# Chapter 3

# Artificial neural networks and support vector machines

## 3.1 Introduction

Neural networks have found extensive use in many industrial applications for pattern recognition. In this thesis the aim is to apply the ANNs and SVMs in pattern recognition as a predictive tool. The idea is to train ANNs and SVMs to predict the ensemble average of a large number of rotation synchronised gear vibration signals using only a portion of the total number of signals. This is in essence non-linear mapping between the input and output space, a task that ANNs and SVMs have handled with success (Fidêncio et al, 2002; Gunn, 1988). The averaging of the rotation synchronised vibration signals is called time domain averaging as discussed in Chapter 2. It was also demonstrated that calculating the TDA by direct averaging can filter out broadband noise over the entire spectrum of the signal leaving only the vibration content of interest. The ANNs and SVMs mapping should therefore retain the non-linear filtering achieved in the frequency domain by the TDA calculated by direct averaging. Before a synchronous time domain averaging model can be developed for gear vibration using ANNs and SVMs, it is essential to have thorough understanding of their underlying mathematics. In this chapter, the theory of Multi-layer Perceptron (MLP) networks, Radial Basis Function (RBF) networks (Bishop, 1995) and Support Vector Machines (SVM) (Vapnik, 1995; Vapnik et al., 1997; Gunn, 1998) in the light of this work is presented. This chapter also presents simulation results based on a preliminary study conducted on a data set from an accelerated gear life test rig. The preliminary study is used to investigate the suitability of these methods for application in the development of a time-domain averaging model for gear vibration.

## 3.2 Artificial neural networks

In this work neural networks are viewed as parameterised non-linear mapping of input data to the output data. Learning algorithms are viewed as methods for finding parameter values that look probable in the light of the data. The learning process occurs by training the network through supervised learning. Supervised learning is the case where the input data set (*X*) and the output data set (*Y*) are both known and neural networks are used to approximate the functional mapping between the two data sets. In this section the theory and application of MLP and RBF networks formulations are presented. Simulations to assess the suitability of each of these formulations for use in the development of a synchronous time domain averaging model using a data set from the accelerated gear life test rig (Stander and Heyns, 2002[b]) are presented.

### 3.2.1 Multi-layer perceptron

The MLP provides a distributed representation with respect to the input space due to the cross-coupling between hidden units. In this study, the MLP architecture contains a hyperbolic tangent basis function in the hidden units and linear basis functions in the output units (Bishop, 1995). A schematic illustration of a 2-layer MLP network is shown in Figure 3.1.

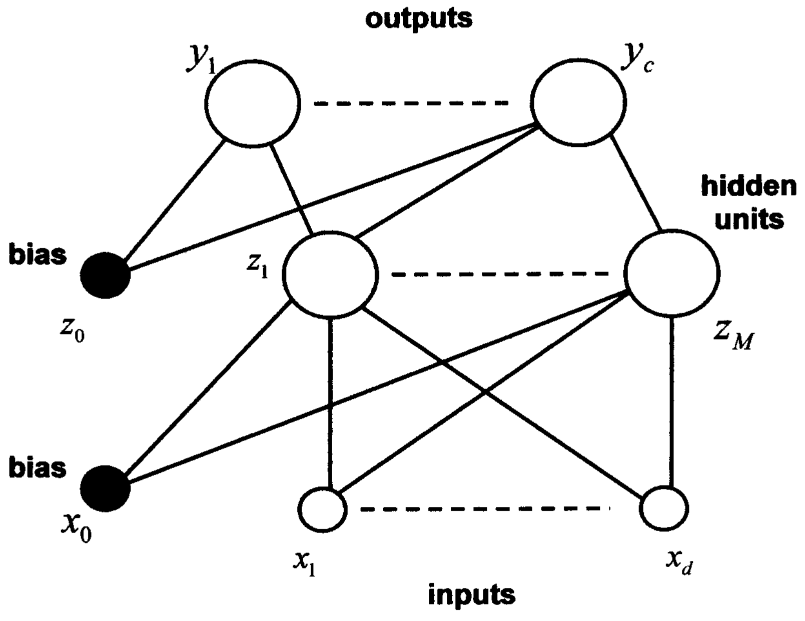

**Figure 3.1** Feed-forward network with two layers of adaptive weights (Bishop, 1995).

The MLP network architecture in Figure 3.1 contains hidden units, output units, and one hidden layer. The bias parameters in the first layer are shown as weights from an extra input having a fixed value of $x_0 = 1$. The bias parameters in the second layer are shown as weights from an extra hidden unit, with the activation fixed at $z_0 = 1$. The model in

Figure 3.1 is able to take into account the intrinsic dimensionality of the data. Models of this form can approximate any continuous function to arbitrary accuracy if the number of hidden units $M$ is sufficiently large. The size of a MLP network can be expanded by considering several layers but this is not necessary because it has been demonstrated through the Universal Approximation Theorem (Haykin, 1999) that a two-layered architecture is adequate for the multi-layer perceptron. As a result of this theorem, in this study a two-layered network shown in Figure 3.1 is chosen.

The output of the MLP network represented in Figure 3.1 is given by Equation 3.1.

$$y_k = f_{outer}\left(\sum_{j=0}^{M} w_{kj}^{(2)} f_{inner}\left(\sum_{i=0}^{d} w_{kj}^{(1)} x_i\right) + w_{k0}^{(2)}\right) \tag{3.1}$$

where $f_{outer}$ and $f_{inner}$ are activation functions, $w_{ji}^{(1)}$ denotes a weight in the first layer, going from input $i$ to hidden unit $j$, $w_{k0}^{(2)}$ is the bias for the hidden unit $k$ and $w_{kj}^{(2)}$ denotes a weight in the second layer. In this work $f_{inner}$ is a hyperbolic tangent function "tanh" and $f_{outer}$ is linear. The linear activation function is defined by Equation (3.2) and it maps the interval (-∞,∞) onto the interval (-∞,∞).

$$f_{outer}(\nu) = \nu \tag{3.2}$$

The hyperbolic tangent function is defined by

$$f_{inner}(\nu) = \tanh(\nu) = \frac{e^{\nu} - e^{-\nu}}{e^{\nu} + e^{-\nu}} \tag{3.3}$$

and it maps the interval (-∞,∞) onto the interval (-1,1). Figure 3.2 below shows the two activation functions used in this study.

