Old Dominion University

ODU Digital Commons

Electrical & Computer Engineering Theses & Dissertations

Electrical & Computer Engineering

Summer 2006

Dimensionality Reduction Using Non-Linear Principal Components Analysis

Tara Singh
Old Dominion University

Follow this and additional works at: https://digitalcommons.odu.edu/ece_etds

Part of the Computer Sciences Commons, Data Science Commons, and the Statistical Methodology Commons

Recommended Citation

Singh, Tara. "Dimensionality Reduction Using Non-Linear Principal Components Analysis" (2006). Master of Science (MS), Thesis, Electrical & Computer Engineering, Old Dominion University, DOI: 10.25777/7b1p-6m64
https://digitalcommons.odu.edu/ece_etds/530

This Thesis is brought to you for free and open access by the Electrical & Computer Engineering at ODU Digital Commons. It has been accepted for inclusion in Electrical & Computer Engineering Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

DIMENSIONALITY REDUCTION USING NON-LINEAR PRINCIPAL COMPONENTS ANALYSIS

By

Tara Singh
B. Tech (Electrical), June 2001, G. B. Pant University of Ag. & Tech, India

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

ELECTRICAL ENGINEERING

OLD DOMINION UNIVERSITY
August 2006

Approved by:

[illegible]
Stephen. A. Zahorian (Director)

[illegible]
Vijayan K. Asari (Member)

Min Song (Member)

ABSTRACT

Dimensionality Reduction using Nonlinear Principal Components Analysis

Tara Singh
Old Dominion University, August 2006
Director: Dr. Stephen A. Zahorian

Advances in data collection and storage capabilities during the past decades have led to an information overload in most sciences. Traditional statistical methods break down partly because of the increase in the number of observations, but mostly because of the increase in the number of variables associated with each observation. While certain methods can construct predictive models with high accuracy from high-dimensional data, it is still of interest in many applications to reduce the dimension of the original data prior to any modeling of the data. Patterns in the data can be hard to find in data of high dimensionality, where the luxury of graphical representation is not available. Linear PCA is a powerful tool for analyzing this high-dimensional data. A common drawback of these classical methods is that only linear structures can be correctly extracted from the data.

If the data represent the complicated interaction of features, then a linear subspace may be a poor representation and a nonlinear subspace may be needed. It is hypothesized that nonlinear (curved) basis vectors will be more efficient for representing some types of data than are linear PCA basis vectors. The neural network method is among the best methods to implement NLPCA due to its capability of accurately approximating any continuous nonlinear function. Linear and nonlinear PCA are compared using speech classification experiments with reduced dimensionality data. For some cases, NLPCA is found to be advantageous over classical PCA.

Copyright © 2006 Old Dominion University, All rights reserved.

Dedicated to my parents
Kishan Singh Negi and Devki Negi

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to my thesis advisor, Dr. Stephen A. Zahorian, for his invaluable advice, guidance, motivation and patience throughout the research work. Without his support, this thesis would not have been possible.

I would like to thank Dr.Vijayan K. Asari and Dr. Min Song for graciously agreeing to be on my thesis advisory committee and for their valuable time and generous assistance.

I would also like to thank all my friends and colleagues in the Speech Communication Lab for establishing a positive working environment and helping me throughout my research work.

Last but not least, I wish to thank my family and friends for their unconditional love and support throughout my educational years.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Advances in data collection and storage capabilities during the past decades have led to an information overload in most sciences. Researchers working in domains as diverse as engineering, astronomy, biology, remote sensing, economics, and consumer transactions, face larger and larger observations and simulations on a daily basis. Such datasets, in contrast with smaller, more traditional datasets that have been studied extensively in the past, present new challenges in data analysis. Traditional statistical methods break down partly because of the increase in the number of observations, but mostly because of the increase in the number of variables associated with each observation [11].

In traditional statistical data analysis, we think of observations of instances of particular phenomena (e.g. instance $\leftrightarrow$ a particular human being). Each instance is described by a vector of measured values or computed variables (e.g. blood pressure, weight, height). In traditional statistical methodology, typically there are many observations and a few, well chosen variables. The trend today is towards more observations accompanied by larger numbers of variables – an automatic, systematic collection of informative detail about each observed instance [11]. We are seeing examples where the observations gathered on individual instances are curves, spectra or images, or even movies, so that a single observation has dimensions in the thousands or billions, while there are only tens or hundreds of instances available for study. The dimension of the data is the number of variables that are measured on each observation.

1.1 Curse of high dimensionality

The curse of dimensionality refers to the apparent intractability of systematically searching through a high-dimensional space, the apparent intractability of accurately approximating a general high-dimensional function, or the apparent intractability of integrating a high-dimensional function [5]. In the field of pattern classification, the curse of dimensionality refers to the phenomenon that classification results (on test data) often decrease as the dimensionality increases.

High-dimensional datasets present many mathematical challenges as well as some opportunities, and are giving rise to new theoretical developments. One of the problems with high-dimensional datasets is that, in many cases, not all the measured variables are “important" for understanding the underlying phenomena of interest. While certain methods can construct predictive models with high accuracy from high-dimensional data, it is still of interest in many applications to reduce the dimension of the original data prior to any modeling of the data.

In many applications of data mining, the high dimensionality of the data restricts the choice of data processing methods. Such application areas include the analysis of market basket data, text documents, image data, and so on. In these cases the dimensionality is large due to an availability of alternative products, a large vocabulary, or the use of large image windows, respectively. A statistically optimal way of dimensionality reduction is to project the data onto a lower-dimensional orthogonal subspace that captures as much of the variation of the data as possible [19]. The best (in a mean-square sense) and most widely used way to do this is principal components analysis (PCA). It is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences.

Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analyzing data. PCA reduces dimensionality without much loss of information, generally making it much easier to identify patterns in the data.

1.2 Linear principal components analysis

Principal components analysis (PCA) is a multivariate procedure which rotates the data such that maximum variability is projected onto a relatively small number of axes. Essentially, a set of correlated variables is transformed into a set of uncorrelated variables which is ordered by reducing variability. The uncorrelated variables are linear combinations of the original variables, and the last of these variables can often be removed with minimum loss of real data.

The first principal component is the combination of variables that explains the greatest amount of variation. The second principal component defines the next largest amount of variation independent of the first principal component. There can be as many principal components as there are variables.

It can be viewed as a rotation of the existing axes to new positions in the space defined by the original variables. In this new rotation, there will be no correlation between the new variables defined by the rotation. The first new variable contains the maximum amount of variation; the second new variable contains the maximum amount of variation unexplained by the first and orthogonal to the first, etc.

There are several algorithms for calculating the principal components. Given the same starting data they will produce the same results with one exception. This exception is that, if at some point, there are two or more possible rotations that contain the same "maximum" variation then any of these may be used. In two dimensions, the data cloud would look like a circle, instead of an ellipse. In a circle, any rotation would be equivalent. In an elliptical data cloud, the first component would be parallel to the major axis of the ellipse.

1.3 Mathematical form of linear PCA

Principal components analysis (PCA) is a classical statistical method. This linear transform has been widely used in data analysis and compression. Principal component analysis is based on the statistical representation of a random variable. Suppose we have a random vector population x, where

$$x = (x_1, x_2 \ldots\ldots, x_n)^T \qquad \textbf{(1.1)}$$

and the mean of that population is denoted by

$$\mu_X = E\{x) \qquad \textbf{(1.2)}$$

and the covariance matrix of the same data set is

$$C_x = E\{(x-\mu_x)(x-\mu_x)^T\}. \qquad \textbf{(1.3)}$$

The components of C_x, denoted by C_{ij}, represent the covariances between the random variable components x_i and x_j. The component c_{ii} is the variance of the component x_i. The variance of a component indicates the spread of the component values around its mean value. If two components x_i and x_j of the data are uncorrelated, their covariance is zero

$(C_{ij} = C_{ji} = o)$. The covariance matrix is, by definition, always symmetric. From a sample of vectors $x_1,......., x_2$, we can calculate the sample mean and the sample covariance matrix as the estimates of the mean and the covariance matrix.

From a symmetric matrix such as the covariance matrix, we can calculate an orthogonal basis by finding its eigenvalues and eigenvectors. The eigenvectors e_i and the corresponding eigenvalues λ_i are the solutions of the equation

$$C_x e_i = \lambda_i e_i, i = 1,......, n. \quad \textbf{(1.4)}$$

For simplicity we assume that the λ_i are distinct. These values can be found, for example, by finding the solutions of the characteristic equation

$$|C_x - \lambda I| = 0, \quad \textbf{(1.5)}$$

where the I is the identity matrix having the same order as C_x and the |.| denotes the determinant of the matrix. If the data vector has *n* components, the characteristic equation becomes of order *n*. This is easy to solve only if *n* is small. The determination of eigenvalues and corresponding eigenvectors is a non-trivial task, and many methods exist [20]. One way to solve the eigenvalue problem is to use a neural solution to the problem. The data is the input, and the network can be trained to converge to the PCA solution. More typically, matrix methods are used to find the eigenvalues and eigenvectors of symmatric matrices [18]. Mathematical programming languages, such as Matlab, typically contain built in routines for this task. Note that the eigenvectors of the covariance matix are the PCA basis vectors. The amount of variance accounted for by each principal component is proportional to the size of the corresponding eigenvalue. By ordering the eigenvectors in the order of descending eigenvalues (largest first), one can create an ordered orthogonal basis with the first eigenvector having the direction of largest variance of the data. In this way, we can find directions in which the data set has the most significant amounts of energy.

