Journal of Engineering and Applied Sciences 1 (4): 530-537 2006

Tool Wear Monitoring by Using Extended Kalman Filter with Functional Update

[1]J. Hameed Hussain, and [2]S. Purushothaman
[1]Department of Mechanical Engineering Sathyabama Deemed University,
Old Mahabalipuram Road, Chennai, 603103, India
[2]Department of Mechanical Engineering Hindustan College of Engineering
Old Mahabalipuram Road Chennai, 603103, India

Abstract: Artificial Neural Network (ANN) is applied for pattern recognition of the tool wear in lathe. Conventional back-propagation algorithm which uses Steepest-Descent Method (SDM), is applied to train the ANN. One of adaptive algorithms which is the Extended Kalman Filter (EKF) algorithm to train the ANN, especially for XOR problem. The ANN has been trained using EKF and EKF with functional update method. To show the supremacy of EKF over SDM, the results of EKF algorithm is found to be faster than the convergence speed of SDM. The performance of EKF algorithm is almost the same as the performance of SDM algorithm. Experiments were conducted on a lathe to collect cutting force data and tool flank wear land width for various machining conditions. The network was trained offline. Fresh patterns were tested by using the weights and thresholds obtained during training. The classification performance of EKF algorithm for the test patterns is above 75%.

Key words: Steepest-Descent Method (SDM), Extended Kalman Filter (EKF), Functional Update Method (FUM)

INTRODUCTION

In this study, a distributed neural network is used to a pattern recognition problem for classification of tool wear in turning, in order to discriminate between acceptable wear of the tool and completely worn-out tool. Tool failure contributes to about 7% downtime of machine tools[1-3]. Tool failure can be due to fracturing or by gradual wear. There are a number of reasons for installing a monitoring system in a manufacturing process. Some of them are to run the machine without any interruption due to the breakage of the tool tip. To achieve this, breakage of the tool has to be sensed and replaced with a new tool. When it is possible to identify the tool breakage, it will help in preventing fatal damage to the system, thus minimizing rejection of study pieces[4].

Tool wear can be monitored by measurements of process variables by using several sensing devices. Tool wear sensing methods can be direct (optical, wear particles and radioactivity test and tool/ work distance) and indirect (cutting force, acoustic emission, sound, vibration, temperature, power input and surface roughness of machined work piece)[5]. In addition to measurement of these variables, strategies are required to control the machining operations on-line automatically. This can be implemented with rule based systems[6], such as statistical techniques (group method data handling, multiple regression analysis) and artificial neural networks.

Artificial neural networks: A neural network is constructed by highly interconnected processing units (nodes or neurons) which perform simple mathematical operations[7]. Neural networks are characterized by their topologies, weight vectors and activation function which are used in the hidden layers and output layer[8]. The topology refers to the number of hidden layers and connection between nodes in the hidden layers. The activation functions that can be used are sigmoid, hyperbolic tangent and sine[9]. The network models can be static or dynamic[10]. Static networks include single layer perceptrons and multilayer perceptrons. A perceptron or adaptive linear element (ADALINE)[11] refers to a computing unit. This forms the basic building block for neural networks. The input to a perceptron is the summation of input pattern vectors by weight vectors. In Fig. 1, the basic function of a single layer perceptron is shown.

Corresponding Author: Hameed Hussain, J., Department of Mechanical Engineering Sathyabama Deemed University, Old Mahabalipuram Road, Chennai, 603103-India