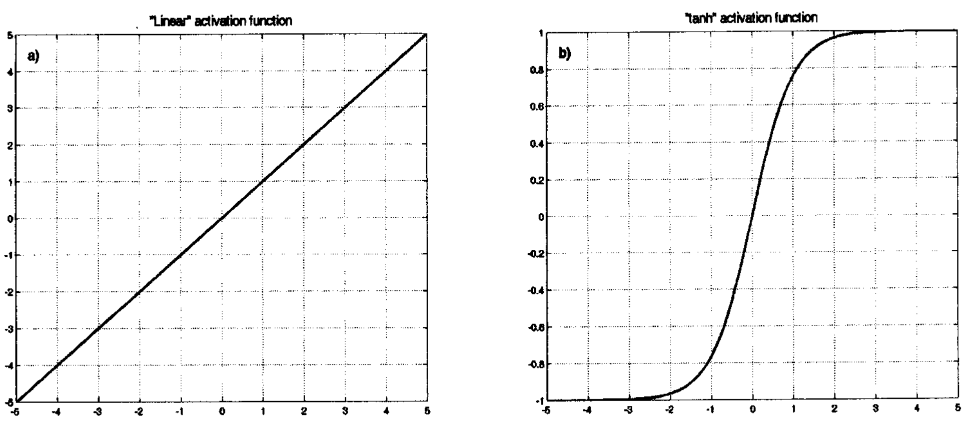

**Figure 3.2** (a) Plot of the linear activation function given by equation (3.2). (b) Plot of the 'tanh' activation function given by Equation (3.3).

Training the neural network is achieved by calculating the weights in Equations (3.1). There are two principal approaches that can be used to train neural networks. These methods are the maximum-likelihood approach and the Bayesian approach (Bishop, 1995). In maximum-likelihood training, optimisation methods are used to identify a set of parameter values that maximises the ability of a network to predict the output whenever presented with the input data. The Bayesian method uses Bayes's theorem (Bishop, 1995) to identify the probability distribution of weights in the light of the training data that are initially set to some prior distribution. The maximum-likelihood method may be viewed as a special case of the Bayesian method. In this work the maximum-likelihood method is implemented for computational efficiency.

### 3.2.2 Maximum-likelihood-based cost function

In the maximum-likelihood approach an optimisation procedure is used to identify the weights and biases of the neural networks in Equation (3.1). A cost function is chosen in order to use the optimisation technique. A cost function is a mathematical representation of the overall objective of the problem (Marwala, 2001). In this work, the overall objective is to identify a set of neural network weights, that can map the rotation synchronised gear vibration signal (input) to the ensemble average of the rotation synchronised gear vibration signals (TDA obtained from the $k$ signals). The mapping should use less rotation synchronised gear vibration signals than would otherwise be required to calculate the TDA using direct averaging. In other words, predicting the TDA of a rotating gear using only a fraction of the number of rotation synchronised gear vibration signals that would be used in the direct averaging approach.

If the training set $D=\{X_k,t_k\}_{k=1}^{N}$ is used and assuming that the targets $t_k$ are sampled independently given the inputs $X_k$ and the weight parameters, $w_{kj}$, the cost function, $E$, may be written as

$$E=\sum_{n}\sum_{k}\{t_{nk}-y_{nk}\}^2+\frac{\alpha}{2}\sum_{j}^{w}w^2 \tag{3.4}$$

where, $n$ is the index for the training pattern and $k$ is the index for the output units. The first term in Equation (3.4) is the sum-of-square-of-errors cost function, which tends to give similar absolute errors for each pattern. This results in poor performance on target values of small magnitude. The other cost function that has been used is the cross-entropy cost function (Hopfield, 1987; Hinton, 1987). Minimisation of the cross-entropy cost function tends to give the same relative errors for small and large targets. The cross-entropy cost function plus the weight decay regularisation parameters may be written as follows:

$$E=-\sum_{n}\sum_{k}\{t_{nk}\ln(y_{nk})+(1-t_{nk})\ln(1-y_{nk})\}+\frac{\alpha}{2}\sum_{j}^{w}w_j^2 \tag{3.5}$$

The cost function in Equation (3.5) has been found to be suited for classification problems while the one in Equation (3.4) has been found to be suited for regression problems (Bishop, 1995). Since the application in this work is a type of regression Equation (3.4) is used.

### 3.2.3 Regularisation

The second term in Equation (3.4) is the regularisation parameter. The regularisation parameter in Equation (3.4) penalises large weights and ensures that the mapping function is smooth (Vapnik, 1995). This regularisation parameter is called the weight decay and its coefficient, α, determines the relative contribution of the regularisation term on the training error. The inclusion of the regularisation parameter has been found to give significant improvements in network generalisation (Hinton, 1987).

In neural networks, to produce an over-fitted mapping with regions of large curvature requires large weights. The weight decay regularisation penalises large weights thereby encouraging the weights to be small and avoiding an over-fitted mapping between the inputs and the outputs. If α is too high then the regularisation parameter over-smoothes the network weights and as result giving inaccurate results. If α is too small then the effect of the regularisation parameter is negligible and unless other measures that control the complexity of the model, such as the early stopping method (Bishop 1995) are implemented, the trained network becomes too complex and thus performs poorly on validation sets.

### 3.2.4 MLP network training

Before minimisation of the cost function is performed, the network architecture needs to be constructed by choosing the number of hidden units, *M*. If *M* is too small, the neural network will be insufficiently flexible and will give poor generalisation of the data because of high bias. However, if *M* is too large, the neural network will be unnecessarily flexible and will give poor generalisation due to a phenomenon known as overfitting caused by high variance (Geman et al.,1992). The weights ($w_i$) and biases in the hidden layers are varied using optimisation methods until the cost function is minimised. Gradient descent methods are implemented and the gradient of the cost function is calculated using the back-propagation method (Bishop, 1995). The details of the back-propagation method are found in Appendix B. In this work it was decided to use the Scaled Conjugate Gradient (SCG) method over Conjugate Gradient (CG). This choice was made because SCG method is more computational efficient than CG while retaining the essential advantages of the CG method (Haykin, 1999; Marwala, 2001). The details of these optimisation methods are explained in Appendix C.