Suppose one has a data set for which the sample mean and the covariance matrix have been calculated. Let A be a matrix consisting of eigenvectors of the covariance matrix as the row vectors. Additonally, for a specifiied mean square representation error, an inspection of eigenvalues allows a straightforward calculation of the number of eigenvectors needed.

By transforming a data vector x, we obtain

$$y = A(x - \mu_x) \tag{1.6}$$

which is a point in the orthogonal coordinate system defined by the eigenvectors. Components of y can are the coordinates in the orthogonal base. We can reconstruct the original data vector x from y by

$$x = A^T y + \mu_x \tag{1.7}$$

using the property of an orthogonal matrix that $A^{-1} = A^T$ where A^T is the transpose of a matrix A. The original vector x was projected on the coordinate axes defined by the orthogonal basis. The original vector can thus be reconstructed by a linear combination of the orthogonal basis vectors.

Instead of using all the eigenvectors of the covariance matrix, we may represent the data in terms of only a few basis vectors of the orthogonal basis. If we denote the matrix having the K first eigenvectors as rows by A_k, we can create a similar transformation as given above, $y = A_k(x - \mu_x)$ and $x = A_k^T y + \mu_x$

This means that we can project the original data vector on the coordinate axes having the dimension *K* and transforming the vector back by a linear combination of the basis vectors. This minimizes the mean-square error between the data and this representation with a given number of eigenvectors.

1.4 Example of linear PCA

In this example, we take a simple set of 2-D data as shown in Fig.1.1 and apply PCA to determine the principal axes. Although the technique can be used with many dimensions, 2-dimensional data makes it easier to visualize. Principal component analysis is then performed. First the covariance matrix is calculated [9].

Cov (x) =

$$\begin{matrix} 7.5649 & 3.8464 \\ 3.8464 & 3.2451 \end{matrix}$$

In step 2, eigenvectors and eigenvalues for this matrix are calculated. The principal components are the eigenvectors arranged in order of decreasing eigenvalues. For this example, the principal components are:

$$PC = \begin{matrix} 0.8630 & -0.5052 \\ 0.5052 & 0.8630 \end{matrix}$$

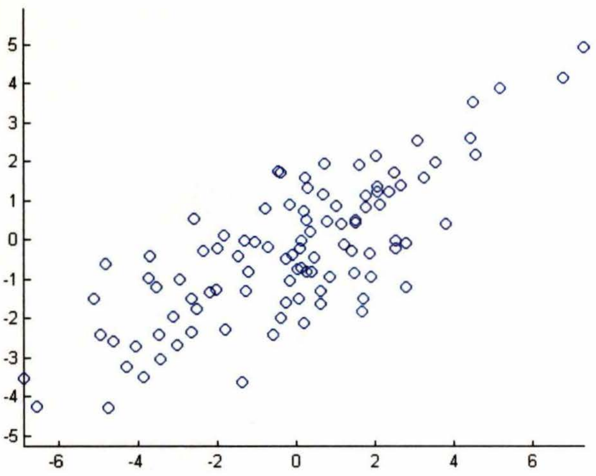

Fig. 1.1. Sample 2-D data for performing PCA.

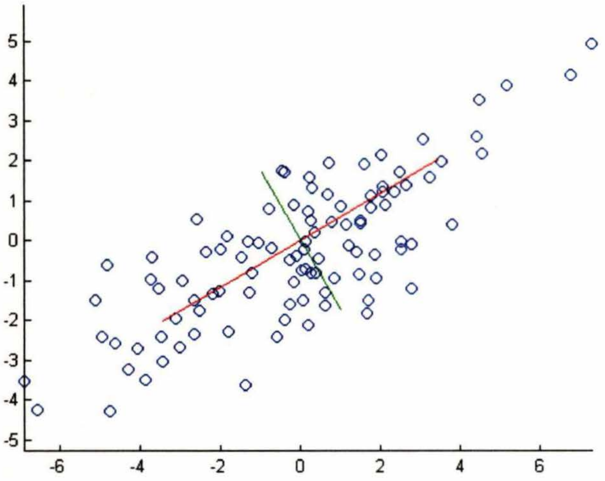

Fig. 1.2. The red line represents the direction of the first principal component and the green is the second. The first principal component lies along the line of greatest variation, and the second lies perpendicular to it. Where there are more than two dimensions, each succeeding component is perpendicular to all the already selected components, and along the line of next greatest variation [9].

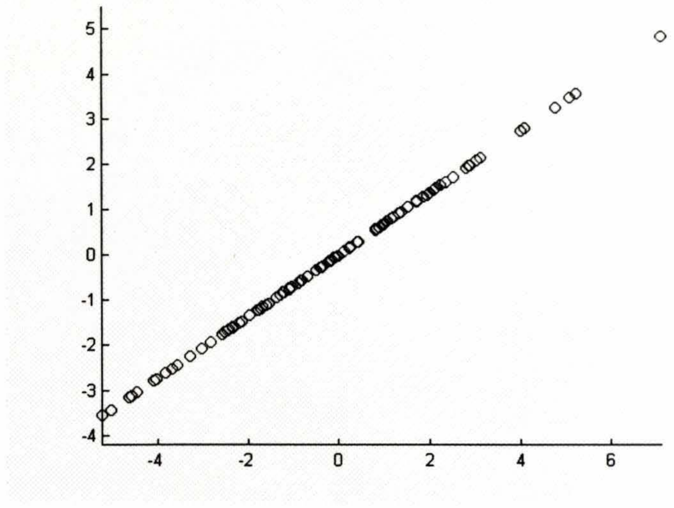

Fig.1.3. The most common use for PCA is to reduce the dimensionality of the data while retaining the most information. Here the data from Fig 1.2 is transformed to a 1-D space.

1.5 General limitation of linear methods

Methods in multivariate statistical analysis are essential for working with large amounts of geophysical data, data from observational arrays, from satellites, or from numerical model output. In classical multivariate statistical analysis, there is a hierarchy of methods, starting with linear regression at the base, followed by principal component analysis (PCA) and finally canonical correlation analysis (CCA). A multivariate time series method, the singular spectrum analysis (SSA), has been a fruitful extension of the PCA technique [1]. The common drawback of these classical methods is that only linear structures can be easily extracted from the data. For example, linear PCA yields a k-dimensional linear subspace of features that best represents the full data according to minimum square error criterion. If the data represent the complicated interaction of features, then the linear subspace may be a poor representation and a nonlinear subspace may be needed. Fig.1.4 illustrates the limitation of linear PCA [6]. The straight line fit to the data obtained by linear PCA does not provide much information about original curve shaped data. A nonlinear method might "discover" the curve that the data lies on.

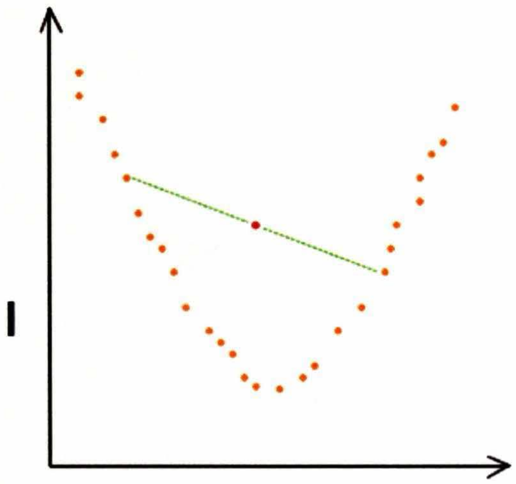

Fig.1.4. Straight line obtained using linear PCA, a poor representation of the original data [6].

1.6 Thesis outline

The general objectives of this thesis are to investigate and develop algorithms for performing dimensionality reduction using nonlinear methods. It is hypothesized that nonlinear (curved) basis vectors will be more efficient for representing some types of data than are linear PCA basis vectors. Linear and nonlinear PCA will be compared using speech classification experiments with reduced dimensionality data.

In chapter 2, a detailed theoretical and mathematical description of Nonlinear Principal Component Analysis (NLPCA) for data compression and dimensional reduction of nonlinear data is presented. Chapter 3 contains the description of NLPCA algorithms used. Experiments with constructed data are presented. In particular, random data is created in two and three dimensional spaces, but constrained so that the data lies on a curved surface in a lower dimensional space. NLPCA is then applied to evaluate the effectiveness of the NLPCA method. The degree of similarity between the original data and recreated data after dimensionality reduction is used to judge the effectiveness of the algorithm. In chapter 4, experiments with the speech vowel data are presented, and results are shown and analyzed. Finally the conclusion and scope of future work are presented in chapter 5.

CHAPTER II

NONLINEAR PRINCIPAL COMPONENTS ANALYSIS

This chapter presents a theoretical and mathematical explanation of nonlinear PCA. In section 2.1, a brief introduction about NLPCA is given in which its need and importance in handling real world problems are described. Section 2.1 describes a nonlinear transformation which forms the basis of NLPCA. This section also provides motivation for using neural networks for implementation of NLPCA. Section 2.3 describes, in detail, the neural network implementation of NLPCA which include the brief introduction of neural networks, its limitations, and finally the methodology to apply NLPCA using neural networks.