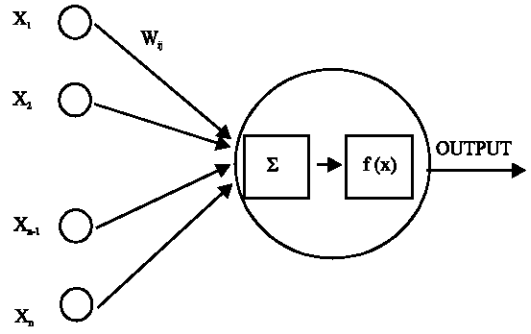

Fig. 1: Operation of a neuron

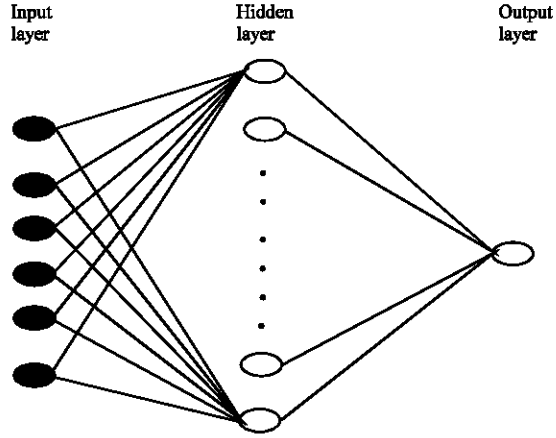

Fig. 2: Multilayer perceptron

In Fig. 2, a multilayer perceptron is shown schematically. Information flows in a feed-forward manner from input layer to the output layer through hidden layers. The number of nodes in the input layer and output layer is fixed. It depends upon the number of input variables and the number of output variables in a pattern. In this study, there are six input variables and one output variable. The number of nodes in a hidden layer and the number of hidden layers are variable. Depending upon the type of application, the network parameters such as the number of nodes in the hidden layers and the number of hidden layers are found by trial and error method[12-14]. For most of the applications one hidden layer is sufficient. The activation function which is used to train the ANN, is the sigmoid function and it is given by:

$$f(x) = \frac{1}{1+\exp(-x)} \quad (1)$$

where f (x) is a non - linear differentiable function,

$$x = \sum_{i=1}^{N} W_{ij}(p) x_i^n(p) + \theta(p),$$

Nn is the total number of nodes in the nth layer

W_{ij} is the weight vector connecting ith neuron of a layer with the j^{th} neuron in the next layer.

θ is the threshold applied to the nodes in the hidden layers and output layer and

p is the pattern number.

For the first hidden layer, x_i is treated as an input pattern vector and for the successive layers, x_i is the output of the ith neuron of the proceeding layer. The output x_i of a neuron in the hidden layers and in the output layer is calculated by :

$$X_i^{n+1}(p) = \frac{1}{1+\exp(-x)} \quad (2)$$

For each pattern, error E (p) in the output layers is calculated by :

$$E(p) = \frac{1}{2} \sum_{i-1}^{N_M} (di(p) - x_i^M(p))^2 \quad (3)$$

where M is the total number of layer which include the input layer and the output layer,

N_M is the number of nodes in the output layer.

$d_i(p)$ is the desired output of a pattern and

$X_i^M(p)$ is the calculated output of the network for the same pattern at the output layer.

The total error E for all patterns is calculated by :

$$E = \sum_{p-1}^{L} E(p) \quad (4)$$

where L is the total number of patterns.

Disadvantages of steepest-descent method: The number of cycles required for E to reach the desired minimum is very large. The E does not reach the desired minimum due to some local minima whose domains of attraction are as large as that for the global minimum. The algorithm converges to one of those local minima and hence learning stops prematurely or the value diverges. The updating of weights will not stop unless every input is outside the significant update region. The significant update region is from 0.1 to 0.9. Due to this, the output of the network will be approaching either 0.0 or 1.0. This requires a large number of iterations for the convergence of the algorithm.

