problem known as the 'recency phenomenon' arises with serial mode when training tends to be influenced by the most recent samples [8]. Consequently, a trained network fails to remember former input–output mappings, and thus, serial mode training reduces training performance. The recency phenomenon can be circumvented using the multi-streaming training technique [36–38]. Multi-streaming EKF training allows multiple training samples to be batched and processed. It involves training M multiple identical neural networks using several training samples followed by weight update using overall networks' errors. The above algorithm can be adjusted to multi-streaming mode by replacing matrix dimension m in Γ, C, α, K, and R with $M \times m$ [14].

4 SVSF-based artificial neural network training

In 2007, the smooth variable structure filter (SVSF) was introduced based on variable structure theory and sliding mode concepts [24]. It implements a switching gain to converge the estimates to within a boundary of the true

states (i.e., existence subspace) [39, 40]. In its present form, the SVSF has been shown to be stable and robust to modeling uncertainties and noise [41, 42]. The basic estimation concept of the SVSF is shown in Fig. 4. The SVSF method is model based and may be applied to differentiable linear or nonlinear dynamic equations. The original form of the SVSF as presented in [24] did not include covariance derivations. An augmented form of the SVSF was presented in [28], which includes a full derivation for the filter.

The SVSF can be applied for training nonlinear feed-forward neural networks by estimating network weights [43]. In the same fashion as the Kalman filter, the SVSF has been adapted to train feed-forward neural networks by visualizing the network as a filtering problem where F, G, and C_k are the system, input, and output matrices, respectively, follows [24]:

$$\hat{w}_{k+1|k} = F\hat{w}_{k|k} + Gu_k \tag{17}$$

$$y_k = C_k(w_{k|k}, u_k) \tag{18}$$

The global SVSF training algorithm is iterative and is summarized by the following steps, assuming a training dataset defined by $\{x_k, z_k\}$:

Step 1 Network weights initialization
A priori state estimates (network weights) $\hat{w}_{k|k}$ are randomly initialized ranging from -1 to $+1$.

Step 2 Calculation of the predicted (a posteriori) weight estimates $\hat{w}_{k+1|k}$ from (17)
For neural networks training, the system matrix F is an identity matrix and the system input u_k is set to zero. Consequently, when the algorithm is initialized, the a posteriori weight matrix is the same as the a priori, and thus, (17) is rewritten as follows [24]:

$$\hat{w}_{k+1|k} = \hat{w}_{k|k} \tag{19}$$

Step 3 Jacobian matrix calculation (linearization) of the measurement matrix C_k
The algorithm for Jacobian matrix calculation is the same as that stated earlier in (7). After applying the algorithm, $C_{k|\text{linearized}}$ is obtained as in (11).

Step 4 Calculation of the estimated network output (measurements) $\hat{z}_{k+1|k}$
The linearized Jacobian measurement matrix $C_{k|\text{linearized}}$ and the a priori network weights $\hat{w}_{k+1|k}$ yield the estimated network output as follows [24]:

$$\hat{z}_{k+1|k} = C_{k|\text{linearized}}\hat{w}_{k+1|k} \tag{20}$$

Step 5 Measurement error $e_{z_{k+1|k}}$ calculation
Using the output $\hat{z}_{k+1|k}$ and the corresponding target (from the neural network training dataset) z_k, the measurement errors $e_{z_{k+1|k}}$ may be calculated as follows [24]:

$$e_{z_{k+1|k}} = z_k - \hat{z}_{k+1|k} \tag{21}$$

Step 6 SVSF gain calculation
The SVSF gain is a function of the a priori and the a posteriori measurement errors $e_{z_{k+1|k}}$ and $e_{z_{k|k}}$, the smoothing boundary layer widths ψ, the 'SVSF' memory or convergence rate γ, as well as the linear measurement matrix $C_{k|\text{linearized}}$. For the derivation of the SVSF gain K_{k+1}, refer to [24, 28]. The SVSF gain may be defined diagonally as follows [24]:

$$K_{k+1} = C^{+}_{k|\text{linearized}}\text{diag}\left[\left(\left|e_{z_{k+1|k}}\right| + \gamma\left|e_{z_{k|k}}\right|\right) \circ \text{sat}\left(\frac{e_{z_{k+1|k}}}{\psi}\right)\right] \times \text{diag}\left(e_{z_{k+1|k}}\right)^{-1} \tag{22}$$

Step 7 Calculation of the updated state estimates $\hat{w}_{k+1|k+1}$
The updated weights are calculated as follows [24]:

$$\hat{w}_{k+1|k+1} = \hat{w}_{k+1|k} + K_{k+1}e_{z_{k+1|k}} \tag{23}$$

Step 8 Calculation of a posteriori output estimate $\hat{z}_{k+1|k+1}$ and measurement errors $e_{z_{k+1|k+1}}$