2.1 Introduction

Linear PCA finds a straight line which passes through the middle of a data cluster. However, a straight line is not a good fit to nonlinear data such as data which follows a curve or a Gaussian surface. The natural extension of linear PCA is to generalize the straight line basis vectors to curved lines. This technique of finding "best fitting" curves to data in a reduced dimensionality space is commonly known as Nonlinear Principal Component Analysis (NLPCA). The fundamental difference between NLPCA and Linear PCA is that Linear PCA only allows the linear mapping between data and its Principal component, while NLPCA allows a nonlinear mapping [1].

A linear principal component minimizes the sum of the orthogonal deviations between a straight line and the data while the nonlinear approach summarizes the data by a smooth curve (principal curve). Principal curves are a generalization of principal components. The calculation of principal curves essentially contains two steps--a projection step and a smoothing step [19]. The calculation is generally started with the principal component as the initial curve. In the projection step, the data points are projected onto the curve. Then, in the smoothing step, the curve is smoothed using techniques such as the locally weighted regression smoother or kernel smoothers. The procedure is iterated between the two steps until convergence results.

NLPCA can be used in a similar way to PCA in data summarization, data visualization, and data exploration. However, unlike linear PCA, NLPCA is ideally suited to nonlinearly correlated data [8]. By nonlinearly correlated data, we simply mean data that is highly constrained in a subspace of the original features along a curved subspace. In principal component analysis, the principal loading vectors are used as a model to generate principal scores for the new data. However the principal curve procedure does not calculate any nonlinear loadings. In industrial process applications, it is desirable to have a nonlinear principal model which can be used to generate nonlinear principal components for new data [3]. In nonlinear PCA, just as with linear PCA there is no response variable; hence, it is more suited to feature extraction than prediction.

Nonlinear PCA enables multivariate process performance monitoring of nonlinear processes. However, if linear principal components analysis reduces the dimensionality of the problem satisfactorily, then a linear approach to monitoring must be adopted since the interpretation of a linear technique is more straightforward for engineers and operators. Movements of scores from a region of nominal operation together with increasing squared prediction error values identify changes in process other than common causes. In some cases, especially with linear systems, the movement of the scores can be visibly different from those of the nominal operating region with different faults, thus causing the scores to move in different directions.

2.2 Nonlinear transformation and motivation for using neural networks for NLPCA

For nonlinear data, the basic concept is not to apply PCA directly to the given data but rather to a transformed version of the data [7]. More precisely, we seek a nonlinear transformation:

$$\phi(.) : R^D \rightarrow R^M \tag{2.1}$$

$$x \rightarrow \phi(x)\,,$$

be such that the structure of the resulting data $\{\phi(x_i)\}$ becomes (significantly more) linear. In machine learning, $\phi(x_i)$ is called the "feature" of the data point x and R^M is called the "feature space". Define the matrix:

$$\Phi = [\,\phi(x_1), \ldots\ldots, \phi(x_N)] \in R^{M*N}\,. \tag{2.2}$$

The principal components in the feature space are given by the eigenvectors of the sample feature covariance matrix:

$$\Sigma_{\phi(x)} = (1/N)\sum_{i=1}^{N} \phi(x_i)\phi(x_i)^T = (1/N)\Phi\Phi^T \in R^{M*M}\,. \tag{2.3}$$

Let $v_i \in R^M$ be the eigenvectors:

$$\Sigma_{\phi(x)}\, v_i = \lambda_i v_i, i = 1, \ldots\ldots, M \tag{2.4}$$

then the d "nonlinear principal components" of every data point x are given by

$$y_i = v_i^T \phi(x) \in R, i = 1, \ldots.., d\,. \tag{2.5}$$

In general, we do not expect that the map $\phi(.)$ is given together with the data. In many cases, searching for the proper map is a difficult task limiting the use of nonlinear PCA. However, in some practical applications, good candidates for the map $\phi(.)$ can be found from the nature of the problem. For an arbitrary nonlinear relationship expressed by $\phi(.)$, a neural network is an excellent approach to use because of its universal approximation property [15]. It is widely known that mapping performed by a neural network can approximate any continuous function with arbitrarily desired accuracy [2]. The concept of extracting features from highly nonlinear data has been discussed by a number of researchers with most techniques reported in the literature based upon artificial neural networks. This is possible due to the capability of neural networks to provide a nonlinear transformation of a feature space.

2.3 Nonlinear principal component analysis by the neural network method

This section describes in detail, the implementation of NLPCA using neural networks. This section is based on the work of William W. Hsieh, University of British Columbia, Canada [1].

2.3.1 Feed forward neural network models

The nonlinear neural network (NN) models originated from research aimed at understanding how the brain functions with its networks of interconnected neurons. There are many types of NN models. Some are only of interest to neurological researchers, while others are general nonlinear data techniques. The most widely used NN models are feedforward NNs, also called multilayer perceptrons, which perform nonlinear regression and classification. The basic architecture consists of a layer of input neurons x_i (a neuron is simply a variable in NN jargon) linked to a layer or more of hidden neurons, which are, in turn, linked to a layer of output neurons y_j.

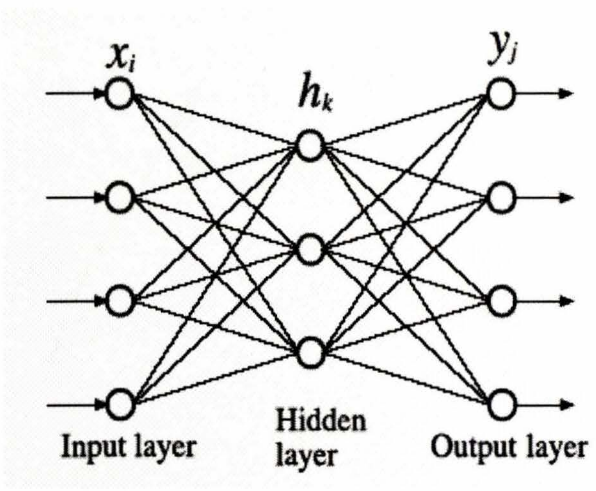

Fig.2.1. Schematic diagram of a feed forward neural network (NN) model, with one "hidden" layer of neurons (i.e. variable denoted by circles) sandwiched between the input layer and the output layer. In the feed forward NN model the information only flows forward starting from the input neurons. Increasing the number of hidden neurons increases the number of model parameters [1].

In fig 2.1, there is only one layer of hidden neurons h_k. A transfer function (an activation function in the NN) maps from the input to the hidden neurons. There are a variety of choices for the transfer function, with the hyperbolic tangent function being a common one i.e.

$$h_k = \tanh(\sum_i w_{ki}x_i + b_k), \qquad \textbf{(2.6)}$$

where w_{ki} and b_k are the weight and bias parameters respectively. The $\tanh(z)$ function is a sigmoidal shaped function, where its two asymptotic values of ± 1 as $z \to \pm\infty$ can be viewed as representing the two states of a neuron (at rest or activated), depending on the strength of the excitation z. (If there is more than one hidden layer, then equations of the same form as equation (2.6) are used to calculate the values of the next layer of the hidden neurons from the current layer of neurons). When the feed forward NN is used for nonlinear regression, the output neurons y_i are usually calculated by a linear combination of the neurons in the preceding layer, i.e.

$$y_j = \tilde{\Sigma} w_{jk} h_k + \tilde{b}_j \ . \tag{2.7}$$

Given observed data y_{oj}, the optimal values for the weight and bias parameters $(w_{ki}, \tilde{w}_{jk}, b_{kj}$ and $\tilde{b}_j)$ are found by training the NN, i.e. performing a nonlinear optimization, where the cost function or objective function

$$J = \langle \sum_j (y_i - y_{oj})^2 \rangle \tag{2.8}$$

is minimized, with J the mean squared error (MSE) of the output. The NN forms a nonlinear regression relation $y_j = f_j(x)$. To approximate a set of continuous functions f_i, only one layer of hidden neuron is enough, provided enough hidden neurons are used in that layer [15]. The NN with one hidden layer is commonly called a two layer NN, as there are two layers of mapping from input to output (equations 2.6 and 2.7). However, there are other conventions for counting the number of layers, and some authors refer to a two layer NN as a three layer NN since there are three layers of neurons.

2.3.2 Limitations of the NN method: Local minima and overfitting

The main difficulty of the NN method is that the gradient search for the NN parameters often encounters multiple local minima in the cost function. This means that starting from different initial guesses for the parameters, the gradient algorithm may converge to different local minima. Many approaches have been proposed to alleviate this problem. A common approach involves multiple optimization runs starting from different random initial parameters so that, hopefully, not all runs will be stranded at shallow local minima.

Another problem with the NN method is overfitting, i.e., fitting to the noise in the data, because of the tremendous flexibility of the NN to fit the data. With enough hidden neurons, the NN can fit the data, including the noise to arbitrary accuracy. Thus, for a network with many parameters, reaching the global minimum may mean nothing more than finding a badly overfitted solution. Usually, only a portion of the data record is used to train (i.e. fit) the NN model; the other is reserved to validate the model. If too many hidden neurons are used, then the NN model fit to the training data will be excellent, but the NN model fit to the validation data will be poor thereby allowing the researchers to gauge the appropriate number of hidden neurons. During the optimization process, it is also common to monitor the MSE over the training data and over the validation data separately. As the number of iterations of the optimization algorithm increases, the MSE calculated over the training data will decrease; however beyond a certain number of iterations the MSE over the validation data will begin to increase, indicating the start of overfitting and hence appropriate time to stop the optimization process. Another approach to avoid overfitting is to add weight penalty terms to the cost function [1]. Yet another approach is to compute an ensemble of NN models stating from different random parameters. The mean of the ensemble of NN solutions tends to give a smoother solution than the individual NN solutions. For poor quality data sets (short, noisy data records) the problems of local minima and overfitting could leave nonlinear NN methods incapable of offering any advantage over linear methods [1] [12].

