Comparison of Gradient descent method, Kalman Filtering and decoupled Kalman in training Neural Networks used for fingerprint-based positioning

Claude Mbusa Takenga, Koteswara Rao Anne, K. Kyamakya, Jean Chamberlain Chedjou
Institute of Communications Engineering, University of Hannover
Appelstr. 9A D-30167, Hannover – Germany
{takenga, raoanne, kyandogh, chedjou}@ant.uni-hannover.de

Abstract
The success of neural network architectures depends heavily on the availability of effective learning algorithms. Radial basis function (RBF) neural networks provide attractive possibilities for solving signal processing and pattern classification problems. Gradient descent training (GD) of RBF networks has proven to be much more effective than more conventional methods. However, gradient descent training can be computationally expensive and its learning speed is very slow. This paper compares (GD) to the method based on either Kalman filtering (KF) or decoupled Kalman filter (DEKF). These new methods prove to be quicker than gradient descent training while still providing good performance at the same level of effectiveness as they are used in fingerprint-based positioning.

Keywords
Neural Network training, Kalman Filter, Gradient Descent, Decoupled Kalman Filter, Positioning

I. INTRODUCTION

The problem of providing a reliable and accurate position of a mobile station (MS) in wireless communication systems has attracted a lot of attention in recent years. Positioning systems can be roughly classified into three major categories: systems using signal strength measurements, systems using either time of arrival and angle of arrival of a radio signal, and systems using dead reckoning techniques. The first two categories can be called radio-location methods as they rely on the propagation properties of radio signals. This paper addresses a novel approach while applying signal strength measurements for the positioning of a GSM-based mobile station. It uses radio signal strengths from the serving and neighboring base stations, which are continuously measured in the mobile station. These are applied to a previously trained artificial neural network for positioning. Fingerprint based Positioning using neural networks can be made using either classification or function approximation. When using classification, the area of interest is divided into small sections and the identification of the section in which the mobile station can be found is the task performed by the classification. For the second case, a function approximates the relationship between the received signal strengths and the distance between the mobile station and the antenna. Trilateration can subsequently be made when at least three distances to known base station locations are known. This is an indirect positioning. Furthermore, the received signal strength input can also be used to directly model the two-dimensional coordinates of the mobile station. This method is referred to as direct positioning. [1]

Park and Sanberg [2] have proven that Radial Basis Function (RBF) neural networks with one layer of RBF functions are capable of universal approximation. For this reason this paper uses a RBF architecture with one hidden layer of neurons in order to compare three key training approaches: Gradient Descent (GD), Extented Kalman Filter (EKF), and Decoupled Extended Kalman Filter.
For fingerprint based positioning, a method that is used in this paper, the position of the mobile system is automatically found knowing the set of signal strength for that point. GPS positions are used as references during the training phase [3]. For this purpose, one need an element or a system capable to relate the received GSM power levels of the surrounding cells in a point and the position of this point given by GPS. This element has to acquire enough intelligence and abstraction of such a not obvious relation, and use it to make future position predictions related to given signal strenghs measurements. A neural network (NN) is used for that purpose.

II. DESCRIPTION OF THE THREE TRAINING METHODS USED: DG, EKF, DKF

There are several algorithms available for training the weights of a neural network [4] [5] [6] . Most of them are based on computation of the gradient of an output error measured with respect to network weights. Recently, several authors [7][8][9][10] have noted that the Kalman Filters (e.g. EKF) can also be used for training networks to perform the desired input-output mappings.

In this paper we use the RBF neural network architecture as it provides attractive possibilities for solving signal processing and pattern classification problems [11]. An architecture of one hidden-layer of RBFs is used to compare the three methods (GD,EKF and DEKF).

The RBF NN can be described as follows. The input data is (are) represented by x in Fig.1, being passed directly to hidden layer. Suppose there are c neurons in the hidden layer. Each of the c neurons in the hidden layer applies an activation function, which is a function of the Euclidean distance (i.e. the square of the Euclidean norm of the two vectors) between the input and the prototype vectors v, as shown in Fig.1. There are many choices for g(.), function in the hidden layer of RBF NN. The most common choice is a Gaussian function of the form

$$g(v) = e^{(-v/\beta^2)} \tag{1}$$

where $g(v)$ is the Gaussian function, β is a real constant, [11] Another choice is the inverse multiquadratic function

$$g(v) = (v^2 + \beta^2)^{-1/2} \tag{2}$$

where β is a real constant [2].