Similar to the EKF strategy, the output estimates and a posteriori measurement errors are calculated, respectively, as follows [24]:

$$\hat{z}_{k+1|k+1} = C_{k|\text{linearized}}\hat{w}_{k+1|k+1} \tag{24}$$

$$e_{z_{k+1|k+1}} = z_{k+1} - \hat{z}_{k+1|k+1} \tag{25}$$

Steps 3–8 are iteratively repeated while shuffling (randomly shifting) the training dataset each epoch. Training

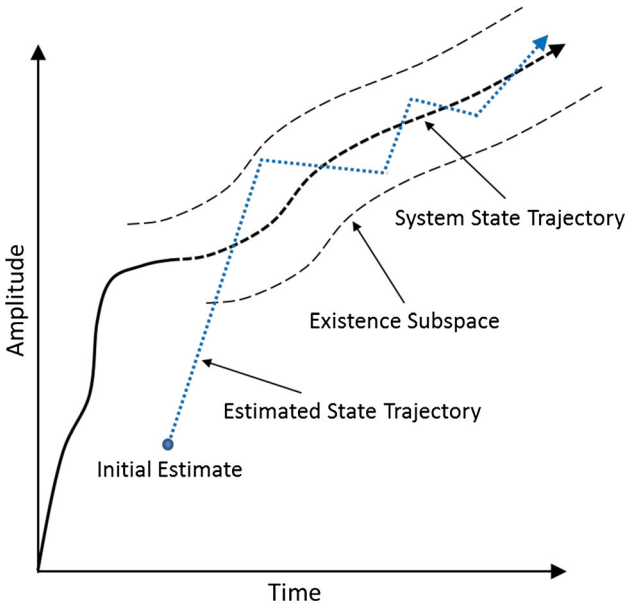

Fig. 4 The SVSF estimation concept [24]

proceeds until one of the stopping conditions (stated later) occurs. As per [24], the estimation process is stable and convergent if the following lemma is satisfied [24]:

$$|e_{k|k}| > |e_{k+1|k+1}| \tag{26}$$

The proof, as defined in [24], yields the derivation of the SVSF gain from (26). Expanding (16) using (15) and the standard SVSF gain yields the following [24]:

$$e_{z,k+1|k+1} = e_{z,k+1|k} - HK_{k+1} \tag{27}$$

Substitution of (27) into (26) yields [24]:

$$|e_{z,k|k}| > |e_{z,k+1|k} - HK_{k+1}| \tag{28}$$

Simplifying and rearranging (28) [24]:

$$|HK_{k+1}| > |e_{z,k+1|k}| + \gamma|e_{z,k|k}| \tag{29}$$

Based on the fact that $|HK_{k+1}| = HK_{k+1} \circ \text{sign}(HK_{k+1})$, the standard SVSF gain can be derived from (39) [24]:

$$K_{k+1} = H^{-1}\left(|e_{z,k+1|k}| + \gamma|e_{z,k|k}|\right) \circ \text{sign}(HK_{k+1}) \tag{30}$$

Equation (30) may be further expanded based on the fact that $\text{sign}(HK_{k+1}) = \text{sign}\left(e_{z,k+1|k}\right)$, as per [24], such that [24]:

$$K_{k+1} = H^{-1}\left(|e_{z,k+1|k}| + \gamma|e_{z,k|k}|\right) \circ \text{sign}\left(e_{z,k+1|k}\right) \tag{31}$$

Note further that the SVSF switching may be smoothed out by the use of a saturation function, such that (41) becomes [24]:

$$K_{k+1} = H^{-1}\left(|e_{z,k+1|k}| + \gamma|e_{z,k|k}|\right) \circ \text{sat}\left(e_{z,k+1|k}\right) \tag{32}$$

where the saturation function is defined by [24]:

$$\text{sat}\left(e_{z,k+1|k}\right) = \begin{cases} 1, & e_{z,k+1|k} \geq 1 \\ e_{z,k+1|k}, & -1 < e_{z,k+1|k} < 1 \\ -1, & e_{z,k+1|k} \leq -1 \end{cases} \tag{33}$$

Finally, a smoothing boundary layer ψ may be added to further reduce the magnitude of chattering, leading to [24]:

$$K_{k+1} = H^{-1}\left(|e_{z,k+1|k}| + \gamma|e_{z,k|k}|\right) \circ \text{sat}\left(e_{z,k+1|k}/\psi\right) \tag{34}$$

Note that the gain described in (34) is slightly different than that presented earlier as (22). A diagonalized form was created, as described in [28, 44], to formulate an SVSF derivation that included a covariance function. The form shown as (34) was presented as the original or 'standard' SVSF in [24].