### 3.2.5 MLP simulation results from a preliminary investigation using data from the accelerated gear life test rig

This section seeks to validate the suitability of the MLP network for use in synchronous TDA model using gear vibration data from the accelerated gear life test rig. This was done in the following steps.

- Data pre-processing. The acceleration signal measured from the gearbox casing was synchronised with the rotation of the gear using the one pulse per revolution shaft encoder signal. This resulted in 160 rotation synchronised gear vibration signals. Each signal contained 8192 sample points. The rotation synchronised gear vibration signals were resampled to 1024 points per signal to reduce computational load. This results in an input space $X_k$ of dimension $(160\times1024)$. The target $t_k$ is the TDA of the gear vibration calculated using the direct averaging approach for 160 gear rotations. The dimensions of the target are $(1\times1024)$.

- Selecting type of network. A two-layer network was selected because the Universal approximation Theorem (Haykin, 1999) states that a two-layered network is sufficient for mapping data of arbitrary complexity (Marwala, 2001).
- Selecting number of hidden units. The number of hidden units was chosen between 3 and 15 and the one that resulted in the least square errors when simulating with unseen validation sets was selected. In this application 10 hidden units were selected because they resulted in a small error without severe computational penalties.
- Selected activation functions. The hyperbolic tangent function was selected as the inner activation function and a linear function was selected as the outer activation function.
- Type of optimisation technique used. The scaled conjugate gradient optimisation technique was used to optimise the cost function because of its computational efficiency.
- Selecting number of inputs. The number of inputs was chosen between 1 and 50 and the one that resulted in the least square errors when simulating with unseen validation sets was selected.
- The regularisation coefficient, $\alpha$ , was selected by trial and error, starting at a value of $\alpha = 0$, and increasing $\alpha$ sequentially in steps $\alpha = 0.1$ until satisfactory smoothness of the predicted result was obtained from simulations with unseen data. In this work $\alpha = 1.5$ was found to be most suitable.

The following plots show the performance of MLP neural network on a preliminary study conducted using vibration data from the accelerated gear life test rig (Stander and Heyns, 2002). Figure 3.3 shows MLP simulation results for a network of 10 hidden units and 40 input vectors (40 rotation synchronised vibration signals), each signal with signal has 1024 points resulting in an input space of (40 □1024). From this simulation it is clearly evident that this MLP network architecture is suitable for use in a synchronous TDA model in that it correctly predicts the target (time domain average after 160 gear rotations with only 40 gear rotation signals).

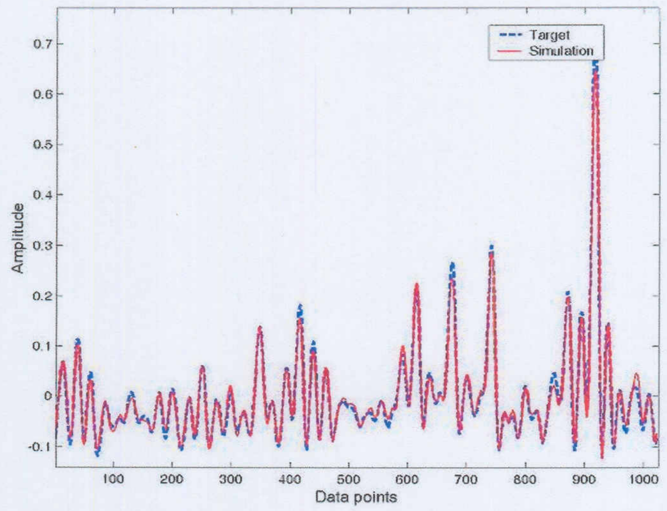

**Figure 3.3** MLP simulation results for network with 40 inputs and 10 hidden units superimposed on TDA calculated by direct averaging.

Figure 3.4 shows the simulation results of validation sets as a function of the number of inputs. It is observed that the RMS error stabilises after 40 input vectors, therefore, 40 inputs are selected as the optimum number of inputs, therefore the amount of data that is required to calculate the TDA is reduces by 75 percent.

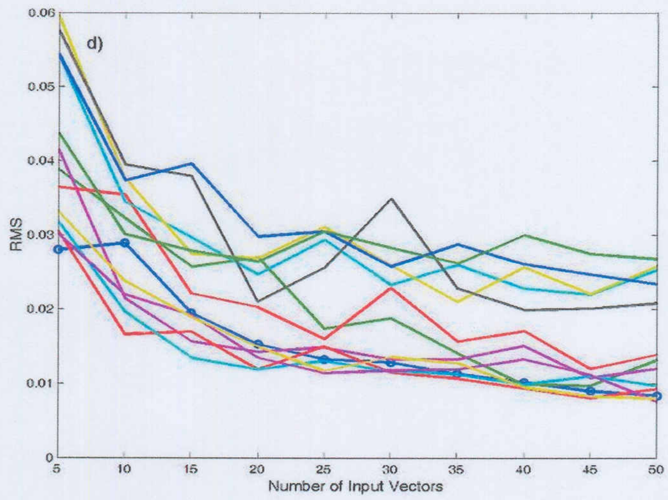

**Figure 3.4** RMS of MLP simulation results for network with 10 hidden units vs. Number of input signals.

### 3.2.6 Radial basis functions

The radial basis function has its origin in techniques for performing exact interpolation of a set of data points in a multi dimensional space (Powell, 1987). The exact interpolation problem requires every input vector to be mapped exactly to the corresponding vector and will be used in the discussion of Radial Basis Function (RBF) network (Bishop, 1995). The RBF neural network can be obtained by introducing a couple of modifications to the exact interpolation process (Broomhead, 1988; Moody and Darken, 1989). The RBF neural networks provide a smooth interpolating function for which the number of basis functions is determined by the complexity of the mapping to be represented rather than the data set as in exact interpolation. In this work, the objective is to identify a set of basis functions, that can map the rotation synchronised gear vibration signal (input) to the ensemble average of the rotation synchronised gear vibration signals (TDA obtained from the *k* signals). The mapping should use less rotation synchronised gear vibration signals than would otherwise be required to calculate the TDA using direct averaging.