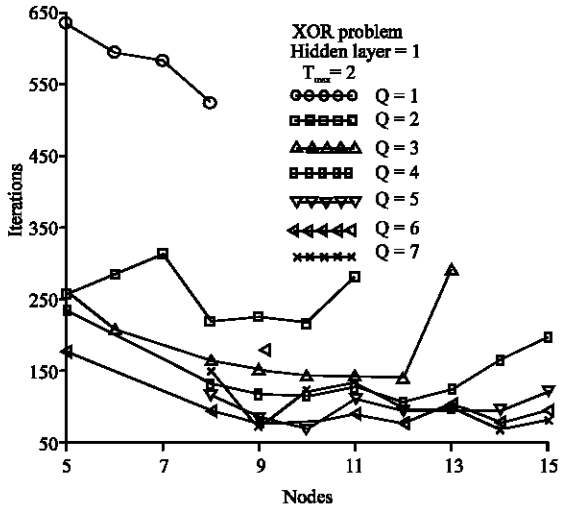

Fig. 3: Effect of initial value of inverse matrix (Q) for a constant T_{max} in EKF algorithm

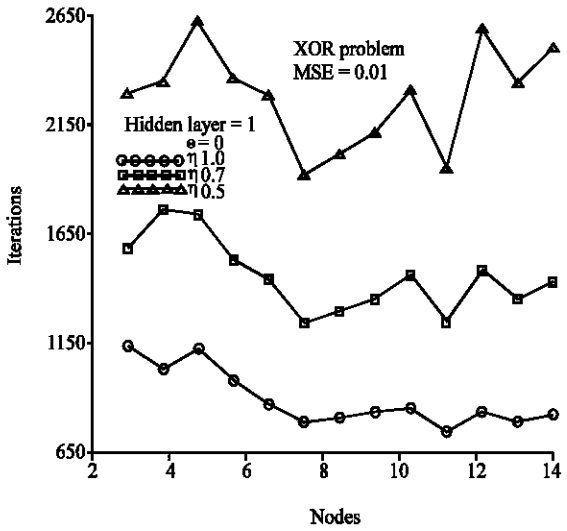

Fig. 4: Effect of learning rate (η) in SDM algorithm

Extended kalman filter (EKF): In this study, EKF algorithm is applied to multilayer perceptron. The EKF algorithm is a state estimation method for a non-linear system which can be used for parameter estimation. A multilayer perceptron is a non-linear system having a layered structure. Its learning method is considered as parameter estimation. This method gives approximately the minimum variance estimates of the link weights[13,14] which results in better convergence. The convergence speeds of EKF and SDM algorithms are compared by using XOR problem Fig. 3 shows the number of iterations required by EKF algorithm for various number of hidden nodes. In Fig. 4, the number of iterations required by SDM algorithm for different number of hidden nodes is shown. As a comparison of EKF and SDM algorithms, for 8 nodes in the hidden layer, it requires only 60 iterations for EKF algorithm (Fig. 3), whereas it requires 700 iterations for SDM algorithm for learning rate (η) to reach the mean squared error (MSE) of 0.01.

Weight updation of the ANN by using EKF algorithm: For each pattern, the output of the nodes in the successive layers is calculated by using Equations (1-4) in a feed-forward manner. The weights W_{ij} are updated from the output layer to the input layer through hidden layers by using the equations given in Table 1. In these equations, λ, δ, ι, α and β are updating parameters and Q is the inverse matrix.

Functional Update Method (FUM): In classification problems, input patterns can be grouped into classified subset and misclassified subset for any given weights[17,18]. The input patterns are said to be misclassified if the error `D' in the output layer is greater than 0.5 The input patterns are said to be classified if D is less than 0.5. Weights are modified only when D is greater than 0.5. The functional update algorithm used is as follows :

Step 1: Initialize the weights randomly.

Step 2: Present a pattern with new inputs and desired outputs.

Step 3: Compute network output by Eq. 2.

Step 4: Determine V^n, the set of valid update data in the output layer for the i^{th} output node by :

$$0.5 \leq D < 1 - \epsilon \qquad (5)$$

where

$$D = \left| d_i(p) - x_i^M(p) \right| \text{ and}$$

ϵ is the error fixed by the programmer

If V^n is empty, i.e. not even one node in the output layer does satisfy Equation (5), go to step 8. Otherwise go to step 5.