5 Description of benchmark problems

PROBEN1 is a standard set of classification oriented problems, conventions, and guidelines that are used in assessing algorithms [30]. PROBEN1 is a collection of 15 datasets form 12 diverse fields that represents real-world data in both continuous and binary values. In addition, PROBEN1 includes a set of rules on how to conduct a benchmark neural network training test to allow easy comparisons of training algorithms. PROBEN1 consists of training datasets from the UCI (University of California, Irvine) learning database archive [30].[1]

In this paper, the datasets are further used for neural network training by splitting them into 50, 25, and 25 % segments representing training, validation, and testing sets, respectively. Three classification problems with different training difficulty levels were selected as follows: cancer, diabetes, and glass problems.

5.1 Cancer problem

Datasets were originally attained from Dr. W. H. Wolberg, University of Wisconsin Hospitals, Madison [45]. This problem represents classification of breast tumor to either malignant or benign. A total of 699 training datasets, with nine inputs and two outputs each, are collected under microscopic investigations. Inputs represent attributes used for cancer classification such as uniformity of cell size, shape, and clump thickness. Three forms of the same problem namely 'cancer1,' 'cancer2,' and 'cancer3' are presented according to the order of dataset. Datasets are split to 50, 25, and 25 % segments for training, validation, and testing. Table 1 shows different class distributions and percentages.[2]

5.2 Diabetes problem

This dataset is from the 'Pima Indians diabetes' folder from the UCI database archive. The problem tackles classification of 'Pima Indian' individuals to either diabetes positive or not. The datasets consist of 768 examples with eight inputs and two outputs. Inputs[3] represent personal and experimental data such as age, plasma glucose concentration, blood pressure, and glucose tolerance. 65.1 % of the examples are diabetes negative. Table 2 shows the distribution and percentage of samples.

5.3 Glass problem

The third dataset is fetched from the 'glass' problem in UCI database archive. This problem tackles classification of glass to one of six categories. Glass datasets consist of

[1] Data are available for download through (ics.ci.edu, directory/pub/machine-learning-database).

[2] There are 16 missing values for attribute (input) 6, and they are replaced by a constant value of 0.3 instead for network training.

[3] The datasets involve zero elements that might be replacing some missing attributes.

Table 1 Cancer problem class distribution

Description	Benign	Malignant	Total
Total number	458	241	699
Total percentage	66	34	100

Table 2 Diabetes problem class distribution

Description	Diabetes	No diabetes	Total
Total number	268	500	768
Total percentage	34	66	100

nine inputs: one represents the glass refractive index, and the remaining eight inputs represent percentage content of eight glass splinter chemical elements. Samples are classified to one of the followings: float processed, non-float processed building windows, vehicle windows, containers, tableware, and head lamps. Two hundred and fourteen datasets are used in this test. All of the inputs are continuous. This benchmark problem is quite challenging as only a few number of inputs are available. Hence, it tests the sensitivity of training algorithms to discard information. Table 3 shows the glass distribution of samples.

5.4 Problem attributes

The three benchmark problems previously stated characterize different degrees of classification difficulties. Cancer provides adequate number of training samples. Classes are partially overlapped with complex boundaries among classes. Hence, it is a fairly easy classification task [46]. Diabetes represents higher level of difficulty by overlapping classes as well as complex boundaries. Glass increases the level of difficulty by providing few training datasets in addition to complex boundaries and overlapping classes. Table 4 summarizes the training data attributes. For each benchmark problem, it shows the number of inputs, input attributes, output classes, number of training datasets, and the entropy. Input attributes are classified as: binary input b, continuous input c, and missing input attribute m. Output attributes are all binary for all problems under investigation. The entropy E is a measure of random variable uncertainty or unpredictability. As entropy increases, the level of unpredictability increases as well. Entropy for class probabilities $P(c)$ is defined as follows:

$$E = \sum_{\text{Classes}_C} P(c)\log_2(P(c)) \tag{35}$$

As shown in Table 4, cancer and diabetes are relatively predictable. Glass is highly unpredictable, which makes it a more difficult classification problem.

Table 3 Glass problem class distribution

Description	1	2	3	4	5	6	Total
Total number	70	76	17	13	9	29	214
Total percentage	32.7	35.5	7.9	6.1	4.2	13.6	100

Table 4 Attribute structure of classification problems

Problem	Input values			Classes b	Training set	E
	b	c	m			
Cancer	0	9	0	2	699	0.9
Diabetes	0	8	0	2	768	0.9
Glass	0	9	0	6	214	2.1

5.5 Benchmarking rules

In this paper, benchmarking rules stated in [30] are followed. Regarding training algorithms used, the first-order gradient-based batch BP algorithm is as stated in [47]. The algorithms used are covered in detail in Chapters 11 and 12 of [48]. The Levenberg–Marquardt algorithm applied in this paper is described in [49, 50]. The quasi-Newton technique is described in [51]. The EKF training algorithm applied is as stated in [37] and was also described earlier. The measurement noise covariance matrix R elements were set to 0.1, and the process noise covariance matrix Q elements were set to 0.01. The initial error covariance matrix $P_{0|0}$ elements are set to $0.1 \times I_{n\times n}$.