The RBF neural network mapping is given by given

$$y_k(\mathbf{x}) = \sum_{j=1}^{M} \omega_{kj}\phi_j(\mathbf{x}) + \omega_{k0} \tag{3.6}$$

where $\omega_{k0}$ are the biases, $\omega_{kj}$ are the basis function weights, $\mathbf{x}$ is the d-dimensional input vector and $\phi_j(\cdot)$ is the j$^{th}$ basis function. Several forms of basis functions have been considered, the most common being the Gaussian given by

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \tag{3.7}$$

where $\mathbf{x}$ is the d-dimensional input vector with elements $x_i$ , and $\boldsymbol{\mu}_j$ is the vector determining the centre of the basis function $\phi_j$ and has elements $\mu_{ji}$. For further detail on the selection of the basis function centres $\mu_{ji}$ see Bishop (1995). The parameter $\sigma$ controls the smoothness properties of the basis function. The Gaussian interpolation function is a localised basis function with the property that $\phi \to 0$ as $|x| \to \infty$. Another basis function that shares the properties of the Gaussian is given by

$$\phi_j(\mathbf{x}) = \left(\mathbf{x}^2 + \sigma^2\right)^{-\alpha} \tag{3.8}$$

It is, however, not necessary for this function to be localised. Other possible choices are the thin-plate spline function given by

$$\phi_j(\mathbf{x}) = \mathbf{x}^2 \ln(\mathbf{x}), \tag{3.9}$$

the function

$$\phi_j(\mathbf{x}) = \mathbf{x}^4 \ln(\mathbf{x}), \tag{3.10}$$

the function

$$\phi_j(\mathbf{x}) = \left(\mathbf{x}^2 + \sigma^2\right)^{\beta}, \qquad 0 < \beta < 1, \tag{3.11}$$

which for $\beta = 1/2$ is known as the multi-quadratic function, the cubic function

$$\phi_j(\mathbf{x}) = \mathbf{x}^3 \tag{3.12}$$

and the 'linear' function

$$\phi_j(\mathbf{x}) = \mathbf{x} \tag{3.13}$$

which all have the property that $\phi \to \infty$ as $x \to \infty$. The RBF network is represented by the diagram in Figure 3.3.

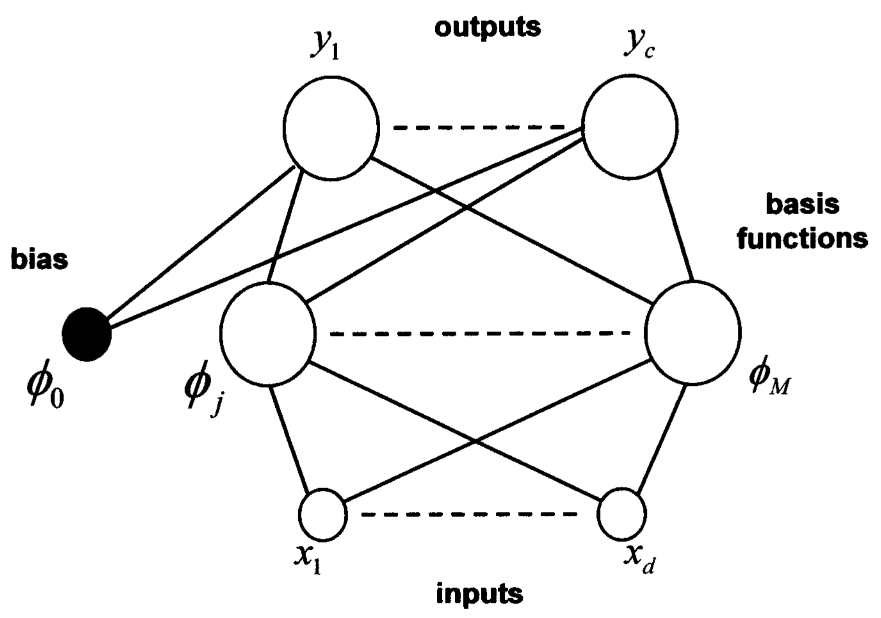

**Figure 3.5** Architecture of a radial basis function network (Bishop, 1995).

In Figure 3.5 each basis function acts like a hidden unit. The lines connecting basis function $\phi_j$ to the inputs represent the corresponding element $\mu_{ji}$ of the vector $\mu_j$. The weights $\omega_{kj}$ are shown as lines from the basis functions to the output units, and the biases are shown as weights from an extra 'basis function' $\phi_0$ whose output is fixed to 1. The Gaussian radial basis function considered above can be generalised to allow for arbitrary covariance matrix $\Sigma_j$ by changing the form of the basis function to

$$\phi_j(\mathbf{x}) = \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_j)^{\mathrm{T}} \sum\nolimits_j^{-1}(\mathbf{x}-\mu_j)\right\}. \tag{3.14}$$

Since the covariance matrices $\Sigma_j$ are symmetric, each basis function has $d(d+30)/2$ independent adjustable parameters (where $d$ is the dimensionality of the input space) (Bishop, 1995).