Step 5: Compute the objective function E (p) by :

$$E(p) = \frac{1}{2n} \Sigma \sum_{x_i^M \in V^n} [di(p) - x_i^M(p)]^2 \qquad (6)$$

Table 1 Algorithm for extended Kalman filter

For p = 1 to . . .

$$\hat{\lambda}(p)=\hat{\lambda}(p-1)+\mu(p)\left[\frac{(d_i(p)-x_i^M(p))^T(d_i(p)-x_i^M(p))}{N_M}-\hat{\lambda}(p-1)\right]$$

$$\hat{y}_i^{M-1}(p)=\hat{x}_i^M(p) \qquad 1\le i\le N_M-1$$

For $n = M-1$ to $1\ Nn_{+1}-1$ throughstep -1

For $i = 1$ to $N_{n+1}-1$

$$\delta_i(p)=xi^M(p)(1-x_i^M(p)) \quad \text{for the output layer}$$

$$\delta_i(p)=xi^M(p)(1-x_i^M(p))\sum_{k-1}\delta_k(p)_{W_{ij}}(p) \quad \text{for the Hidden layer}$$

initialize inverse matrix Q for the output layer and hidden layer

$$\Psi_i^n(p)=Q_i^n(P-1)\hat{X}_i^n(p); \quad \alpha_i^n(p)=(\hat{x}^n)^T\Psi_i^n(p)$$

$$\beta_i^n(p)=(\delta_i^n(p))^T\delta_i^n(p); \quad Zi(p)=\hat{\lambda}(p)+\alpha_i^n(p)\beta_i^n(p)$$

Updation of weights

Between the output layer and previous hidden layer

$$R_i(P)=d_i(p)-y_i^n(p); \text{ For other layers } R_i(p)=x_i^n(p)-y_i^n(p)$$

$$W_{ij}^n(p)=w_{ij}^n(p)-1+\frac{\delta_i^n(p)R_i(p)\Psi_i^n(p))}{Z_i(p)}$$

Updation of inverse matrix Q

Updation of $\hat{y}$ Only for hidden layers

$$\hat{y}_{i+1}^n(p)=\hat{y}_i^n(p)+\delta_i^n(p)\sum_{i-1}^{Nn}\hat{x}_i^n(p)(\hat{w}_{ij}^n(p)-\hat{w}_{ij}^n(p-1)); \quad \hat{y}_i^{n-1}=\hat{y}_{Nn+1}^n$$

Table 2 Equations for calculating the computational effort required SDM, EKF and FUEKF algorithms

Forward calculation

Linear

$$\sum_{i-1}^{L-1}(4n_i+5)n_{i+1}$$

Backward calculation

SDM

$$9n_L+7\sum_{i-L}^{2}n_i n_{i-1}+\sum_{i-L}^{3}(4n_i+5)n_{i-1}$$

EKF

$$15+5n_L+\sum_{i-L}^{2}(n^3{}_{i-1}+28n^2{}_{i-1}+18n_{i-1}+2)+\sum_{i-L}^{3}(4n_i+5)n_{i-1}$$

FUEKF

$$15+7n_L+\sum_{i-L}^{2}(n^3{}_{i-1}+28n^2{}_{i-1}+18n_{i-1}+2)+\sum_{i-L}^{3}(4n_i+5)n_{i-1}$$