A further choice of the $g(.)$ function is

$$g(v) = [g_0(v)]^{1/(1-p)} \tag{3}$$

where $g_0(v) = av + b$ (4)

g_0 is called generator function, p is a real number greater than 1, a>0 and b ≥ 0 , [6]. If *a=1* and *p=3*, the hidden layer function is reduced to the inverse multiquadratic function.

One output of the RBF in Fig.1 can be written as follows:

$$\hat{y} = \begin{bmatrix} w_{10} & w_{11} & \dots & w_{1c} \\ w_{20} & w_{21} & \dots & w_{2c} \\ \dots & \dots & \dots & \dots \\ w_{n0} & w_{n1} & \dots & w_{nc} \end{bmatrix} \begin{bmatrix} 1 \\ g(\| x - v_1 \|^2) \\ \dots \\ g(\| x - v_c \|^2) \end{bmatrix} \tag{5}$$

For all the outputs

$$\begin{bmatrix} \hat{y}_1 & \dots & \hat{y}_M \end{bmatrix} = W \begin{bmatrix} 1 & \dots & 1 \\ g(\| x_1 - v_1 \|^2) & \dots & g(\| x_M - v_1 \|^2) \\ \dots & \dots & \dots \\ g(\| x_1 - v_c \|^2) & \dots & g(\| x_M - v_c \|^2) \end{bmatrix} \tag{6}$$

Thus, let's write : $\hat{Y} = WH$ [11] . (7)

In the following the three different training policies are described in detail.

A. Gradient Descent Training Method

To use the gradient descent to minimize the training error, one does define an error function,

$$E = \tfrac{1}{2} \| Y - \hat{Y} \|_F^2 \tag{8}$$

where Y is the matrix of desired values for the RBF output, and $\| . \|_F^2$ is the square of the Froebinius norm of a matrix, which is equal to the sum of the squares of the elements of the matrix. It has been shown; see Ref. [2], which is given in this case by:

$$\frac{\partial E}{\partial w_i} = \sum_{k=1}^{M} (\hat{y}_{ik} - y_{ik}) h_k \quad (i = 1 \dots n), \tag{9}$$

$$\frac{\partial E}{\partial v_j} = \sum_{k=1}^{M} 2g'(\| x_k - v_j \|)(x_k - v_j) \sum_{i=1}^{n} (y_{ik} - \hat{y}_{ik}) w_{ij} \quad (j = 1 \dots c) \tag{10}$$

where $\hat{y}_{ik}$ is the element in the i_{th} row and k_{th} column of the $\hat{Y}$ matrix of Eq.7. and y_{ik} is the corresponding element in the Y matrix.

The RBF can be optimized by performing the following updates of weights (w) and prototypes (v):

$$w_i = w_i - \eta \frac{\partial E}{\partial w_i} \quad (i = 1 \dots n), \tag{11}$$

$$v_j = v_j - \eta \frac{\partial E}{\partial v_j} \quad (j = 1 \dots c) \tag{12}$$

where η is the step size of the gradient descent method. This optimization stops when w_i and v_j reach local minima [11].

B. Extended Kalman Filter (EKF) Training Method

The Kalman Filter is also used to train a general multi-input, muti-output RBF networks. For linear dynamic systems with white noise process and white measurement noise, the Kalman filter is known to be an optimal estimator [12]. For nonlinear systems with colored noise, the Kalman filter can be extended by linearizing the system around the current parameter estimates [6, 12]. There are several algorithms available for training the weights of the NN. Most of them are based on computation of the gradient of an output error measured with respect to the network weights. Recently, several authors [4, 9] have noted that the extended Kalman filter can also be used for the purpose of training networks to perform desired input-output mappings.

Assume the state space model below, [12]:

$$x_{k+1} = f(k, x_k) + w_k \tag{13}$$

$$y_k = h(k, x_k) + v_k \tag{14}$$

where x is the state of the system, y is the measurement model, w_k and v_k are independent, zero-mean, Gaussian noise processes of the covariance matrices Q_k and R_k respectively.

The first step of EKF is computing the linearized state matrices :

$$F_{k+1} = \frac{\partial f(k,x)}{\partial x}|_{x=x_k} \quad (15)$$

$$H_k = \frac{\partial h(k,x)}{\partial x}|_{x=x_k^-} \quad (16)$$

Once matrices $F_{k+1,k}$ and H_k are evaluated, they are then used in first order Taylor approximation of nonlinear functions.

$$F(k,x_k) \approx F(x,\hat{x}_k) + F_{k+1,k}(x,\hat{x}_k), \quad (17)$$

$$H(k,x_k) \approx H(x,\hat{x}_k^-) + F_{k+1,k}(x,\hat{x}_k^-), \quad (18)$$

The training problem using Kalman filter theory can now be described as finding the minimum mean-squared error estimate of the state *x* using all observed data.

