



# An effective LS-SVM-based approach for surface roughness prediction in machined surfaces



Nian Zhang<sup>a,\*</sup>, Devdas Shetty<sup>b</sup>

<sup>a</sup> Department of Electrical and Computer Engineering, University of the District of Columbia, 4200 Connecticut Avenue, NW, Washington, DC 20008, USA

<sup>b</sup> School of Engineering and Applied Sciences, University of the District of Columbia, 4200 Connecticut Avenue, NW, Washington, DC 20008, USA

## ARTICLE INFO

### Article history:

Received 2 April 2015

Received in revised form

16 July 2015

Accepted 31 August 2015

Available online 17 March 2016

### Keywords:

Surface roughness prediction

Least squares support vector machine (LS-SVM)

Neural networks

Levenberg–Marquardt algorithm

ANOVA

Machined surfaces

## ABSTRACT

An effective least squares support vector machine (LS-SVM)-based approach was developed to predict the surface roughness in machined surface. The real AISI4340 steel and AISI2 steel data set was used to conduct the experiments. The analysis of variance (ANOVA) was used to validate the assumption of normal distribution, as well as the independent distribution of the errors. For the neural networks model, with 70%, 15%, and 15% of data as training, validation, and testing data, respectively, the best validation error is 0.0097343. The training error is  $9.08888 \times 10^{-4}$  and the testing error is  $1.09510 \times 10^{-1}$  accordingly. NN methods also discovered the correlation between the predicted surface roughness (Ra) and the actual surface roughness in the form of predicted  $Ra \cong 0.41 \times \text{Actual } Ra + 0.2$ . The LS-SVM performance was also compared to the analysis of variance (ANOVA) method, and neural networks model trained by Levenberg–Marquardt algorithm. The experimental results showed that the proposed LS-SVM algorithm produced a determination coefficient of  $=0.9439$ , which is higher than the ANOVA and NN results of 0.1917 and 0.7266.

© 2016 Elsevier B.V. All rights reserved.

## 1. Introduction

Surface roughness is a critical quality index which determines the quality of machined surfaces and is influenced by the cutting parameters [1]. Surface roughness prediction has very important applications in manufacturing industries, environmental sciences, and military applications [2].

Many techniques have been developed to predict surface roughness. Sonar et al. [3] used radial basis neural networks for prediction of surface roughness in turning of mild steel and concluded that radial basis neural networks model are slightly inferior when compared to multilayer perceptron model. Naidu et al. [4] presented that the neural networks models are superior to regression models. Caydas and Ekici [5] developed three different types of support vector machines (SVMs) tools including least squares SVM (LS-SVM), Spider SVM and SVM-KM models to estimate the surface roughness values. The prediction results showed that all the SVMs performed better than neural networks. However, the training data are relatively intact. Wang et al. [6] applied LS-SVM algorithms for prediction model of surface roughness for grinding machining operation. Result shows that LS-SVM

outperformed the RBF neural networks method. However, the LS-SVM needs to be compared with more efficient neural networks.

Given this context, it is imperative to develop an effective LS-SVM and compare its performance with the neural networks and analysis of variance methods for surface roughness prediction.

This paper is organized as follows. In Section 2, problem statement is described. In Section 3, the principle of least squares support vector machines (LS-SVM) and Neural Networks with Levenberg–Marquardt optimization are illustrated in detail. In Section 4, experimental results and discussions are demonstrated. In Section 5, the comparison of ANOVA, NN, and LS-SVM on the surface roughness prediction was presented. In Section 6, the conclusions are given.

## 2. Problem statement

Given the sample of  $N$  points  $\{x_i, y_i\}_{i=1}^N$ , with input vectors  $x_i \in \mathbb{R}^p$  and output values  $y_i \in \mathbb{R}$ , the goal is to estimate a model of the following form:

$$y_i = w^T \phi(x_i) + b + \varepsilon_i \quad (i = 1, 2, \dots, l) \quad (1)$$

where  $\phi(\cdot): \mathbb{R}^p \rightarrow \mathbb{R}^{n_h}$  is the mapping to a high dimensional (and possibly infinite dimensional) feature space, and the residuals are assumed to be independent and identically distributed with zero mean and constant and finite variance.

\* Corresponding author.

E-mail addresses: [nzhang@udc.edu](mailto:nzhang@udc.edu) (N. Zhang), [devdas.shetty@udc.edu](mailto:devdas.shetty@udc.edu) (D. Shetty).