2.3.3 NN architecture for NLPCA

To perform NLPCA, the feed forward NN in Fig.2.2 contains three hidden layers of neurons between the input and output layers of variables. The NLPCA is basically a standard feed forward NN with four layers of transfer functions mapping from the input to the outputs. One can view the NLPCA network as composed of two standard two layer feed forward NNs placed one after the other. First a two-layer network maps from the input x through a hidden layer to the bottleneck layer with only one neuron u, i.e. a nonlinear mapping $u = f(x)$.

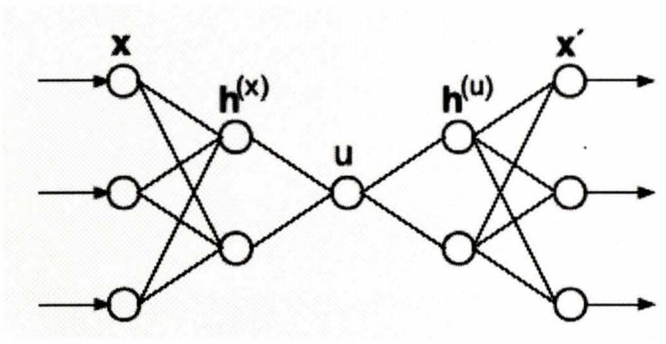

Fig.2.2. The NN model for calculating nonlinear PCA (NLPCA).

Fig.2.2 shows the NN model for NLPCA. There are 3 `hidden' layers of variables or `neurons' (denoted by circles) sandwiched between the input layer x on the left and the output layer x' on the right. Next to the input layer is the encoding layer, followed by the `bottleneck' layer (with a single neuron u), which is then followed by the decoding layer. A nonlinear function maps from the higher dimension input space to the lower dimension bottleneck space, followed by an inverse transform mapping from the bottleneck space back to the original space represented by the outputs, which are to be as close to the inputs as possible by minimizing the cost function in equation (2.9). Data compression is achieved by the bottleneck, with the bottleneck neuron giving u, the nonlinear principal component. [1] [14].

The next two-layer feed forward NN inversely maps from the reduced dimensionality u back to the original higher dimensional x space, with the objective that outputs, $x' = g(u)$ be as close as possible to the input x (thus the NN is said to be as autoassociative). Note $g(u)$ nonlinearly generates a curve in x space and hence a one dimensional (1-D) approximation of the original data. To minimize the MSE of this approximation, the cost function

$$j = < \|x - x'\|^2 > \qquad \textbf{(2.9)}$$

is minimized to solve for the weights and bias parameters of the NN. Squeezing the input information through a bottleneck layer with only one neuron accomplishes the dimensionality reduction. In effect, the linear relation in PCA is now generalized to $u = f(x)$, where f can be any nonlinear continuous function representable by a feed forward NN mapping from the input layer to the bottleneck layer and instead of $\langle\|x(t) - au(t)\|^2\rangle$, $\langle\|x - g(u)\|^2\rangle$ is minimized. If all the transfer functions are replaced by identity functions, linear PCA will result as the nonlinear capability of NLPCA was removed. In such a case, the forward map to u involves only a linear combination of the original variables as in linear PCA.

2.3.4 Mathematical model for NN based NLPCA

In Fig.2.2, the transfer function f_1 maps from x, the input column vector of length l, to the first hidden layer (the encoding layer), represented by $h^{(x)}$, a column vector of length m, with elements:

$$h_k^{(x)} = f_1[(w_1^{(x)} x + b_1^{(x)})_k], \qquad \textbf{(2.10)}$$

Where $w_1^{(x)}$ is an $m*l$ weight matrix, $b_1^{(x)}$ is a column vector of length m containing the bias parameters, and $k = 1,....,m.$ Similarly, a second transfer function f_2 maps from the encoding layer to the bottleneck layer containing a single neuron, which represents the nonlinear principal component u,

$$u = f_2(w_2^{(x)}.h^{(x)} + b_2^{-(x)}). \quad \textbf{(2.11)}$$

The transfer function f_1 is generally nonlinear (usually the hyperbolic tangent or the sigmoidal function), while f_2 is usually the identity function.

Next a transfer function f_3 maps from u to the final hidden layer (the decoding layer) $h^{(u)}$,

$$h_k^{(u)} = f_3[(w_3^{(u)}u + b_3^{(u)})_k], \quad \textbf{(2.12)}$$

$(k = 1,.....,m)$, followed by f_4 mapping from $h^{(u)}$ to x', the output column vector of length l, with

$$x'_i = f_4[(w_4^{(u)}h^{(u)} + b_4^{-(u)})_i]. \quad \textbf{(2.13)}$$

The cost function $j = \langle \|x - x'\|^2 \rangle$ is minimized by finding the optimal values of $w_1^{(x)}, b_1^{(x)}, w_2^{(x)}, b_2^{-(x)}, w_3^{(u)}, b_3^{(u)}, w_4^{(u)}$ and $b_4^{-(u)}$. The mean square error (MSE) between the NN output x' and the original data x is thus minimized. The NLPCA is often implemented using the hyperbolic tangent function for f_1 and f_3 and the identity function for f_2 and f_4, so that

$$u = w_2^{(x)}.h^{(x)} + b_2^{-(x)}. \quad \textbf{(2.14)}$$

and $x_i' = (w_4^{(u)}h^{(u)} + b_4^{-(u)})_i$. **(2.15)**

The choice of m, the number of hidden neurons in both the encoding and decoding layers is an important parameter. A larger m increases the nonlinear modeling capability of the network but can also lead to an overfitted solution (i.e. the solution which fits the noise in the data). If f_4 is an identity function and $m = 1$, then equation (2.15) implies that all x'_i are linearly related to a single hidden neuron. Hence there can only be a linear relation between the x'_i variables. Thus for nonlinear solutions, we need to look at $m \geq 2$. It is also possible to have more than one neuron at the bottleneck layer. For instance, with two bottleneck neurons, the dimension of original data will be reduced to a 2-D surface rather than a 1-D curve.

2.4 Summary

The neural network method is among the best methods to implement NLPCA due to its capability of accurately approximating any continuous nonlinear function [2]. The NN architecture described in this chapter forms the basis for all the experiments carried out in this thesis for NLPCA. In chapter 3, the experiments using NN based NLPCA are performed with artificially constructed 2 and 3 dimensional data. The results obtained are analyzed and significant outcomes as well as learning are further taken into consideration while performing classification experiments with speech data of higher dimensionality in chapter 4.

CHAPTER III

ALGORITHM VERIFICATION FOR NEURAL NETWORK BASED NONLINEAR PRINCIPAL COMPONENTS ANALYSIS

This chapter presents the specific algorithm used for dimensionality reduction using neural network based nonlinear PCA. In this chapter, the verification of this algorithm is done by applying it to 2 and 3-D data. The basic steps in the algorithm are:

1. Set up identical input and target values, keeping in mind that the neural network will be trained as an identity network.
2. Establish a bottleneck neural network architecture with the number of bottleneck neurons as the number of reduced dimensions. The selection of non linear transfer functions for the nodes at each layer is very important. Also carefully choose the number of neurons in each hidden layer. Networks are sensitive to the number of neurons in their hidden layers. Too few neurons can lead to underfitting. Too many neurons can contribute to overfitting, in which all training points are well fit, but the fitting curve takes wild oscillations between these points. There is no direct way to know the ideal number of neurons in the hidden layer, but this can be done by a trial and error method. An appropriate initial guess is to choose any number between half and twice the number of the dimension of the data and see how well the network performs.
3. Train the neural network with input data.
4. Extract the weight and bias matrix from the network.

5. Transform the data with the network trained as described in step 1-4, and extract the outputs of the network at the bottleneck. This will be the data with reduced dimensionality.

This algorithm is illustrated in much more detail by describing some experiments conducted with pseudo-random data created so that all the data points lie on a curved surface in the space. The 2-D data created follows a curve similar to the parabola as shown in Fig.3.1. Linear PCA will find a straight-line similar to Fig.1.4 which is a poor approximation of the original data. NLPCA is expected to find an approximation of the data which adequately describes the information contained in the original data as a curve passing through the original data. For the 3-D case, we created random numbers which lie on a Gaussian surface. The output of a bottleneck neural network with 2 nodes at the bottleneck is expected to closely match the input data.

All the experiments were carried out in Matlab 7.1 using the neural network toolbox. Due to random initialization of network parameters in every individual case, about 3-5 runs of the Matlab program were made in all the experiments carried out with 2 and 3-dimensional data to ensure the reliability of the results. For all 16 experiments performed, the numbers of neurons in the hidden nodes for the NN architecture in each experiment were increased gradually in order to evaluate the effect of the number of neurons on the performance of NN to arrive at a correct solution. The same data was used for test and training.

3.1 Experiments with 2-D data

For a simple case, an artificial 2-D input data was created with 3000 data points using the Matlab random number generator. The data was normalized and constrained to follow a curved pattern as shown in Fig.3.2.