When assuming that $w_{k+1} = w_k + \omega_k$ is the state of the neural network, and $y_k = h_k(w_k, u_k, v_j) + v_k$ the observation or measurement equation which represents the network's desired response vector y_k as a nonlinear function of input vector u_k, the weight parameter vector w_k, and for the RBF the prototype vector parameter v_j.

The solution to the training problem is given by the following recursion [12] :

$$A_k = \left[R_k + H_k^T P_k H_K\right]^{-1}, \quad (19)$$

$$K_k = P_k H_k A_k, \quad (20)$$

$$\hat{w}_{k+1} = \hat{w}_k + K_k \xi_k, \quad (21)$$

$$P_{k+1} = P_k + K_K H_k^T P_k + Q_k. \quad (22)$$

This Kalman recursion process can be explained with following words. An input training pattern u_k is propagated through the network to produce an output vector $\hat{y}_k$. The derivative matrix H_k is obtained, then tke Kalman gain matrix is computed according to Eq.20, This step include the computation of the global scaling matrix A_k. The network weights vector is updated using the Kalman gain matrix, the error vector ξ_k, and the current value of the weight vector $\hat{w}_k$ as in Eq.21. At the end, the approximate error covariance matrix is updated as in Eq.22.

In order to apply the optimization problem in a form suitable for Kalman Filtering in the case of a RBF NN, we let the elements of the weight matrix (*w*) and the elements of the prototypes (*v*) constitute the state of the nonlinear system. And the output of the RBF network constitutes the output of the nonlinear system. The state of the nonlinear system model is represented by $X = [w_1 ... w_n v_1 ... v_c]$, [11].

The computational effort of Kalman is in the order of $O[(AB)^2]$, where A is the dimension of the output dynamic system and B is the number of parameters. In the case of concern in this paper, there are nM outputs and $[n(c+1)+mc]$, i.e., $[n(c+1)]$ weights and mc-prototypes, where n is the dimension of the RBF output, M is the number of training samples, c is the number of prototypes (*v*) --see Fig.1, and m is the dimension of the RBF input. Therefore, the computational expense of Kalman filter is in the order of $O\left(nM[n(c+1)+mc]^2\right)$. [11].

C. Decoupled Extended Kalman filter (DEKF)Training Method

The classical disadvantage of Extended Kalman Filter is its computational expense, which is the obstacle for its use for larger networks. Thus, simplified variants must be found, that preserve the most useful property of the EKF while requiring less computation per time step [13]. The parameter-based DEKF algorithm is derived from EKF by assuming that the iterations between certain weight estimates can be ignored. This simplification introduces many zeros into the matrix P_k. If the weights are decoupled in a way such that the weight groups become mutually exclusive of one another, then P_k can be arranged into a block-diagonal form. Let's g refers to the number of such weight groups. Then, for group i, the vector $\hat{w}_k^i$ refers to the estimated weight parameters, H_k^i is the sub-matrix of derivatives of network outputs with respect to the i_{th} group's weights, p_k^i is the weight group's approximate error covariance matrix, and K_k^i is its Kalman gain matrix.

The DEKF for the i_{th} weight group is given, see [12] The computational training expense of Kalman filter is reduced in the order of $O\left(nM\left[(c+1)^2+(mc)^2\right]\right)$, [11].

The ratio between the computational training expense of the EKF and that of the DEKF is in the order of: $O\left(\frac{[n(c+1)+mc]^2}{(c+1)^2+(mc)^2}\right)$.

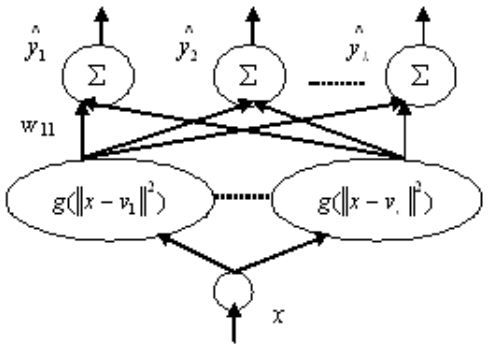

Figure 1. Radial Basis Function (RBF).