### 3. Proposed methods

#### 3.1. Principle of least squares support vector machine regression

Least squares support vector machine (LS-SVM) formulates a regularized cost function and changes its inequation restriction to equation restriction. As a result, the solution process becomes a solution of a group of equations which greatly accelerates the solution speed [7]. To solve the problem stated in (1), the following optimization problem with a regularized cost function is formulated:

$$\min_{w,b,\varepsilon_i} \left( \frac{1}{2} w^T w + \frac{C}{2} \sum_{i=1}^l \varepsilon_i^2 \right) \quad (2)$$

The solution of LS-SVM regressor will be obtained after we construct the Lagrangian function. The extreme point of  $Q$  is a saddle point. Differentiating  $Q$  and using Lagrange multipliers, one obtains the following optimality conditions:

$$\frac{\partial Q}{\partial w} = w - \sum_{i=1}^l \alpha_i \phi(x_i) = 0 \quad (3)$$

$$\frac{\partial Q}{\partial b} = - \sum_{i=1}^l \alpha_i = 0 \quad (4)$$

$$\frac{\partial Q}{\partial \alpha} = w^T - \phi(x_i) + b + \varepsilon_i - y_i = 0 \quad (5)$$

$$\frac{\partial Q}{\partial \varepsilon_i} = C \varepsilon_i - \alpha_i = 0 \quad (6)$$

where  $\alpha \in \mathbb{R}$  are the Lagrange multipliers. From formulas above, we can obtain:

$$\frac{1}{2} \sum_{i=1}^l \alpha_i \phi(x_i) \sum_{j=1}^l \alpha_j \phi(x_j) + \frac{1}{2C} \sum_{i=1}^l \alpha_i^2 + b \sum_{i=1}^l \alpha_i = \sum_{i=1}^l \alpha_i y_i \quad (7)$$

The formula above can be expressed in matrix form:

$$\begin{bmatrix} 0 & e^T \\ e & \Omega + C^{-1}I \end{bmatrix} (l+1)(l+1) \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ Y \end{bmatrix} \quad (8)$$

In this equation

$$e = [1, \dots, 1]_x^T$$

$$\Omega_{ij} = K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \quad (9)$$

Formula (7) is a linear equation set corresponding to the optimization problem and can provide us with  $\alpha$  and  $b$ . Thus, the prediction output decision function is

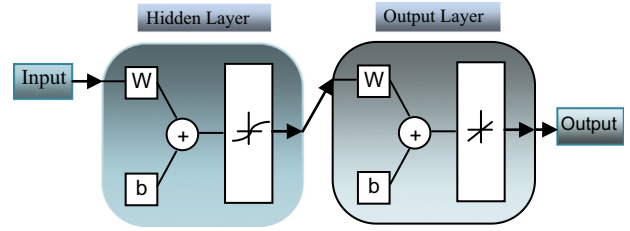
$$y(x) = \sum_{i=1}^l \alpha_i K(x_i, x) + b \quad (10)$$

where  $K(x_i, x)$  is the core function.

#### 3.2. Fundamentals of neural networks with Levenberg–Marquardt optimization

The proposed neural network model is a two-layer feedforward network, with a sigmoid transfer function in the hidden layer and a linear transfer function in the output layer, as shown in Fig. 1.  $W$  is the weight matrix, and  $b$  is the bias.

In this section, the above neural network is trained using Levenberg–Marquardt backpropagation algorithm. It is a network training function that updates weight and bias values according to Levenberg–Marquardt optimization. It is often the fastest backpropagation algorithm for training moderate-sized feedforward neural networks (up to several hundred weights), although it does require more memory than other algorithms, such as conjugate gradient backpropagation.



**Fig. 1.** Neural networks architecture for the predictive model. The network is a two-layer feedforward network, with a sigmoid transfer function in the hidden layer and a linear transfer function in the output layer. This network also uses tapped delay lines to store previous values of  $y(t)$  sequences.  $W$  is the weight matrix, and  $b$  is the bias.

Like the quasi-Newton methods, the Levenberg–Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feedforward networks), then the Hessian matrix can be approximated as

$$H = J^T J \quad (11)$$

and the gradient can be computed as

$$g = J^T e \quad (12)$$

where  $J$  is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and  $e$  is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix.

The Levenberg–Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update [8]:

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \quad (13)$$

When the scalar  $\mu$  is zero, this is just Newton's method, using the approximate Hessian matrix. When  $\mu$  is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift toward Newton's method as quickly as possible. Thus,  $\mu$  is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function is always reduced at each iteration of the algorithm.

### 4. Experimental results and discussion

For experimental study, AISI4340 steel and AISID2 steel are used as the workpiece materials with average surface hardness values of 50, 55, and 60 HRC, respectively [9]. The cutting parameters include cutting speed (m/min), feed rates (mm/rev), and depths of cut (mm). The data set has a total of 34 combinations of the cutting parameters with the actual surface roughness.

First, we performed a normal probability plot on the residuals to identify substantive departures from normality, as shown in Fig. 2. The normal probability plot of the residuals appears as a straight line, which indicates that the assumption of normal distribution is valid [10].

The plot of the residuals versus the fitted values is illustrated in Fig. 3. Given that the dots are evenly distributed around the abscissa without a clear trend, the errors are independently distributed and the variance is constant [11].

Download English Version:

<https://daneshyari.com/en/article/411396>

Download Persian Version:

<https://daneshyari.com/article/411396>

[Daneshyari.com](https://daneshyari.com)