1. The input and target data are as shown in Fig.3.2.
2. The neural network consists of 3 hidden layers. The first hidden layer consists of 5 neurons. The middle hidden layer is the bottleneck layer containing only one neuron, thus reducing the dimension of 2-D data to 1 dimension. The third layer again consists of 5 neurons. Finally, the output layer transforms the data back to 2 dimensions. The hyperbolic tangent sigmoid transfer function is used for transforming the inputs to the first hidden layer while a linear transfer function is used for the transformation of variables from the first hidden layer to the bottleneck layer. Similarly the hyperbolic tangent sigmoid transfer function and linear transfer functions are used for transforming the bottleneck layer to the third hidden layer and the third hidden layer to the output layer respectively. This neural network architecture is the one recommended in “Nonlinear Multivariate and Time Series Analysis by Neural Networks Method” by William W. Hsieh, University of British Columbia, Canada [1], and as shown in Fig.2.2 in chapter 2. The node transfer functions are illustrated in fig.3.1.

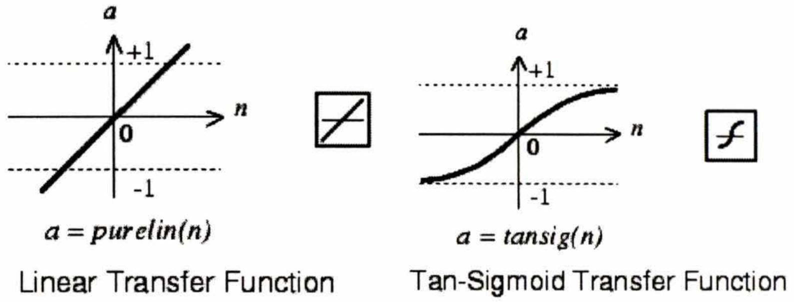

Fig. 3.1. Transfer functions used in neural network architecture.

3.2 Training method

For the purpose of NLPCA, the training method is a critical criterion. Both according to [1] and from our own pilot experiments, typical backpropagation training often results in an ill formed solution for NLPCA. For example, in some pilot testing with backpropagation training, the resulting solution from the neural network was a straight line fit to data, rather than the curve that should have resulted. [1] recommends training with regularization. In our experiments, the method with automatic regularization, as implemented in Matlab7.1 (trainbr) was employed [21]. This method finds optimal parameters for regularization. The typical performance function that is used for training feed forward neural networks is the mean sum of squares of the network errors:

$$F = mse = 1/N\sum_{i=1}^{N}(e_i)^2 = 1/N\sum_{i=1}^{N}(t_i - a_i)^2 \ , \qquad \textbf{(3.1)}$$

where t_i represents the target and a_i represents the actual output for data point *i*. It is possible to improve generalization if we modify the performance function by adding a term that consists of the mean of the sum of squares of the network weights and biases

$$msereg = \gamma mse + (1-\gamma)msw \ , \qquad \textbf{(3.2)}$$

where, γ is the performance ratio and

$$msw = 1/n\sum_{j=1}^{n} w_j^2 \ . \qquad \textbf{(3.3)}$$

Using this performance function will cause the network to have smaller weights and biases, and this will force the network response to be smoother and less likely to over fit. The weights and biases of the network are assumed to be random variables with specified distributions. The regularization parameters are related to the unknown variances associated with these distributions. We can then estimate these parameters using statistical techniques.

Both the original data, and the data after passing through the trained neural network, are shown in Fig.3.2. As anticipated, the neural network transformed data lies on a curved line, which appears to be a good curved-line fit to the original data. The neural network output data was expected to form a line, since the bottleneck layer reduced the data to one dimension. Note, that when the network was trained with standard backpropagation without regularization, the network output data typically formed a straight line, rather than the curved line as shown in Fig.1.4.

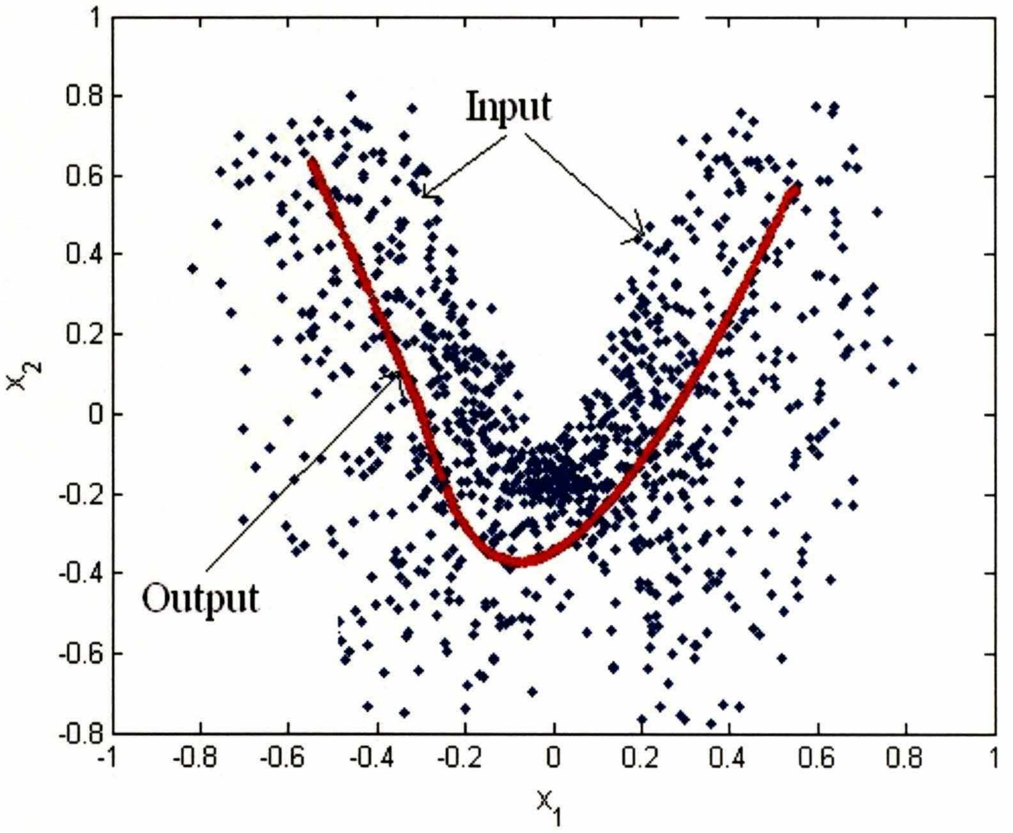

Fig. 3.2. Plot of input and output data for an artificially created 2-D data. The output data is a plot of reconstructed data obtained after passing the input data through the trained neural network.

3.3 Experiments with 3-D data

The second experiment was carried out with 3-D data. A Gaussian function was used to create a nonlinear surface. The 3-D data was created using the Matlab7.1 random number generator, configured so as to create random points on the Gaussian surface. This data is thus nonlinear in nature as it follows a curve which is ideal for testing NLPCA performance. The data thus created was again used for experiments with an identity bottleneck neural network. Several experiments were done with different Gaussian surfaces with degrees of "sharpness" (i.e. with different standard deviations) and various network sizes. The goal was to experimentally investigate the NLPCA process for a variety of conditions. A summary of these experiments is given below. Recall that each neural network was trained so that the output of each network would ideally equal the input data. It is important to find the best network architecture for each case, and especially to determine the sensitivity of the neural network for NLPCA to network parameters. Since the network is initialized randomly for each run of the Matlab code, it is important to run the code several times before reaching a conclusion about performance.

3.3.1 Experiment-1

The input data was created using a Gaussian function with standard deviation of 0.1 and the Matlab random number generator. Data values were constrained to be between -1 and 1. In Fig.3.4, the blue plot shows the input data while the red plot shows the reconstructed output data after passing through the neural network trained for NLPCA.

Table 3.1 shows all the experimental results for these experiments. L-1, L-2, L-3, and L-4 stand for corresponding layers in the network. Fig.3.3 shows the network structure. The learning rate specified for each experiment was 0.1. Experiments were also done with learning rates of 0.05, 0.15, 0.20, and 0.25, but the best results were obtained with a rate of 0.1. The Matlab training function 'trainbr' was used for training which has built-in automatic regularization functionality. 'trainbr' is a network training function that updates the weight and bias values according to Levenberg-Marquardt optimization [21]. It minimizes a combination of squared errors and weights, and then determines the correct combination so as to produce a network that generalizes well. The process is called Bayesian regularization. When using 'trainbr,' it is important to let the algorithm run until the effective number of parameters has converged. The training may stop with the message "Maximum MU reached." This is typical, and is a good indication that the algorithm has truly converged. We can also tell that the algorithm has converged if the sum squared error (SSE) and sum squared weights (SSW) are relatively constant over several iterations.

Termination error is the value of the error at the time of termination or convergence of the program. The minimum value for error below which the program would terminate was set as 1.0 e^{-5}. Epoch is the total number of iterations for which the program is allowed to run before termination. The program terminates either on reaching the Epoch limit or the set termination error is reached.

In each of the 16 cases specified in Table 3.1, the data is reduced to 2-dimensions through the bottleneck layer. In all cases, the algorithm converges and the output plot is similar to the one shown in Fig.3.4.