The SVSF algorithm, defined earlier, has a smoothing boundary layer thickness ψ set to 0.01 and γ was set to 0.2. The state error covariance matrix P, measurement noise covariance matrix R, and the process noise covariance matrix Q are not required as in the case of the EKF, and thus, fewer parameters are required to tune. By using the SVSF, only one tuning parameter is required that is the smoothing boundary layer thickness ψ. However, three tuning parameters are required for the EKF: the state error covariance matrix P, system noise covariance matrix Q, and measurement noise covariance matrix R. For the EKF and SVSF, weights and biases are updated once at the end of each epoch, using the multi-streaming approach as discussed earlier.

Training is stopped whenever one of the following three conditions occurs: if the GL_α stopping criterion discussed below is achieved, if the training progress is below a specified value 0.1, or if the maximum number of epochs (1000) are achieved. For the first stopping criterion, training is terminated when the generalization loss (GL) is higher than a certain threshold.

According to [30], the generalization loss at epoch t is defined as the relative increase in the validation error over the minimum-so-far (in percent).

$$GL(t) = 100\left(\frac{E_{\text{valid}}(t)}{E_{\text{Optimal}}(t)} - 1\right) \tag{36}$$

$$E_{\text{Optimal}}(t) = \min_{t' \le t} E_{\text{valid}}(t') \tag{37}$$

where $E_{\text{valid}}(t)$ is the validation set squared error at epoch t, where E is the error function defined in (40). $E_{\text{Optimal}}(t)$ is the least validation set squared error obtained up to epoch t. When high generalization loss occurs, it indicates that the algorithm is focusing on training signal peculiarities and ignoring general regular behavior. Accordingly, training should be terminated whenever a GL exceeds a predetermined threshold value α (set to 0.01).

The second stopping criterion is called the training progress. It is performed by calculating the squared error of training and validation dataset after a prespecified number of epochs called the strip length k (set to five epochs in this paper). The sequence is numbered as $n+1, n+2, \ldots n+k$ provided that n is divisible by k. Training progress is calculated using the following formula:

$$P_k(t) = 1000\left(\frac{\sum_{t' \in t-k+1 \ldots t} E_{\text{train}}(t')}{k \cdot \min_{t' \cdot \in t-k+1 \ldots t} E_{\text{train}}(t')} - 1\right) \tag{38}$$

This ratio specifies the relation between the average and the minimum training error that occurs within one strip. The network performance function (error function) used is the mean squared error between the network and target outputs. For the ith output node, the squared error per training sample is calculated as follows:

$$E(o,t) = \sum_i (o_i - t_i)^2 \tag{39}$$

where o_i is output target node and t_i is the target. Mean squared error (MSE) is calculated using the whole dataset. Error normalization is performed, and squared error percentage is calculated as follows [30]:

$$E = 100\frac{o_{\max} - o_{\min}}{N \cdot P}\sum_{P=1}^{P}\sum_{i=1}^{N}(o_{pi} - t_{Pi})^2 \tag{40}$$

where N represents the number of output nodes. P indicates the number of training samples used. $o_{\max}$ and $o_{\min}$ represent maximum and minimum values for the output nodes, respectively. In addition to MSE variations during training, training algorithms are evaluated according to classification performances [46].

Table 5 Architecture for the three benchmark problems

Problem	Inputs	Layer #1	Layer #2	Outputs
Cancer	9	12	12	2
Diabetes	8	11	10	2
Glass	9	12	11	6

6 Results of benchmark problems

A fully connected feed-forward multilayer perceptron with two hidden layers is used. Table 5 summarizes the networks' architecture used for the three proposed benchmark problems. A nonlinear sigmoidal activation function $\varphi(\cdot)$ as shown below is used in the hidden layers, while linear neurons are involved in the output layer.

$$\varphi(w) = \frac{1}{1+e^{-aw}} \quad a > 0 \quad \text{and} \quad -\infty < w < \infty \tag{41}$$

In this paper, multiple training and testing runs have been carried out followed by statistical analysis. This is due to the fact that random initialization arises in training a neural network. Consequently, different results may occur even by applying a similar training technique using the same training dataset. Accordingly, 30 runs have been carried out followed by calculating the mean, standard deviation, and the 'best' run using test, training, and validation set error.