In the above equations

L - the number of layers

n_i - number of nodes in the i^{th} layer

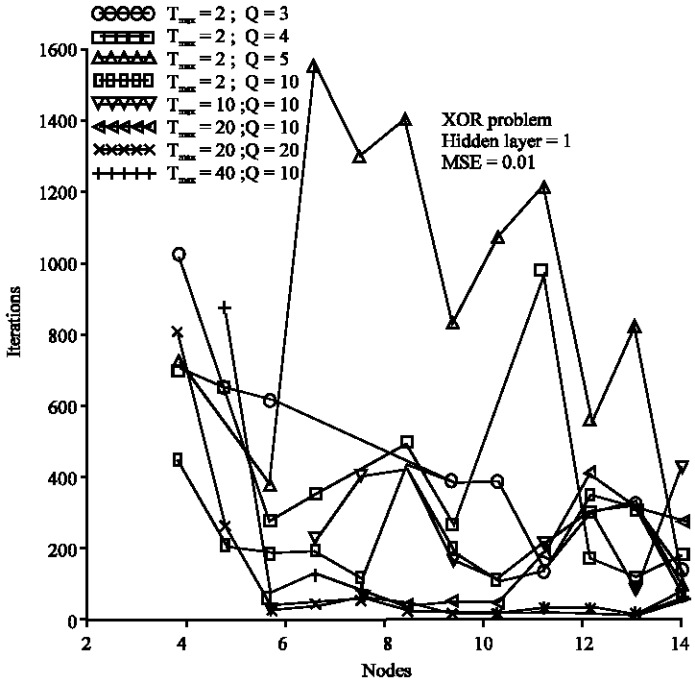

Fig. 5: Effect of initial value of inverse matrix (Q) for a constant T_{max} in EKF with FUM algorithm

Step 6: For EKF algorithm with FUM, adapt weights by using equations given in Table 1.

Step 7: Repeat by going to step 3.

Step 8: Change the sigmoid function of the output neuron to the signum function

The main advantage of FUM is that it will stop as soon as the misclassified set is empty. The number of iterations required by EKF algorithm with FUM for different nodes in the hidden layer is shown in Fig. 5. From this figure, it is clear that the number of iterations required by EKF with FUM for 8 nodes in the hidden layer to reach MSE of 0.01 is only 9, whereas the number of iterations required for EKF algorithm is 60 (Fig. 3). The equations for calculating computational effort required by the computer for SDM, EKF and EKF with FUM algorithms are given in Table 2. As a comparison of the algorithms, the number of arithmetic operations required for SDM, EKF and EKF with FUM algorithms to train a 6-6-1 network configuration is given in Table 3.

Table 3: Computational effort required by SDM, EKF and FUEKF algorithms

Forward computation	203
Backward Computation	
SDM	357
EKF	9412
FUEKF	9414

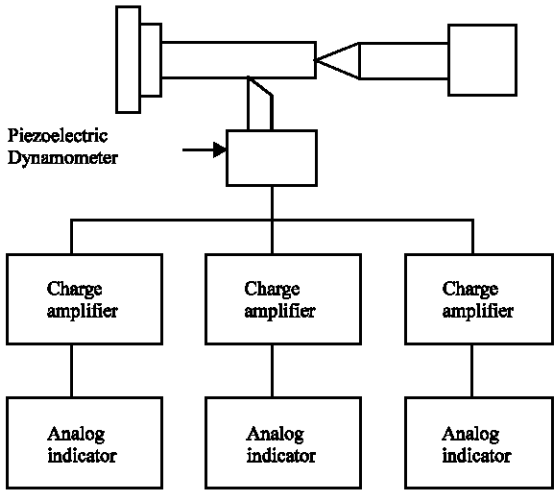

Fig. 6: Experimental setup

DESCRIPTION OF EXPERIMENTS

Experimental set-up: Schematic diagram of the experimental set-up is shown in Fig. 6. The experimental study was carried out on a precision high speed Lathe. The turning operation was carried out on a spheroidal graphite cast iron of 220-240 HB. The tool material used was widalon HK 15. The cutting Speed (S) is from 200- 500 m/min with depth of cut (D_c) from 0.5 - 2.0 mm and feed rate (F) 0.063-0.25 mm/rev. Data collected include static forces axial force (F_x), radial force (F_y), tangential force (F_z) and flank wear land width (Vb) for various combinations of S, F and D_c. The static forces were measured, using a three component piezoelectric crystal type dynamometer (Kistler type 9441) as a force

transducer. Charge amplifiers were used for signal amplification. Forces were read from the analog meter attached to the dynamometer through charge amplifiers. Tool flank wear land width (V_b) was measured using tool-maker's microscope. About 113 patterns were collected.