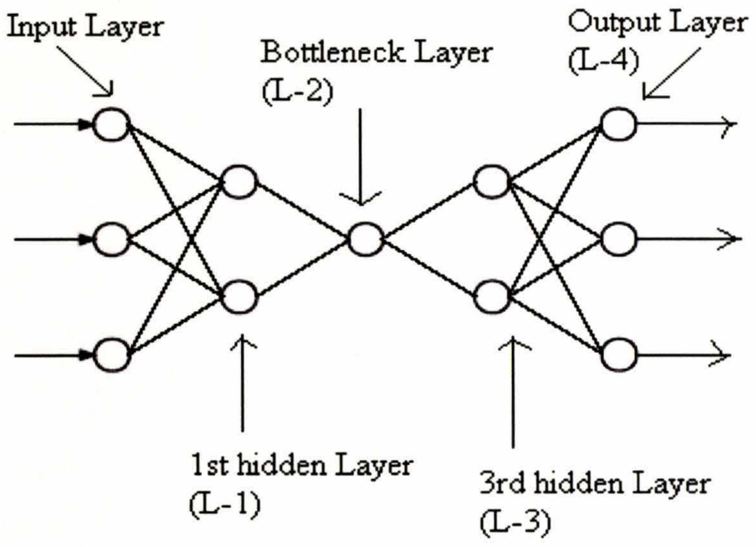

Fig. 3.3. The neural network structure for the experiments.

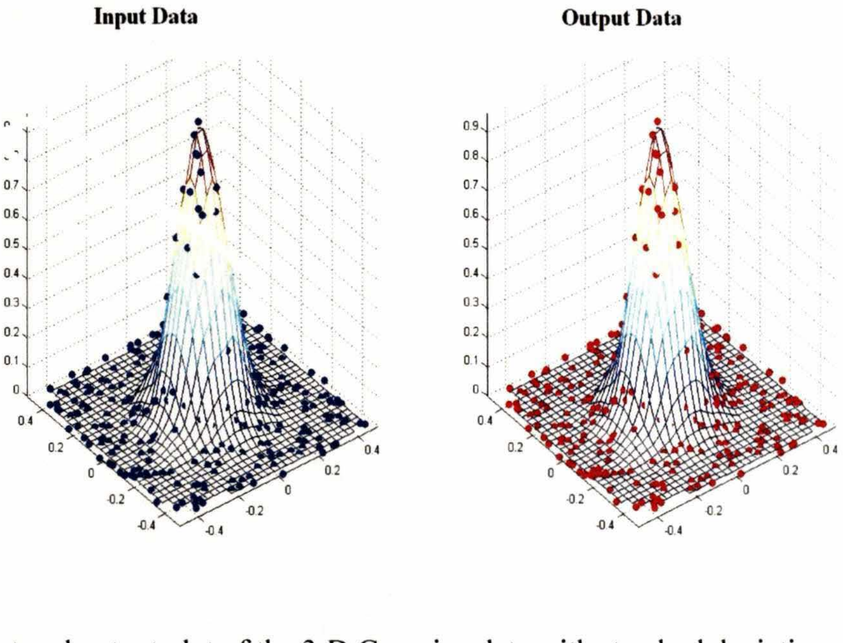

Fig. 3.4. Input and output plot of the 3-D Gaussian data with standard deviation of 0.1 before and after using neural network for NLPCA.

TABLE 3.1

SUMMARY OF EXPERIMENTS WITH 3-D GAUSSIAN SURFACE WITH STANDARD DEVIATION OF 0.5.

Sl. No.	Network architecture(No. of neurons in each layer)				Learning rate	Training function	Termination error (set=e-5)	Epochs (Actual/Set)
	L-1	L-2	L-3	L-4 (O/P)				
1	3	2	3	3	0.1	trainbr	2.575	3000/3000
2	3	2	5	3	0.1	trainbr	1.57	444/3000
3	3	2	10	3	0.1	trainbr	0.041	515/3000
4	3	2	25	3	0.1	trainbr	0.000107	3593/10000
5	5	2	3	3	0.1	trainbr	2.87	10000/10000
6	5	2	5	3	0.1	trainbr	0.5059	885/10000
7	5	2	10	3	0.1	trainbr	0.117	901/10000
8	5	2	25	3	0.1	trainbr	0.02324	1875/10000
9	10	2	3	3	0.1	trainbr	2.16	10000/10000
10	10	2	5	3	0.1	trainbr	0.263	2037/10000
11	10	2	10	3	0.1	trainbr	0.11	2975/10000
12	10	2	25	3	0.1	trainbr	0.002	5384/10000
13	25	2	3	3	0.1	trainbr	0.6	15000/15000
14	25	2	5	3	0.1	trainbr	0.26	3863/15000
15	25	2	10	3	0.1	trainbr	0.02	2748/15000
16	25	2	25	3	0.1	trainbr	goal met	4886/15000

3.3.2 Experiment-2

In this experiment, the Gaussian surface was created using the Matlab random number generator, with points lying on the Gaussian surface with a standard deviation of 0.2. Fig.3.5 shows the input and output data for the neural network for NLPCA. Table 3.2 presents the summary of experiments carried out with this data. In all the cases, the data was reduced to 2-dimensions. The program converged in each case with most of the cases showing the output data to be the same as the input data. However, generally the degree of matching of input and output data is better when more neurons are used at the first and third hidden layer. The Gaussian surface in this experiment is smoother than for the previous experiment. It is hypothesized that this smoother (less "nonlinear") surface was the reason that the network generally converged to a solution more rapidly, as can be seen by comparing the number of iterations (Epoch) at which both experiments converged.

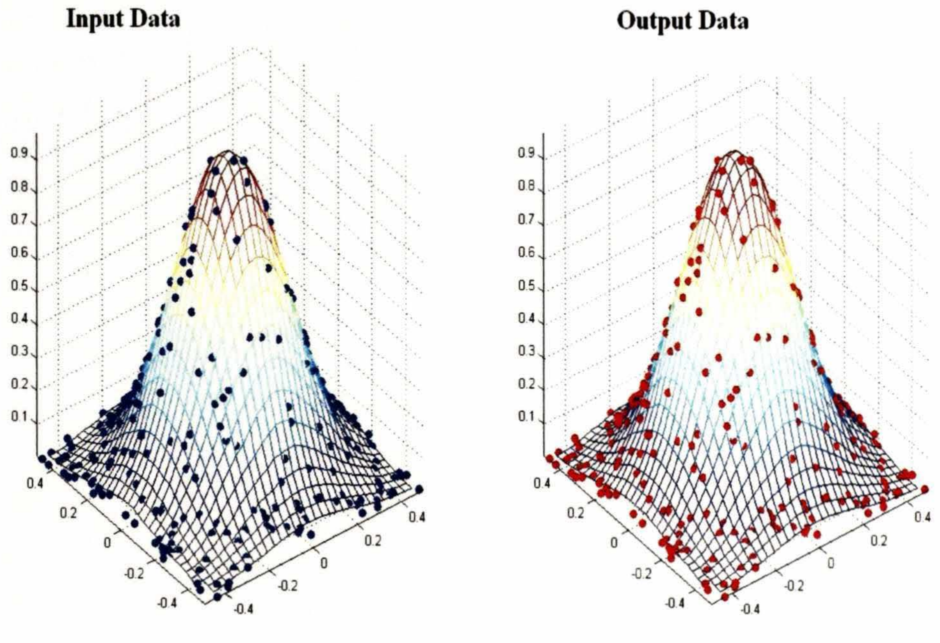

Fig. 3.5. Input and output plot of the 3-D Gaussian data with standard deviation of 0.2 before and after using neural network for NLPCA.

TABLE 3.2

SUMMARY OF EXPERIMENTS WITH 3-D GAUSSIAN DATA WITH STANDARD DEVIATION OF 0.2.

Sl. No.	Network architecture(No. of neurons in each layer)				Learning rate	Training function	Termination error (set=e-5)	Epochs (Actual/Set)
	L-1	L-2	L-3	L-4 (O/P)				
1	3	2	3	3	0.1	trainbr	4.42	3807/15000
2	3	2	5	3	0.1	trainbr	0.42	192/15000
3	3	2	10	3	0.1	trainbr	0.003	1027/15000
4	3	2	25	3	0.1	trainbr	0.21	981/15000
5	5	2	3	3	0.1	trainbr	0.77	681/15000
6	5	2	5	3	0.1	trainbr	0.15	763/15000
7	5	2	10	3	0.1	trainbr	goal met	1026/15000
8	5	2	25	3	0.1	trainbr	goal met	962/15000
9	10	2	3	3	0.1	trainbr	0.57	2222/15000
10	10	2	5	3	0.1	trainbr	0.083	1125/15000
11	10	2	10	3	0.1	trainbr	0.0012	2003/15000
12	10	2	25	3	0.1	trainbr	0.042	1270/15000
13	25	2	3	3	0.1	trainbr	0.48	10738/15000
14	25	2	5	3	0.1	trainbr	0.52	2814/15000
15	25	2	10	3	0.1	trainbr	0.264	15000/15000
16	25	2	25	3	0.1	trainbr	goal met	765/15000

3.3.3 Experiment-3

In this experiment the sharpness of the Gaussian data was decreased by increasing the standard deviation of the Gaussian surface to 0.5. The motive behind carrying out experiments at different sharpness levels was to test the algorithm for different degrees of nonlinearities. Fig.3.6 shows input and output data for the neural network for NLPCA. Table 3.3 presents a summary of all experiments. The network architecture was the same as for earlier cases. The notable point here is that algorithm converges by reaching termination error more times than for the previous experiments, presumably because the data was on a smoother surface.