Figures 5, 6, and 7 show the changing MSE over time for the three different cases using the same network initialization and architecture. For the three benchmark problems, the SVSF requires fewer epochs until convergence compared with Levenberg–Marquardt, quasi-Newton, and the batch first-order BP algorithm. The SVSF achieves comparable performance to the EKF in regards to the number of epochs required until convergence. For the cancer problem, at the second epoch, the SVSF has reached an MSE of 0.060, while the MSE for the EKF was 0.095. At the third epoch, the MSE for the SVSF was 0.024, and the EKF was 0.032. This makes the SVSF advantageous when online training is implemented. For the glass problem, at the second epoch, the SVSF has reached an MSE of 0.125, while the MSE for the EKF was 0.147. At the third epoch, the MSE for the SVSF was 0.095, and the EKF was 0.107.

For the diabetes problem, the SVSF has an MSE of 0.175 at the third epoch, compared with an MSE of 0.201 for the EKF. The quasi-Newton, Levenberg–Marquardt, and first-order BP achieved an MSE of 0.263, 0.279, and 0.670, respectively. The SVSF provides stability, as described by the stability lemma, and reaches the minimum MSE in the fewest number of epochs compared with the other popular training techniques.

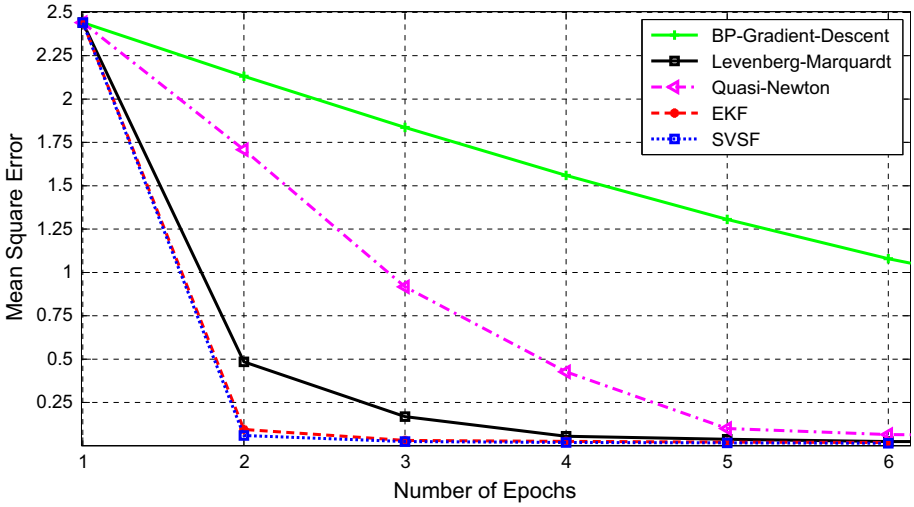

Fig. 5 Cancer mean square error variation versus number of epochs

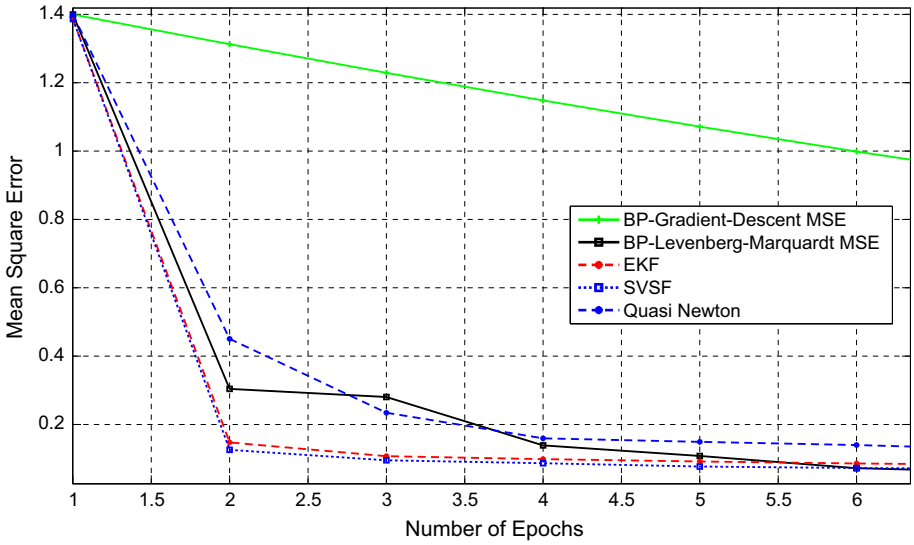

Fig. 6 Glass mean square error variation versus number of epochs

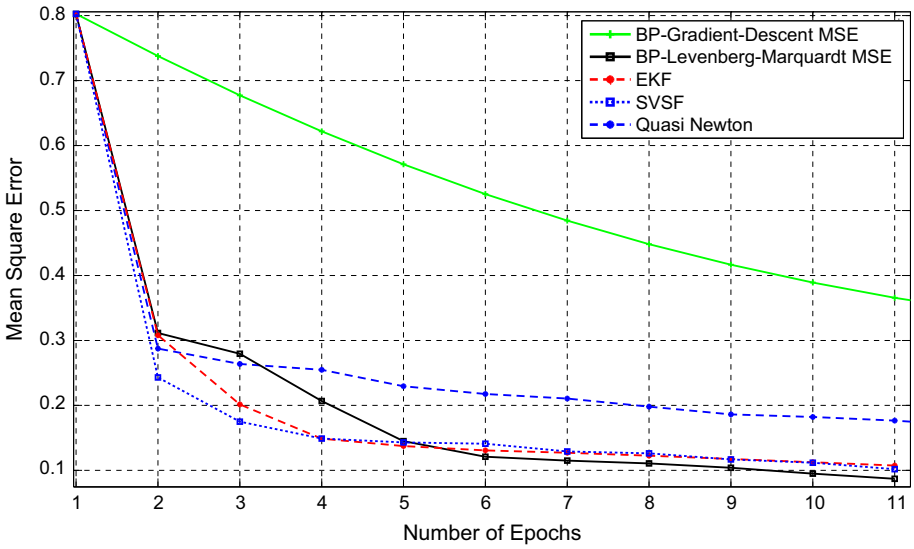

Fig. 7 Diabetes mean square error variation versus number of epochs

Tables 6, 7, and 8 show the results using the proposed SVSF training algorithm along with other popular training algorithms. The tables show training and generalization (testing) results on the three selected benchmark problems. Mean square error percentage is shown as the performance measure. For the cancer problem, the SVSF has shown better results compared to the EKF, LM, and BP. During testing, which is the most important performance measure as it tests trained networks generalization capability, the SVSF provides the best mean generalization after QN. The SVSF's mean testing error is close to quasi-Newton. However, QN algorithm requires a longer training time, as well as increased computational complexity, which provides an advantage for the SVSF.

In addition, the SVSF achieves the least standard deviation during testing compared with the EKF, LM, and BP. The SVSF's standard deviation is comparable to the QN. Regarding best and worst runs, SVSF, EKF, LM, and QN achieve the same best run followed by BP. Regarding the worst run, QN achieves the best performance, followed by the SVSF and EKF with the same performance. The LM method follows these, while the first-order batch BP is far behind. Regarding the number of epochs, the SVSF, EKF, and LM achieve the same number of epochs, followed by QN and first-order BP.