III. SYSTEM DESCRIPTION AND EXPERIMENTAL SETTING

Signal strength measurements in a GSM mobile terminal are the input data used to train a RBF neural network using one of three algorithms (GD, EKF, DEKF), whereby a target position (or sector) is given for every point. During the training phase, the neural network does realize a mapping between the signal strengths and target positions (sector) for all sample data provided.

In the testing phase, providing a set received signal strengths at the input of the NN, the position (Sector) of the current mobile station is predicted at the output.
For the experiments conducted in this paper for illustration, we do use measurement data collected on one street of the town Hanover (Schneiderberg Str.) in Germany. The street has a total length of 450 meters. While moving with a constant speed, RSSI values were being collected every 5 seconds. Thus, 150 data points could be recorded. Each data point contains the RSSI values from the 4 strongest neighboring cells.
The classification method is used for the NN training. The street is divided into 15 equal parts (segments). Thus, every segment has a length of 30 meters and does contain 10 successive of the collected data points.
The NN used in the experiments, consisted of four input vectors (corresponding to the RSSI measurement for the 4 strongest cells) and 15 outputs, each output corresponding to one of the 15 sectors. The number of RBF functions has been varied from 1 to 50. The learning rate in GD is taken equal to 0,002 in the first experiment where by a number of iteration is counted for every fixed number of neurons in the hidden layer, as this has guarantied a monotonic reduction of the error during the training process. The RBF network were trained using the hidden layer function of Eq.3. with the linear generator function of Eq.4. The Kalman filter parameters of Eqs.19-22 have been initialized with $P_o = 40I$, $Q = 40I$, *and* $R = 40I$, where I is the identity matrix of appropriate dimensions. the parameters for DEKF (P_o^i, Q^i, R^i) were initialized in a similar way [11].

IV. SIMULATION RESULTS

Fig.2 does present a comparison of the positioning performance of NN trained using the three different schemes. The three training algorithms (GD, EKF and DEKF) were terminated when the error function of Eq.8 decreased by less than the given training error. The number of neurons in the hidden layer was varied. It appeared that the methods based on Kalman Filtering (EKF, DKF) are providing less errors. For example, with the positioning quadratic average error with Kalman method is less than 40 meters with a probability of 67%. For GD, however, the positioning quadratic average error is up to 65 meters with a probability of 67%.
The difference in accuracy between EKF and GD becomes less, as the number of neurons in the hidden layer is increased.
The EKF and DEKF are providing the same accuracy as they are both based on Kalman Filtering, see Figs.2-3. Since the DEKF algorithm is in principle derived from EKF in that it is assumed that the connections between certain weight estimates can be ignored, it consequently requires fewer operations in one iteration (if compared to EKF). If one does record the time needed for the training process, the difference in terms of training effort between EKF and DEKF will become clearer, especially if the NN is large (a large number of parameters: inputs, output, weights, prototypes. The DKF method is generally preferred because of its saving of training time. For Fig.3, for example, the training time could be recorded and is used here for comparison. Following parameter setting has been used: a RBF NN of 30 neurons, with 4 inputs vectors, each having 150 samples; and 15 ouputs (classification ones). The computer platform used is a Pentium IV (2 GHZ and 512 MB RAM). The EKF training required 12 minutes, whereas the DEKF ones needed only 8 minutes, both for 11 iterations.

Fig.4 compares the number of iterations to converge the training error up to a fixed threshold. It is seen that methods based on Kalman filtering do converge in fewer iterations compared to the GD, provided a given training error and for a fixed number of neurons in the hidden layer.

V. CONCLUSION

This paper has compared a well-known Gradient descent method training of a NN to the ones based on Kalman filtering (EKF and DEKF). The application scenario in this work is a fingerprint positioning using GSM RSSI

data. The experiments conducted in this work have shown that the Kalman filtering based training of the NNs does lead to a better positioning performance while requiring the lowest training effort. The computational savings achieved by the DEKF method compared to EKF will be more significant for cases where large NN are needed.

REFERENCES

[1] Z. Salcic and E. Chan, "Mobile station positioning using GSM cellular phone and artificial neural networks," *Wireless Personal Communications*, vol. 14, pp. 235-254, 2000.

[2] J. Park and I. W. Sanberg, "Universal approximation using radial-basis function networks," *Neural Computation*, vol. 3, pp. 246-257, 1991.

[3] K. Kyamakya, *DOM-Der orientierte Mensch*: SHAKER VERLAG, 2003.

[4] S. Sin and R. D. Figueiredo, "Efficient learning procedures for optimal interpolative nets," *Neural Networks*, vol. 6, pp. 99-113, 1993.