Experimental procedure: Experiments were conducted for various sets of cutting conditions - S, F, D_c. Static forces and V_b measurements were taken at different intervals of time. Depending on the length of cut, machining was stopped after every 60-80 seconds and Vb of the tool was measured. Static forces were recorded at 2 or 3 intermediate points between two wear measurements. The sets of measurements immediately prior to a wear measurement have been used for training the neural network. During reinsertion of tool inserts after every wear measurement, inserts were slugged into the slot made out in the tool holder, so that there is no change in the tool overhang.

Selection of data: Selection of patterns for training the neural network is important as they should be representative of all the patterns collected during machining. Therefore, statistical techniques have been used to select the patterns out of 113 patterns collected during the experiment. The number of classes selected are two. Patterns with maximum variance VE_i^2 are selected. The maximum VEi^2 of a pattern is calculated by:

$$VE_i^2 = \frac{\sum_{j-1}^{L} (x_{ij} - \bar{x}_j)^2}{\sigma_i^2} \qquad (7)$$

where

$$\sigma^2 i = \frac{1}{L} \sum_{i-1}^{L} (x_{ij} - \bar{x}_j)^2 \text{ and}$$

nf is the number of features.

Training the network: Training was done offline using PC-AT386 computer. The features - S, F, D, F_x, F_y, F_z are the inputs to the input layer and V_b is the output in the output layer.

Training and testing the network with SDM & EKF algorithms: The value of each variable for all the patterns is divided by its maximum value, so that the values lie between 0.0 and 1.0. Variable in the output should neither be 0.0 nor 1.0. The value 0.0 is considered as 0.00001 and 1.0 as 0.99999. This is, because the output of the sigmoid function will never reach 0.0 or 1.0. In this study, the actual number of classifications required is two. The range of classifications is given in Table 4.

Table 4: Range of output for each class

Output feature	Class I	Class II
V_b (μm)	<= 100	101 - 1300

For training the network with SDM algorithm, the conditions used are η= 1.0, W_{ij} = 0.2-0.4 and θ = 0.0 with network configuration as 6-6-1. The conditions used for training the network with EKF algorithm are $\mu(p)$ = 1/20, with initial value for the inverse matrix Q = 10, W_{ij} = 0.2 - 0.4 and θ = 0.0.

Training extended Kalman filter with functional update method (FUEKF): The patterns are binary coded. Single-layer perceptron is used. The network configuration used is 31-4 (31 nodes in the input layer and 4 nodes in the output layer). The conditions used for FUEKF algorithm are μ (p) = 1/20 and initial value for the inverse matrix Q = 10, Wij = 0.2 - 0.4 and θ = 0.2 - 0.4.

RESULTS

During turning operation, 113 patterns were collected. Out of 113 patterns, 30 patterns were selected for training the neural network. Thirty patterns used for training are given in Table 5. The algorithms used for weight updates are Steepest-Descent Method (SDM),

Table 5: Patterns used for training the ANN

	Inputs						Output
	S	F	D	F_x	F_y	F_z	V_b
Class I	450	0.10	2.0	180	130	450	15
	450	0.10	1.5	150	115	350	15
	450	0.10	1.0	115	105	250	20
	500	0.25	0.5	60	115	215	20
	500	0.20	0.5	60	110	195	20
	400	0.25	0.5	80	125	230	20
	450	0.10	2.0	100	140	450	30
	450	0.10	1.5	130	115	345	30
	450	0.10	1.0	100	105	250	40
	450	0.10	2.0	160	140	470	55
	450	0.10	1.5	150	105	330	65
	450	0.10	1.0	115	110	260	75
	450	0.10	2.0	110	140	470	80
	450	0.10	1.5	135	110	325	94
	450	0.10	0.5	75	150	150	100
Class II	450	0.10	1.5	680	580	560	375
	450	0.10	1.5	750	650	500	400
	450	0.10	2.0	850	700	750	400
	450	0.10	1.5	240	850	620	540
	450	0.10	1.0	550	590	430	550
	450	0.10	2.0	1100	1200	840	585
	450	0.10	1.5	260	1200	800	690
	450	0.10	1.0	570	700	450	755
	450	0.10	2.0	1200	1400	1000	785
	450	0.10	1.5	320	1800	1400	825
	450	0.10	2.0	1500	1800	1000	880
	450	0.10	1.0	950	700	500	980
	450	0.10	1.5	380	1880	1500	980
	450	0.10	2.0	1250	1440	1040	990
	450	0.10	1.0	900	840	500	1300