In this paper, an accurate SVSF-based feed-forward multilayer perceptron (MLP) training technique is developed and tested on several real-world applications. The SVSF, in addition to being an excellent parameter and state estimation strategy, can be tailored to train feed-forward MLP. The SVSF performance is tested on three real-world benchmark problems and compared to other classical training techniques, namely BP, quasi-Newton (QN), Levenberg–Marquardt (LM), and the extended Kalman filter (EKF). In general, the proposed technique achieves guaranteed stability, excellent static input–output mapping, good generalization capability, and minimum number of epochs compared to other classical, commonly used, training methods.

For the three benchmark problems (cancer, glass, and diabetes), simulation results show that the proposed SVSF training technique requires minimum number of epochs to reach convergence similar to the EKF and LM. In addition, results show that the SVSF-based ANN training technique

Table 6 Error rates using the cancer dataset for various training techniques

Technique	Training				Generalization				Epochs
	Mean	SD	Best	Worst	Mean	SD	Best	Worst	
BP	6.05	4.41	3.14	16.28	7.06	3.74	4.60	16.66	28
LM	1.40	0.71	0.57	2.57	5.34	1.67	3.44	8.62	13
QN	2.37	0.55	1.71	3.71	4.31	0.72	3.44	5.17	26
EKF	1.54	0.62	0.57	2.28	4.77	0.94	3.44	6.32	13
SVSF	1.85	0.52	0.85	2.85	4.65	0.73	3.44	6.32	13

Table 7 Error rates using the glass dataset for various training techniques

Technique	Training				Generalization				Epochs
	Mean	SD	Best	Worst	Mean	SD	Best	Worst	
BP	69.03	6.86	57.75	88.20	71.32	8.88	52.83	90.57	30
LM	29.48	6.83	17.39	52.17	40.50	7.18	32.08	64.15	12
QN	46.40	6.13	32.30	57.76	53.40	8.23	35.85	64.15	27
EKF	18.36	6.63	8.07	32.30	34.28	3.51	28.30	39.62	12
SVSF	25.61	8.53	14.91	62.11	35.66	4.01	28.30	49.05	12

Table 8 Error rates using the diabetes dataset for various training techniques

Technique	Training				Generalization				Epochs
	Mean	SD	Best	Worst	Mean	SD	Best	Worst	
BP	36.15	13.44	21.36	66.14	38.60	12.10	23.96	64.06	41
LM	18.86	2.64	14.32	24.74	25.40	3.01	18.75	33.33	10
QN	21.95	1.56	19.01	26.56	24.15	1.48	20.31	26.56	25
EKF	16.14	2.22	11.46	21.35	26.32	2.00	21.87	30.21	10
SVSF	18.40	6.03	14.32	47.66	26.44	4.40	22.92	47.92	10

provides comparable performance to other training techniques in terms of generalization capability, which is the most important aspect, especially for offline training as it tests the ability of trained networks to classify new data not previously seen during training phase.