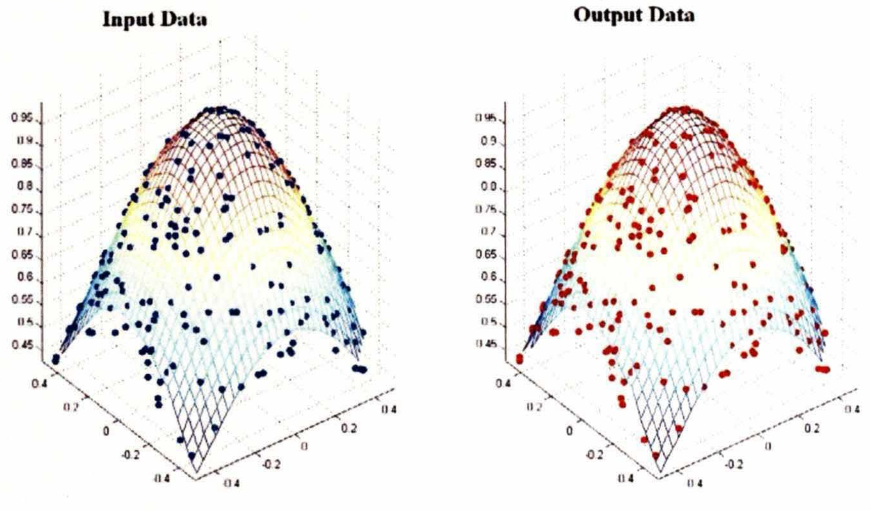

Fig. 3.6. The input and output data of the experiments with 3-D Gaussian data with standard deviation of 0.5.

TABLE 3.3

SUMMARY OF EXPERIMENTS USING 3-D GAUSSIAN DATA WITH STANDARD DEVIATION OF 0.5.

Sl. No.	Network architecture (No. of neurons in each layer)				Learning rate	Training function	Termination error (set=e-5)	Epochs (Actual/Set)
	L-1	L-2	L-3	L-4 (O/P)				
1	3	2	3	3	0.1	trainbr	0.04	15000/15000
2	3	2	5	3	0.1	trainbr	0.001	870/15000
3	3	2	10	3	0.1	trainbr	goal met	663/15000
4	3	2	25	3	0.1	trainbr	goal met	396/15000
5	5	2	3	3	0.1	trainbr	0.03	15000/15000
6	5	2	5	3	0.1	trainbr	0.0005	807/15000
7	5	2	10	3	0.1	trainbr	goal met	730/15000
8	5	2	25	3	0.1	trainbr	goal met	57/15000
9	10	2	3	3	0.1	trainbr	0.03	5973/15000
10	10	2	5	3	0.1	trainbr	0.0002	2537/15000
11	10	2	10	3	0.1	trainbr	goal met	236/15000
12	10	2	25	3	0.1	trainbr	goal met	75/15000
13	25	2	3	3	0.1	trainbr	0.04	15000/1500
14	25	2	5	3	0.1	trainbr	goal met	11845/15000
15	25	2	10	3	0.1	trainbr	goal met	58/15000
16	25	2	25	3	0.1	trainbr	goal met	88/15000

3.3.4 Experiment-4

This section presents the results of the experiment carried out by increasing the sharpness of the 3-D Gaussian data.

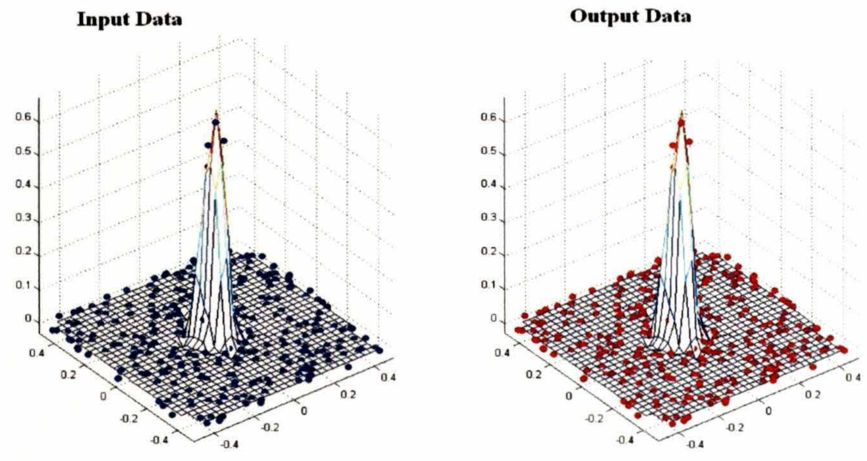

Fig. 3.7. The input and output data of the experiments with 3-D Gaussian data with standard deviation of 0.03.

TABLE 3.4

SUMMARY OF EXPERIMENTS USING 3-D GAUSSIAN DATA WITH STANDARD DEVIATION OF 0.03.

Sl. No.	Network architecture (No. of neurons in each layer)				Learning rate	Training function	Termination error (set=e-5)	Epochs (Actual/Set)
	L-1	L-2	L-3	L-4 (O/P)				
2	3	2	5	3	0.1	trainbr	0.012	1916/15000
3	3	2	10	3	0.1	trainbr	0.01	6454/15000
4	3	2	25	3	0.1	trainbr	0.00019	4768/15000
5	5	2	3	3	0.1	trainbr	1.5	242/15000
6	5	2	5	3	0.1	trainbr	0.88	15000/15000
7	5	2	10	3	0.1	trainbr	0.007	3117/15000
8	5	2	25	3	0.1	trainbr	0.0005	7729/15000
9	10	2	3	3	0.1	trainbr	1.5	2024/15000
10	10	2	5	3	0.1	trainbr	0.049	1100/15000
11	10	2	10	3	0.1	trainbr	0.002	5042/15000
12	10	2	25	3	0.1	trainbr	0.0002	8862/15000
13	25	2	3	3	0.1	trainbr	1.5	932/15000
14	25	2	5	3	0.1	trainbr	0.0512	9792/15000
15	25	2	10	3	0.1	trainbr	0.003	13715/15000
16	25	2	25	3	0.1	trainbr	7.50E-05	15000/15000

In experiment 4, the standard deviation for the Gaussian surface was set to be 0.03. In Fig.3.7, we can see that there is a high degree of nonlinearity present in the data. Table 3.4 lists the results for all experiments carried out with this data. Significantly, the algorithm did not converge to minimum error in any of the cases which is presumably because of the high nonlinearity present in the data. The high nonlinearity also resulted in convergence of algorithm in high number of iterations in approximately each case.

3.4 Discussion and chapter conclusion

In all the experiments carried out with 2 and 3-Dimensional data, approximately 3-5 runs of the Matlab program were made to ensure the reliability of the results. In most of the cases, there was less variability from run to run but in a few cases we found that the output result was not as desired. As an example, one such case is shown in Fig.3.8, where the output data is not the same as the input data. Some of the poor results may be attributed to improper initialization of the network, improper training, and/or poor scaling of input data.

Similar kinds of results as shown in Fig.3.8 were also obtained when the standard backpropagation training was used. One of the problems that can occur during standard backpropagation training is overfitting. The error on the training set is driven to a very small value, but when new data is presented to the network, the error is large. The network has memorized the training examples, but it has not learned to generalize to new situations. One method for improving network generalization is to use a network that is just large enough to provide an adequate fit. Unfortunately, it is difficult to know beforehand how large a network should be for a specific application. Hence, another method for improving generalization that is implemented in the Matlab neural network toolbox, regularization, can be (and was) used.

Since we used the same data for training and testing, this generalization was not the issue for our experiments, but could have been an issue if we had used separate training and test data.

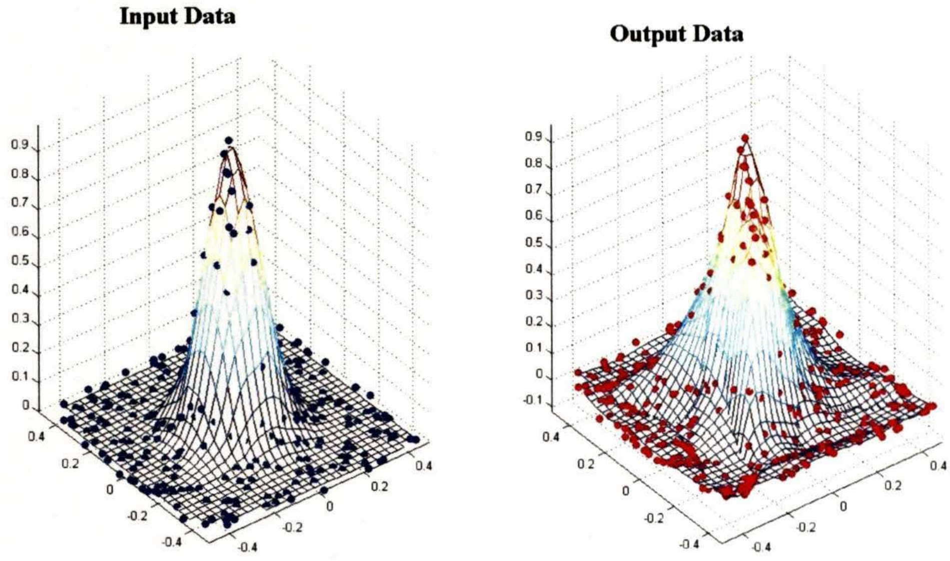

Fig. 3.8. This figure shows a case where output data does not match input data after passing through a neural network trained for NLPCA.