## 3.2.7 RBF network training

The training of the radial basis function takes place in two stages. In the first stage the input data set $\mathbf{x}^n$ alone is used to determine the parameters of the ($\mu_j$ and $\sigma_j$ for the spherical radial basis function). After the first stage of training the basis functions are then kept fixed and the second layer of weights are found in the second training phase presented below. When the bias parameters in Equation (3.6) are absorbed into the weights the resulting equation is

$$y_k(\mathbf{x}) = \sum_{j=0}^{M} \omega_{kj}\phi_j(\mathbf{x}) \tag{3.15}$$

where $\phi_0$ is an extra 'basis function' with activation value fixed at $\phi_0 = 1$. In matrix form this equation is

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}\boldsymbol{\varphi} \tag{3.16}$$

where $\mathbf{W} = (\omega_{kj})$ and $\boldsymbol{\varphi} = (\phi_j)$. Since the basis functions are considered fixed, the network is equivalent to a single–layer network that can be optimised by minimisation of a suitable error function. The sum-of-square error function is given by

$$E = \frac{1}{2}\sum_n \sum_k \left\{ y_k(\mathbf{x}^n) - t_k^n \right\}^2 \tag{3.17}$$

where $t_k^n$ is the target value for the unit $k$ when the network is presented with input vector $\mathbf{x}^n$. Since the error function is a quadratic function of the weights, its minimum can be found in terms of the solution of a set of linear equations. The weights are determined by the linear equation

$$\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{W}^{\mathrm{T}} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{T} \tag{3.18}$$

where $(\mathbf{T})_{nk} = t_k^n$ and $(\mathbf{\Phi})_{nj} = \phi_j(\mathrm{x}^n)$. The formal solution of the weights is given by

$$\mathbf{W}^{\mathrm{T}} = \mathbf{\Phi}^{\dagger}\mathbf{T}, \qquad \text{where } \mathbf{\Phi}^{\dagger} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}. \tag{3.19}$$

$\mathbf{\Phi}^{\dagger}$ is the pseudo-inverse of $\mathbf{\Phi}$. In practice Equations (3.18) are solved using singular value decomposition, to avoid problems due to possible ill conditioning of the matrix $\mathbf{\Phi}$.

For regression the basis function parameters can be found by treating the basis function centres and widths, along with the second-layer weights, as adaptive parameters to be determined by the minimisation of an error function. For the case of the sum-of-squares error, and spherical Gaussian basis functions Equation (3.7), the following expressions are obtained for the derivatives of the error function with respect to the basis function parameters

$$\frac{\partial E}{\partial \sigma_j} = \sum_n \sum_k \left\{ y_k\left(\mathrm{x}^n\right) - t_k^n \right\} \omega_{kj} \exp\left( -\frac{\left\| \mathrm{x}^n - \mu_j \right\|^2}{2\sigma_j^2} \right) \frac{\left\| \mathrm{x}^n - \mu_j \right\|^2}{\sigma_j^3} \tag{3.20}$$

$$\frac{\partial E}{\partial \mu_{ji}} = \sum_n \sum_k \left\{ y_k\left(\mathrm{x}^n\right) - t_k^n \right\} \omega_{kj} \exp\left( -\frac{\left\| \mathrm{x}^n - \mu_j \right\|^2}{2\sigma_j^2} \right) \frac{\left\| x_i^n - \mu_{ji} \right\|^2}{\sigma_j^2} \tag{3.21}$$

where $\mu_{ji}$ denotes the *i*th component of $\mu_j$. These expressions for the derivative are used in conjunction with standard optimisation strategies. The setting of the basis function parameters by supervised learning represents a non-linear optimisation problem that is computationally expensive and may be prone to finding local minima in the error function but if the relevant basis functions are identified the training procedures can be significantly speeded up (Bishop, 1995).

### 3.2.8 RBF simulation results from a preliminary investigation using data from the accelerated gear life test rig

In this section the suitability of the RBF neural networks for use in synchronous TDA model is assessed using gear vibration data from the accelerated gear life test rig. To achieve this the following steps were followed.

- Data pre-processing. The data was pre-processed as explained in Section 3.2.5 resulting in an input space $X_k$ of dimension (160×024) and a $t_k$ of dimensions (1×1024).
- Selecting number of basis. The number of number of basis was chosen between 1 and 10 and the one that resulted in the least square errors when simulating with unseen validation sets was selected. In this application 5 number of basis were

selected because they resulted in a small error without severe computational penalties.

- Selected type of basis functions. The activation functions that were investigated for setting the radial basis function structures were: the radially symmetric Gaussian function, the thin plate splines 'tps' $\phi(x) = x^2 \ln(x)$, and the $\phi(x) = x^4 \ln(x)$ activation functions. The output error function was defined as linear. The type that resulted in the least square errors when simulating with unseen validation sets was selected was the thin plate splines therefore it was implemented in this study.
- Selecting number of inputs. The number of inputs was chosen between 1 and 40 and the one that resulted in the least square errors when simulating with unseen validation sets was selected. In this application 40 inputs resulted in the smallest square errors, therefore resulting in a data reduction of 75 percent for calculating the TDA.

Figures 3.6 and 3.7 show the performance of RBF neural network on a preliminary study conducted using vibration data from the accelerated gear life test rig.

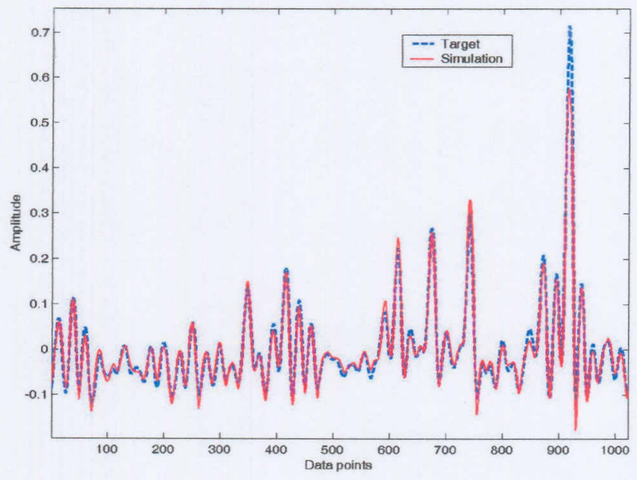

**Figure 3.6** RBF simulation results for network with 40 inputs and 5 basis functions superimposed on TDA obtained by direct averaging.