[5] R. Duro and J. Reyes, "Discrete-time backpropagation for training synaptic delay-based artificial neural networks," *IEEE Trans on Neural Networks*, vol. 10, pp. 779-789, 1999.

[6] M. Vidyasagar, *Learning and generalization with applications to Neural networks*, second edition ed: Springer, 1997.

[7] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," *IEEE Trans. Neural Networks*, vol. 05, pp. 279-297, 1994.

[8] M. Birgmeier, "A fully Kalman-trained radial basis function network for nonlinear speech modeling," presented at IEEE International Conference on Neural Networks, 1995.

[9] R. J. Williams, "Training recurrent Networks using the extended Kalman Filter," presented at International Joint Conference on Neural Networks, 1992.

[10] J. Sum, C. Leung, G. Young, and W. Kan, "On the Kalman filtering method in neural network training and pruning," *IEEE Transactions on Neural Networks*, vol. 10, pp. 161-166, 1999.

[11] D. Simon, "Training Radial Basis Neural Networks with the Extended Kalman Filter," *Neurocomputing*, vol. 48, pp. 455-475, 2002.

[12] S. Haykin, *Kalman filtering and neural networks*: John Wiley & Sons, inc., 2001.

[13] S. Haykin, *Neural networks, a comprehensive foundation*, 2nd edition ed: Prentice Hall, 1999.

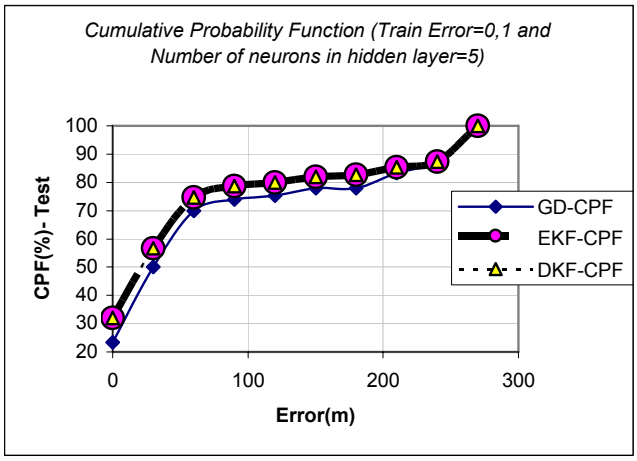

Figure 2. Cumulative Probability Functions (CPF) of the positioning error for NNs trained with the three schemes.

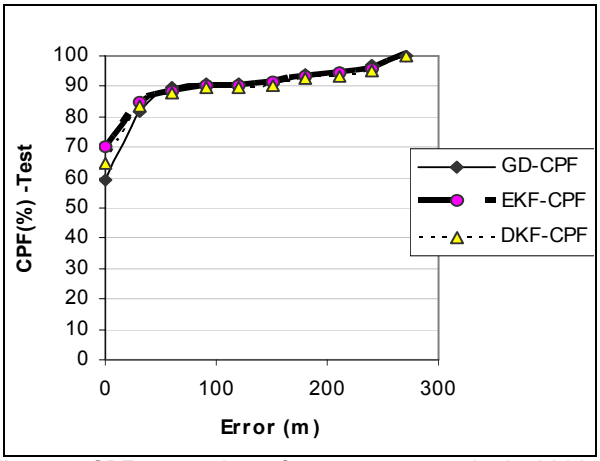

Figure 3. CPF comparison for more neurons in the hidden layer (Training error = 0.1; 30 neurons).

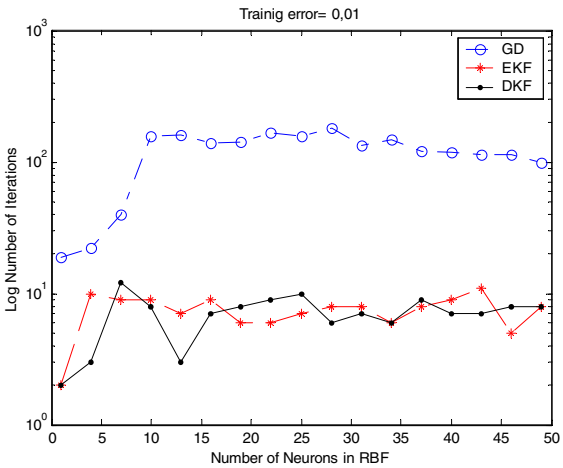

Figure 4. Comparison of the number of iterations in training (GD,EKF,DKF) the neural networks while varying the number of neurons in the hidden layer.