S – m/min, F – mm/rev. D_c – mm, F_x, F_y and F_z–N, V_b- μm

Table 6: Number of iterations required by SDM, EKF and FUEKF algorithms

Algorithm Used	Range of threshold	Mean squared error	Number of iterations	Configuration of the network
SDM	-	0.065	250	6-6-1
EKF	-	0.065	160	6-6-1
FUEKF	0.2-0.4	0.500	70	31-1

Table 7 Result of the network for 84 test patterns

		Algorithm		
		SDM	EK	FUEKF
Class	Total No. of test patterns used	Number of patterns matching. Percentage is given in brackets.		
I	42	42 (100%)	42 (100%)	42 (100%)
II	42	23 (54%)	25 (59%)	21 (50%)

Table 8 Overall performance of the network

	SDM	EKF	FUEKF
Performance in percentage	77.30	79.76	75.00

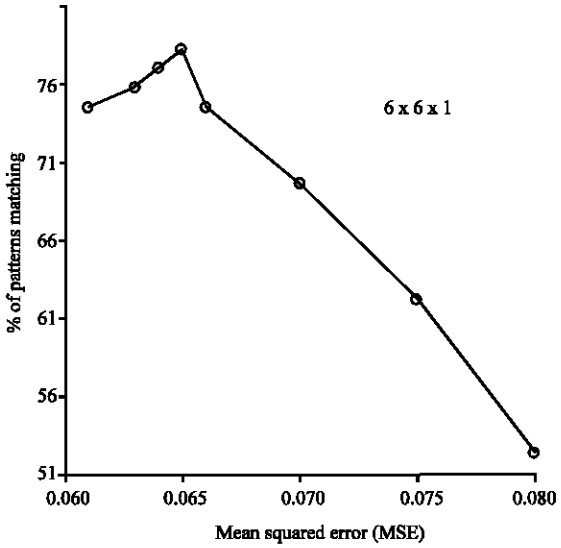

Fig. 7: Classification performance of SDM algorithm

Extended Kalman Filter (EKF) and EKF with functional update method (FUEKF). Training was done offline. The convergence speed of EKF algorithm is faster than the convergence speed of SDM algorithm. The training was stopped once the error 'E' reached 0.065. It is found that the performance of the network will deteriorate when E is too high or too low. In Fig. 7, performance of the network for various E, trained by using SDM algorithm is shown.

In Table 6, the number of iterations required for training the network with SDM, EKF and FUEKF algorithms is given. When the network was trained with functional update, the training was stopped once the error E reached 0.5. Since the signum function is used to update the weights in FUM, it is sufficient to train the network till the error comes to 0.5. In Table 7, performance of SDM, EKF and FUEKF algorithms for 84 test patterns is given. The performance of the network for Class I is 100% with all the algorithms, but for Class II the performance is less than 60%. This is due to the large discontinuity of V_b in Class II (Table 5). The overall performance of the network is given in Table 8.

CONCLUSION

In this work, a supervised neural network has been trained with three update methods: Steepest-Descent Method (SDM), Extended Kalman Filter (EKF) and extended Kalman filter with functional update (FUEKF). The configuration of the network for SDM and EKF algorithms is 6-6-1. The data used is normalized analog data collected during turning operation. The configuration of the network used for FUEKF algorithm is 31-4 without a hidden layer; the data used is binary coded data of the analog signals. The main importance is given to EKF and FUEKF algorithms. The iterations required for EKF algorithm is less than the iterations required for SDM algorithm, to reach the Mean Squared Error (MSE) of 0.065 (Table 6), whereas the computational effort required for EKF algorithm is much higher than the computational effort required for SDM algorithm (Table 3).