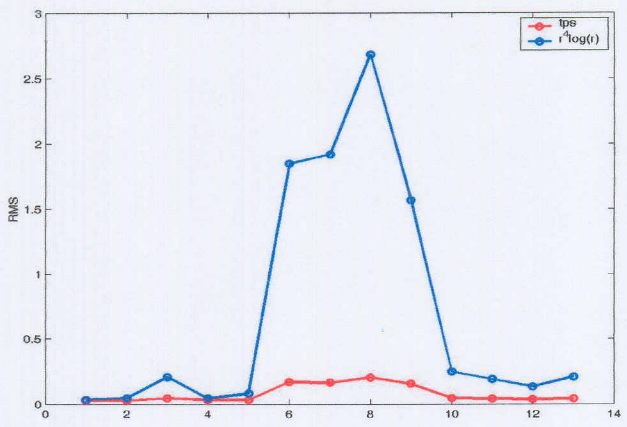

**Figure 3.7** Performance of the thin plate spline 'tps' and the $\boldsymbol{\varphi} = \mathbf{x}^4 \ln(\mathbf{x})$ basis functions on 13 validation sets.

Figure 3.6 shows the simulation results of a RBF neural network with 5 basis functions and 40 input vectors simulated using an unseen validation data set. From this simulation it is observed that the RBF network can be used to predict the time domain average of the gearbox signal using less data. Figure 3.7 shows the performance of thin-plate spline'tps' $\phi(x) = x^2 \ln(x)$, and the $\phi(x) = x^4 \ln(x)$, using the validation sets. From this simulation it is observed that the 'tps' activation function performs better than the $\phi(x) = x^4 \ln(x)$ activation function because it results in a constant prediction error for all the validation sets. This is because the 'tps' activation generalises well to all the validation sets as opposed to the $\phi(x) = x^4 \ln(x)$ activation function that does not generalise resulting in a peak in prediction error between validation set 5 and validation set 10. This is because the validation sets that were used during simulation were taken from different stages of the gear life and had different vibration signatures representing the progression in the level of damage. In this work 'tps' activation function is therefore selected for all RBF analysis.

## 3.3 Support vector machines

Traditional neural network approaches have suffered difficulties with generalisation, producing models that can overfit the data. This is a consequence of the optimisation algorithms used for parameter selection and the statistical measures used to select the 'best' model (Gunn, 1998). The foundations of Support Vector Machines (SVMs) have been developed by Vapnik (1995) and are gaining popularity due to many attractive features, and promising empirical performance. The SVM formulation embodies the Structural Risk Minimisation (SRM) principle, which has been shown to be superior (Gunn et al., 1997), to traditional Empirical Risk Minimisation (ERM) principle, employed by conventional neural networks. SRM minimises an upper bound on the expected risk, as opposed to ERM that minimises the error on the training data. It is this difference that equips SVMs with a greater ability to generalise, which is the goal in statistical learning. SVMs were originally developed for classification problems, but recently have been extended to regression problems (Vapnik et al., 1997). In this work we are interested in the regression properties of support vector machines and therefore the term support vector machines refers to SVMs applied to regression problems. The objective is to map the rotation synchronised gear vibration signal (input) to the ensemble average of the rotation synchronised gear vibration signals (TDA obtained from the *k* signals) using less rotation synchronised gear vibration signals than would otherwise be required to calculate the TDA using direct averaging.

## 3.3.1 Linear regression

When considering a simple linear regression problem of approximating a set of data,

$$D=\left\{\left(x^{1},y^{1}\right),\mathrm{K},\left(x^{l},y^{l}\right)\right\}, \qquad x\in ¡^{n}, y\in ¡ , \tag{3.22}$$

with a linear function ,

$$f\left(x\right)=\left\langle w,x\right\rangle+b. \tag{3.23}$$

The optimal regression function is given by the minimum of the functional,

$$\Phi\left(w,\xi\right)=\frac{1}{2}\left\|w\right\|^{2}+C\sum_{i}\left(\xi_{i}^{-}+\xi_{i}^{+}\right), \tag{3.24}$$

where $C$ is a pre-specified value, $w$ are the weights and $\xi^{-},\xi^{+}$ are slack variables representing the lower and upper constraints on the outputs of the system. When support vector machines are applied to regression problems loss functions that include a distance measure are used. Figure 3.8 below shows some of the possible loss functions.