For the cancer problem, QN achieved best generalization followed by the SVSF, and then, the EKF, LM, and the BP achieve worst performance. However, QN requires more epochs to train. For the glass problem, the EKF provides best generalization followed by the SVSF, then the LM, QN, and finally the BP. For diabetes problem, QN achieves the best generalization followed by the LM, EKF, SVSF, and BP. However, the QN requires more than double the number of epochs to train compared to EKF, SVSF, and LM.

The SVSF and EKF in their global form are computationally expensive compared to first- and second-order, optimization-based training techniques. However, the EKF and SVSF can avoid premature convergence to local minima problems by incorporating second-order information in the state error covariance matrix P. In addition to the state error covariance matrix, the SVSF can avoid local minima problems by using a switching chattering action in updating network's weights. By comparing the original SVSF to the EKF, the SVSF requires only one tuning parameter (boundary layer thickness ψ), while three tuning parameters are required in case of the EKF, namely the error covariance matrix P, system noise covariance matrix Q, and measurement noise covariance matrix R.

7 Conclusions

In this research, an accurate SVSF-based feed-forward multilayer perceptron (MLP) training technique was developed and tested on several real-world applications. The SVSF, in addition to being an excellent parameter and state estimation strategy, can be tailored to train feed-forward MLP. The SVSF performance was tested on three benchmark problems and was compared with other popular training techniques, namely BP, quasi-Newton (QN), Levenberg–Marquardt (LM), and the extended Kalman filter (EKF). In general, the proposed technique achieves guaranteed stability, excellent static input–output mapping, a good generalization capability, and a minimum number of epochs for convergence, which makes it an attractive method of training ANNs.

References

1. Runxuan Z (2005) Efficient sequential and batch learning artificial neural network methods for classification problems. Ph.D. Thesis, Nanyang Technological University, Singapore
2. Warner B, Misra M (1996) Understanding neural networks as statistical tools. Am Stat 50:284–293
3. Cybenko GV (1989) Approximation by superpositions of a sigmoidal function. Math Control Signals Syst 2:303–314
4. Patuwo E, Hu MY, Hung MS (1993) Two-group classification using neural networks. Decis Sci 24:825–845
5. Teixeira RA, Baraga AD, Menezes BD (2000) Control of a robotic manipulator using artificial neural networks with on-line adaptation. J Neural Process 12(1):19–31
6. Werbos P (1990) Backpropagation through time: what it does and how to do it. Proc IEEE 78:1550–1560
7. Haykin S (2007) Neural networks—a comprehensive foundation, 3rd edn. Prentice Hall, Englewood Cliffs, NJ
8. Haykin S (2001) Kalman filtering and neural networks, 3rd edn. Prentice Hall, Englewood Cliffs, NJ
9. Saarinen S, Bramley R, Cybenko G (1991) The numerical solution of neural-network training problems, CRSD report 1089. Center for Supercomputing Research and Development, University Illinois, Urbana, IL
10. Zhou G, Si J (1998) Advanced neural-network training algorithm with reduced complexity based on Jacobian deficiency. IEEE Trans Neural Netw 9(3):448–453
11. Hagen MT, Menhaj MB (1994) Training feedforward networks with the Marquardt algorithm. IEEE Trans Neural Networks 5:989–993
12. Watrous RL (1987) Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimization. In: IEEE conference on neural networks, vol 2, pp 619–627
13. Setiono R, Hui LCK (1995) Use of a quasi-Newton method in a feedforward neural network construction algorithm. IEEE Trans Neural Netw 6:273–277
14. Heimes F (1998) Extended Kalman filter neural network training: experimental results and algorithm improvements. In: Proceedings of the IEEE conference on systems, man, and cybernetics, vol 2, pp 1639–1644
15. Haykin S (2001) Kalman filtering and neural networks, ISBNs: 0-471-36998-5; 0-471-22154-6 ed
16. Singhal S, Wu L (1989) Training multilayer perceptrons with the extended Kalman algorithm. Adv Neural Inf Process Syst 1:133–140
17. Puskorius G, Feldkamp L (1991) Decoupled extended Kalman filter training of feedforward layered networks. In: Proceedings of the IJCNNÕ91 I, Seattle, pp 771–777
18. Williams R (1992) Training recurrent networks using the extended Kalman filter. In: Proceedings of the IJCNN'92 IV, pp 241–246
19. Sun P, Kenneth M (1998) Training recurrent neural networks for very high performance with the extended Kalman algorithm. Intell Eng Syst Through Artif Neural Netw 8:121–126
20. Deng X, Jianying X, Guo W, Liu J (2005) A new learning algorithm for diagonal recurrent neural network. In: First international conference on natural computation
21. Feldkamp L, Feldkamp T, Prokhorov D (2001) Neural network training with the nprKF. In: Proceedings of the international joint conference on neural networks
22. Wan EA, van der Merwe R (2000) The unscented Kalman filter for nonlinear estimation. In: Proceedings of the IEEE adaptive systems for signal processing, communications, and control symposium
23. Julier SJ, Uhlmann J, Durrant-Whyte HF (1995) A new approach for filtering nonlinear systems. In: Proceedings of the American control conference
24. Habibi SR (2007) The smooth variable structure filter. Proc IEEE 95(5):1026–1059
25. Habibi SR, Burton R Parameter identification for a high-performance hydrostatic actuation system using the variable structure