The performance classification of EKF algorithm is almost the same as that of the performance classification of SDM algorithm (Table 8). When the network was trained with FUEKF algorithm, the MSE was fixed at 0.5 and hence the iterations required was very less. The performance of the FUEDF algorithm is almost the same as the performance of the EKF and SDM algorithms (Table 8). Except for the computational time required by EKF and FUEKF algorithms, the EKF and FUEKF algorithms are superior to SDM algorithm in terms of convergence rate and classification of performance.

REFERENCES

1. Kegg, R.L., 1984. On-line machine and process diagnostics, Annals of CIRP, 32: 469-473.
2. Lee, D., I. Hwang and D. Dornfeld, 2004. Precision Manufacturing Process Monitoring With Acoustic. Emission, Proceedings AC'04, Zakopane, Poland.
3. Xiaoli, Li and Xin Yao, 2005. Multi-Scale Statistical Process Monitoring in Machining, IEEE Trans On Industrial Electronics, Vl.52, N.3, pp: 924-926.
4. Tonshoff, H.K., J.P. Wulfsberg, H.J.J. Kals and W. Konig, 1988. Developments and trends in monitoring and control of machining processes, Annals of CIRP, 37: 611 - 621.
5. LiDan and J. Mathew, 1990. Tool wear and failure monitoring techniques for turning a review, Int. J. Mach. Tools and Manufact, 30: 579-597.
6. Chryssolouris, G., M. Domrose and P. Beaulieu, 1992. Sensor synthesis for control of manufacturing processes, ASME J. Engg. for Ind., 114: 158-174.
7. Fortuna, L., S. Graziani, M. LoPresti and G. Muscato, 1992. Improving back-propagation learning using auxilliary neural networks, Int. J. Cont., 55: 793-807.
8. Lippmann, R.P., 1987. An introduction to computing with neural nets, I IEEE Trans. On Acoustics, Speech and Signal Processing Magazine, (V35, N4, 1987), pp: 4-22.
9. Yao, Y.L. and X.D. Fang, 1993. Assessment of chip forming paterns with tool wear progression in machining via neural networks, Int. J. Mach. Tools and Manufact., 33: 89-102.
10. Hush, D.R. and B.G. Horne, 1993. Progress in supervised neural networks, IEEE Signal Processing Magazine, pp: 8-38.
11. Bernard Widrow, 1990. 30 Years of adaptive neural networks: Perceptron, madaline and back-propagation, Proceedings of the IEEE , 18: 1415-1442.
12. Hirose, Y., K.Y. Yamashita and S. Hijiya, 1991. Back-propagation algorithm which varies the number of hidden units, Neural Networks, 4: 61-66.
13. Purushothaman, S. and Y.G. Srinivasa, 1994. A back-propagation algorithm applied to tool wear monitoring, Int. J. Mach. Tools and Manufact, 34: 625-631.
14. Purushothaman, S. and Y.G. Srinivasa, Tool wear monitoring using artificial neural networks, Procee. of Xvth AIMTDR Conf. Coimbatore, India, pp: 594-599.
15. Liguni, Y. *et al.*, 1992. A real time learning for a multilayered neural network based on the extended Kalman filter, IEEE Transactions on Signal Processing, 40: 959-966.
16. Purushothaman, S. and Y.G. Srinivasa, 1994. An adaptive algorithm for weight update in back-propagation and its application to pattern recognition in turning, Int. Conf. On Adv. Mfg. Tech. Johor Bahru, Malaysia, 30: 485-495.
17. Shih-Chi Huang and Yih-Fang Haung, 1990. Learning algorithms for perceptrons using back-propagation with selective updates, IEEE Control Systems Magazine, pp: 56-61.
18. Purushothaman, S. and Y.G. Srinivasa, 1993. Application of a modified back-propagation algorithm for pattern classification in turning, procee. of the 1993 ASPE conf. Seattle, Washington, U.S.A., pp: 250-253.