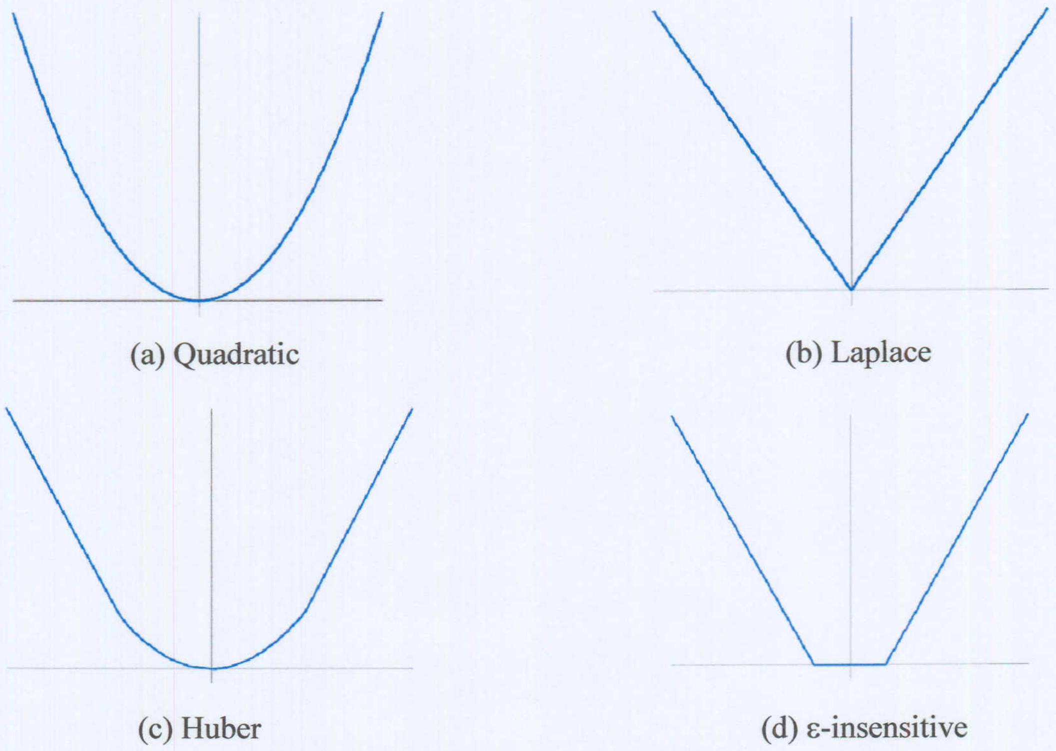

**Figure 3.8** Possible loss functions support vector regression.

The loss function in Figure 3.8 (a) corresponds to the conventional least squares error criterion. The loss function in Figure 3.8 (b) is a Laplacian loss function that is less sensitive to outliers than the quadratic loss function. The Huber loss function in Figure 3.8 (c) is a robust loss function with optimal properties when the underlying distribution of the data is not known. These three loss functions produce no sparseness in the support vector. To address this issue Vapnik proposed the loss function in Figure 3.8 (d) as an approximation to the Huber loss function. The ε-insensitive enables the sparse set of support vectors to be obtained.

Using the ε-insensitive loss function in Figure 3.8 (d),

$$L_{\epsilon}(y) = \begin{cases} 0 & \text{for} \quad |f(\mathrm{x}) - y| < \varepsilon \\ |f(\mathrm{x}) - y| - \varepsilon & \text{Otherwise} \end{cases} \tag{3.25}$$

the solution to the optimal regression function in Equation (3.24) is given by,

$$\max_{\alpha,\alpha^*} W\left(\alpha,\alpha^*\right) = \max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\left\langle x_i,x_j\right\rangle + \sum_{i=1}^{l}\alpha_i\left(y_i-\in\right)-\alpha_i^*\left(y_i+\in\right) \tag{3.26}$$

or alternatively,

$$\bar{\alpha},\bar{\alpha}^* = \arg\min_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\left\langle x_i,x_j\right\rangle - \sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right)y_i + \sum_{i=1}^{l}\left(\alpha_i+\alpha_i^*\right)\in \tag{3.27}$$

with constraints,

$$\begin{aligned} &0\le\alpha_i,\alpha_i^*\le C, \qquad i=1,\mathrm{K},l \\ &\sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right)=0. \end{aligned} \tag{3.28}$$

Solving Equation (3.26) with constraints in Equation (3.28) determines the Lagrange multipliers, $\alpha,\alpha^*$ and the regression function is given by Equation (3.23), where

$$\begin{aligned} \bar{w} &= \sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right)x_i \\ \bar{b} &= -\frac{1}{2}\left\langle \bar{w},\left(x_r+x_s\right)\right\rangle. \end{aligned} \tag{3.29}$$

The Karush-Kuhn-Tucker (KKT) conditions (Vapnik, 1995) that are satisfied by the solution are,

$$\bar{\alpha}_i\bar{\alpha}_i^* = 0, \qquad i=1,\mathrm{K},l. \tag{3.30}$$

Therefore the support vectors are points where exactly one of the Lagrange multipliers are greater than zero. When $\in=0$, we get the $L_1$ loss function and the optimisation problem is simplified to

$$\min_{\beta} \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\beta_i\beta_j\langle x_i,x_j\rangle-\sum_{i=1}^{l}\beta_i y_i \tag{3.31}$$

with constraints,

$$\begin{aligned}&-C\le\beta_i\le C, \qquad i=1,\mathrm{K}\ ,l\\ &\sum_{i=1}^{l}\beta_i=0,\end{aligned} \tag{3.32}$$

and the regression function is given by Equation (3.23), where

$$\begin{aligned}\bar{w}&=\sum_{i=1}^{l}\beta_i x_i\\ \bar{b}&=\frac{1}{2}\langle\bar{w},(x_r+x_s)\rangle.\end{aligned} \tag{3.33}$$

Details for the implementation of the other three loss functions are presented in Appendix D.

### 3.3.2 Non-linear regression

In cases where non-linear regression is required, non-linear mapping is used to map the data to a higher dimensional feature space where linear regression is performed. The kernel approach is employed to address the curse of dimensionality. The non-linear support vector regression solution, using the ε-insensitive loss function is given by

$$\begin{aligned}\max_{\alpha,\alpha^*}W(\alpha,\alpha^*)=\max_{\alpha,\alpha^*}&\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i^*(y_i-\in)-\alpha_i(y_i+\in)\\ &-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i^*-\alpha_i)(\alpha_j^*-\alpha_j)K(x_i,x_j)\end{aligned} \tag{3.34}$$

with constraints,

$$\begin{aligned}&0\le\alpha_i,\alpha_i^*\le C, \qquad i=1,\mathrm{K}\ ,l\\ &\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)=0.\end{aligned} \tag{3.35}$$

Solving Equation (3.34) with the constraints in Equation (3.35) determines the Lagrange multipliers, $\alpha, \alpha^*$ and the regression function is given by

$$f(\mathrm{x}) = \sum_{i=1}^{l} \left(\bar{\alpha}_i - \bar{\alpha}_i^*\right) K\left(\mathrm{x}_i, \mathrm{x}\right) + \bar{b} \tag{3.36}$$

where

$$\begin{aligned} \langle \bar{w}, x \rangle &= \sum_{SVs} \left(\alpha_i - \alpha_i^*\right) K\left(x_i, x_j\right) \\ \bar{b} &= -\frac{1}{2} \sum_{i=1}^{l} \left(\alpha_i - \alpha_i^*\right)\left(K\left(x_i, x_r\right) + K\left(x_i, x_s\right)\right). \end{aligned} \tag{3.37}$$

The optimisation criteria for other loss functions are similarly obtained by replacing the dot product with kernel functions. The ε-insensitive loss function is attractive because unlike the quadratic and Huber cost functions, where all the data points will be support vectors, the support vector solution can be sparse, therefore the ε-insensitive loss function was selected in this work. Different kernels were investigated for mapping the data to a higher dimensional feature space where linear regression is performed.