filter concept. J Dyn Syst Meas Control Trans ASME 129(2):229–235
26. Wang S, Burton R, Habibi SR (2007) A smooth variable structure filter for state estimation. Control Intell Syst 35(4):386–393
27. Gadsden SA, Habibi SR (2009) Target tracking using the smooth variable structure filter. In: ASME dynamic systems and control conference (DSCC)
28. Gadsden SA, Habibi SR (2010) A new form of the smooth variable structure filter with a covariance derivation. In: IEEE conference on decision and control, Atlanta, Georgia
29. Ahmed R, El Sayed M, Andrew Gadsden, JTASH (2013) Fault detection of an engine using a neural network trained by the smooth variable structure filter. In: IEEE Journal of Vehicular Technology
30. Prechelt L (1994) PROBEN1—a set of neural network benchmark problems and benchmarking rules. Technical Report, Karlsruhe, Germany
31. Iiguni Y, Sakai H, Tokumaru H (1992) A real-time learning algorithm for a multilayered neural network based on the extended Kalman filter. IEEE Trans Signal Process 40(4):659–966
32. Anderson BDO, Moore JB (1979) Optimal filtering. Prentice-Hall, Englewood Cliffs, NJ
33. Trebaticky P, Jiri P (2008) Neural network training with extended Kalman filter using graphics processing unit. In: ICANN
34. Le Yang YX (2006) Development of a new recurrent neural network toolbox. In: A Course Project Report on Training Recurrent Multilayer Perceptron and Echo State Network
35. Singhal S, Wu L (1989) Training multilayer perceptrons with the extended Kalman algorithm. Adv Neural Inf Process Syst 133–140
36. Feldkamp L, Puskorius G (1998) A signal processing framework based on dynamic networks with application to problems in adaptation, filtering, and classification. In: IEEE
37. Feldkamp LA, Puskorius GV (1994) Training controllers for robustness: multi-stream DEKF. In: IEEE international conference on neural networks, Orlando
38. Feldkamp LA, Puskorius GV (1994) Training of robust neurocontrollers. In: IEEE international conference on decision and control, Orlando
39. Gadsden SA, Habibi SR (2013) A new robust filtering strategy for linear systems. ASME J Dyn Syst Meas Control 135(1):014503-1-9
40. Gadsden SA, Song Y, Habibi SR (2013) Novel model-based estimators for the purposes of fault detection and diagnosis. IEEE/ASME Trans Mechatron 18(4):1237–1249
41. Habibi SR, Burton R (2003) The variable structure filter. J Dyn Syst Meas Control (ASME) 125:287–293
42. Habibi SR, Burton R (2007) Parameter identification for a high performance hydrostatic actuation system using the variable structure filter concept. ASME J Dyn Syst Meas Control 129(2). doi:10.1115/1.2431816
43. Ahmed R, El Sayed M, Gadsden SA, Habibi SR, Tjong J (2015) Engine fault detection and classification using artificial neural network techniques. IEEE Trans Veh Technol 64(1):21–33
44. Gadsden SA (2011) Smooth variable structure filtering: theory and applications. Hamilton, Ontario
45. Wolberg WH. Cancer dataset: Wiliams H. Wolberg, Center for machine learning and intelligent systems. University of California, Irvine
46. Heinke D, Hamker F (1998) Comparing neural networks: a benchmark on growing neural gas, growing cell structures, and fuzzy ARTMAP. IEEE Trans Neural Netw 9:1279–1291
47. Rumelhart DE, Hinton GE, Williams RJ, Rumelhart DE, McClelland J (1986) Learning internal representations by error propagation, vol 1. MIT Press, Cambridge, MA, pp 318–362
48. Hagan MT, Demuth HB, Beale MH (2002) Neural network design. Natick, MA
49. Levenberg K (1944) A method for the solution of certain problems in least squares. Q Appl Math 2:164–168
50. Marquardt D (1963) An algorithm for least-squares estimation of nonlinear parameters. SIAM J Appl Math 11:431–441
51. Dennis J, Schnabel R (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs, NJ