### 3.3.3 SVM simulation results from a preliminary investigation using data from the accelerated gear life test rig

In this section the suitability of SVMs for use in synchronous TDA model is assessed using gear vibration data from the accelerated gear life test rig. To achieve this the following steps were followed.

- Data pre-processing. The data was pre-processed as explained in Section 3.2.5 resulting in an input space $X_k$ of dimension $(160 \times 1024)$ and a $t_k$ of dimensions $(1 \times 1024)$.
- Selected type of Kernel functions. The Kernel functions that were investigated in this study were: Exponential Radial Basis Function (ERBF), the Gaussian Radial Basis function (RBF), the spline and the b-spline kernels and the one that resulted in the smallest error was selected. (see Appendix E).
- Selecting number of inputs. The number of inputs was chosen between 1 and 40 and the one that resulted in the least square errors when simulating with unseen

validation sets was selected. In this application 40 inputs resulted in a smallest square errors. This is effectively a data reduction of 75 percent.

Figures 3.9 to 3.11 show the performance SVMs on a preliminary study conducted using vibration data from the accelerated gear life test rig.

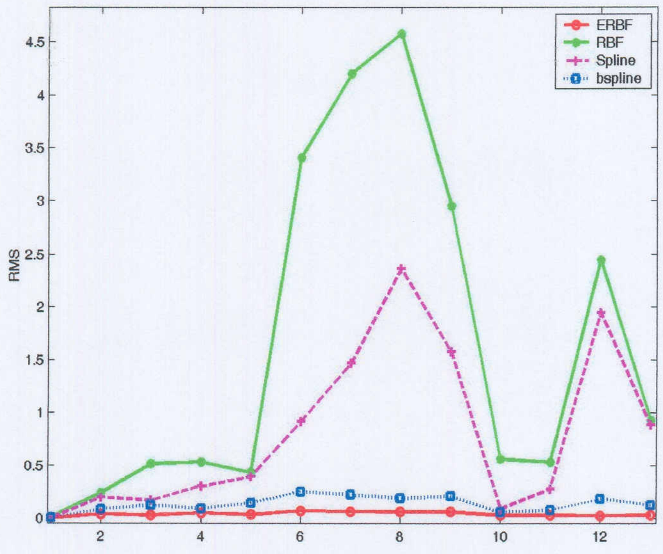

**Figure 3.9** Performance of different Kernel function on 13 unseen validation sets

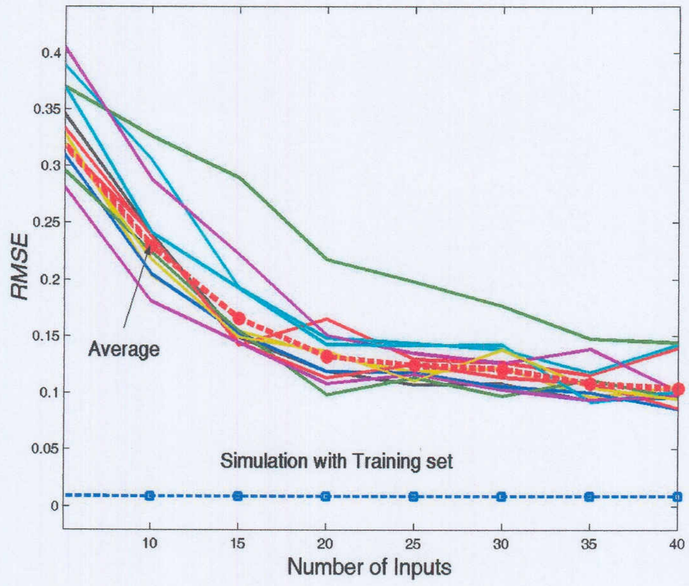

**Figure 3.10** RMS of SVM simulation vs. Number of input signals

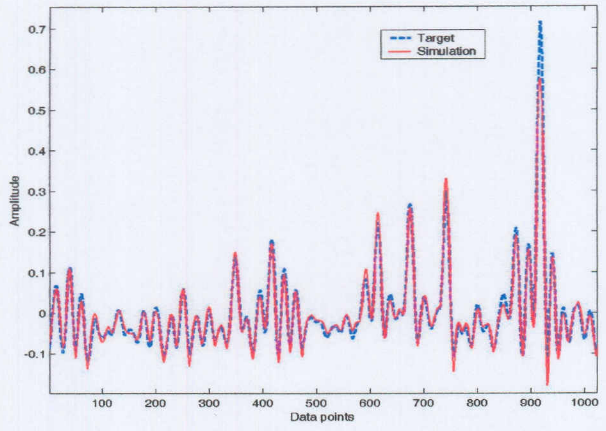

**Figure 3.11** Prediction of SVM with 40 inputs and ERBF kernel superimposed on target

Figure 3.9 shows the performance of different Kernel functions on 13 validation sets from different stages of the gear life. It is observed that the ERBF Kernel gives the best mapping for all the validation sets. It is also observed that the RBF, spline and b-spline Kernel functions have a peak at validation set 8 and validation set 12. This is because the SVMs were trained with data from the early stages of the gear life. As the gear life progresses the vibration signature changes significantly. The peaks at validation set 8 and validation set 12 indicate that SVMs with RBF, spline and b-spline Kernel functions do no generalise well to the vibration signatures at these stages of gear life. Figure 3.10 shows the performance of SVM with an ERBF kernel as a function of number of input vectors on validation sets. From this plot it is observed that the RMS decreases as the number of inputs increases. Figure 3.12 shows the simulation results SVM with an ERBF Kernel and 40 unseen input signal superimposed on the TDA calculated using the direct time domain averaging approach. This plot indicates that SVMs can be effectively used to predict the TDA of the gearbox signal using less data than would be used to calculate the TDA using the direct averaging approach.

## 3.4 Conclusion

From the simulation results obtained in the preliminary study, it is observed that all three formulations can successfully map the input space to the TDA calculated by time domain averaging using only 25 percent of the input data. It is therefore concluded from the preliminary study that all three formulations are suitable for use in the development of a synchronous filter for time domain averaging of gear vibration data. Chapter 4 deals with the development issues for the synchronous filter.