Picking the learning rate for a nonlinear network is another difficulty. As with linear networks, a learning rate that is too large leads to unstable learning. Conversely, a learning rate that is too small results in excessively long training times. Unlike linear networks, there is no easy way of picking a good learning rate for nonlinear multilayer networks.

The error surface of a nonlinear network is more complex than the error surface of a linear network. The problem is that nonlinear transfer functions in multilayer networks introduce many local minima in the error surface [10]. As gradient descent is performed on the error surface it is possible for the network solution to become trapped in one of these local minima. This may happen depending on the initial starting conditions. Settling in a local minimum may be good or bad depending on how close the local minimum is to the global minimum and how low an error is required. In any case, be cautioned that although a multilayer backpropagation network with enough neurons can implement just about any function, backpropagation will not always find the correct weights for the optimum solution. In such cases it is best to reinitialize the network and retrain several times to find the "best" solution.

CHAPTER IV

VOWEL CLASSIFICATION EXPERIMENTS

This chapter presents various experiments with speech data and their results. In chapter 3, we found that NLPCA works well for 2-D and 3-D data, in the sense that 2-D data that lies on a curved line can be well represented by 1 NLPCA dimension, and 3-D data that lies on a curved surface can be well represented by 2 NLPCA dimensions. The natural progression is to evaluate NLPCA for higher dimensions and with "real" data. Here 5-D, 10-D, and 15-D data from 10 vowel sounds extracted from continuous speech sentences were used for experimentation purposes. The basic test consisted of neural network vowel classification for various dimensionality subspaces. Four different methods were used for dimensionality reduction and classification. Both training and testing results are tabulated and compared.

4.1 Methods

Four methods were used for dimensionality reduction, as summarized below.

4.1.1 Original data

For this method, subsets of the original vowel data features, the DCTCs as explained below, were used as the features. The number of DCTCs used ranged from 1 to 15, with the cosine basis vector index used to order the DCTCs.

4.1.2 Linear PCA

For this method, dimensionality reduction was performed using linear PCA.

4.1.3 NLPCA- 1

Both the third and fourth methods are NLPCA based dimensionality reduction using neural networks. In the third method, the input data is passed through a bottleneck neural network trained as an identity map, as described in chapter 3. Data with reduced dimension is extracted from the bottleneck layer and then classified using a neural classifier.

4.1.4 NLPCA- 2

In the fourth method, the input data is used with a bottleneck neural network trained as a classifier. The number of nodes in the bottleneck is considered to be the dimensionality of the NLPCA, since all the data is encoded at this dimensionality as it passes through the network.

The results are tabulated and compared analytically as well as graphically for the various methods, and for various numbers of initial dimensions.

4.2 Experimental set up

This section presents the details of the database used in the experiments and feature extraction parameters. A description of the classifier is also provided, and a flowchart of the overall experimental process is shown.

4.2.1 Database used for experiments

For experimentation with higher dimensional data, the NTIMIT database was used which was collected by transmitting all 6300 original TIMIT database. (The TIMIT database was provided by a joint effort between Texas Instruments and MIT, and was developed primarily to aid acoustic phonetic speech recognition research [22]. It consists of acoustic files from 630 speakers from 8 dialect regions (DR 1~8) with 10 sentences per speaker) utterances through various channels in the NYNEX telephone network and re-digitizing them.

In this chapter, the experiments were performed using the NTMIT data from Dialect region -2 (DR2, Northern Dialect) in which 760 sentences (from 76 speakers) were used for training and 260 sentences (from 26 speakers) for testing. The experiments were specifically based on the 10 vowels 'ue', 'ae', 'ah', 'aw', 'ee', 'eh', 'ih', 'oo', 'uh' and 'ur' (Darpabet codes). In all 1853 tokens were used for training and 629 for testing. These vowels were selected as 30 ms frames centered at the labeled midpoint from the sentences mentioned.

4.2.2. Feature extraction parameters

In all, 15 DCTCs (Discrete Cosine Transform Coefficients) were extracted from waveform file using the ODU Speech lab program 'tfrontm.' A 30 millisecond window for each waveform was used with a frequency range of 300-4000Hz and with a warping factor of 0.45. These 15 DCTCs constitute the 15-dimensional data for our experimentation and were written in ASCII format for processing. The program was set up to create 10 files for training, and 10 files for testing, with each file containing all the tokens for a particular vowel.

4.2.3 Classifier summary

A feedforward NN classifier with 25 hidden nodes was used for the classification. For the first three methods, the architecture for the classifier was 'M-25-10', where 'M' represents number of (reduced) data dimensions, 25 denotes the number of hidden nodes, and 10 is the number of vowels used. The architecture used for NLPCA-2 classifier was 'N-6-M-6-10' for 5-D data, while for 10 and 15-D data the classifier format used was 'N-10-M-10-10', where N denotes the number of original dimensions, and M denotes the number of reduced dimensions. The total number iterations used for both classifiers are 150000.

4.2.4 Process flowchart

The overall experimental set up and processes are shown in the Fig.4.1. In the next section, the experiments are presented and results are analyzed.

4.3 Experiments with 5-D speech data

In this section, the details of all experiments carried out with 5-dimensional speech data are presented. These 5 dimensions (or features) are the first 5 features from the original 15 dimensional data. All the experiments were performed with 10 vowels.

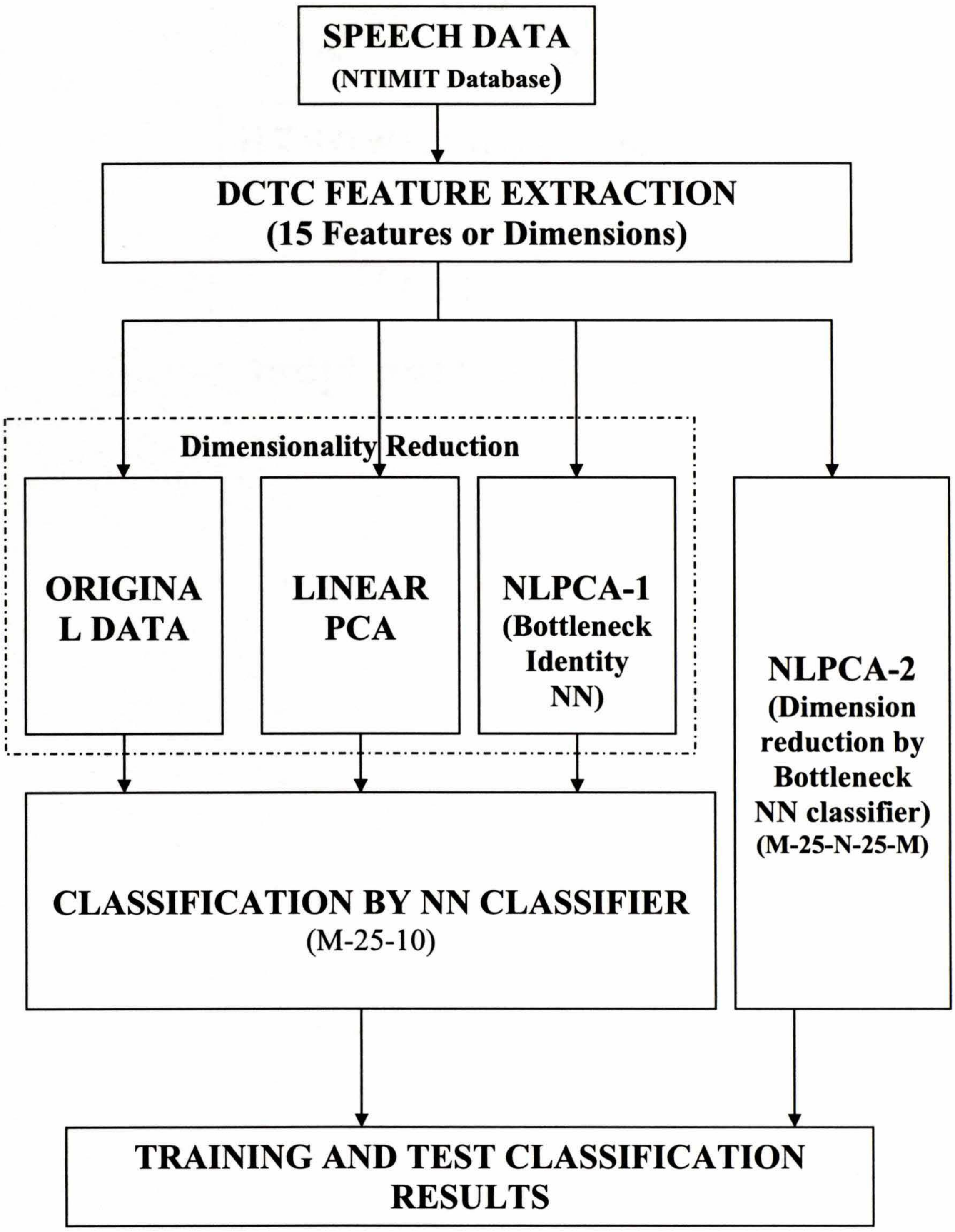
SPEECH DATA
(NTIMIT Database)
DCTC FEATURE EXTRACTION
(15 Features or Dimensions)
Dimensionality Reduction
ORIGINA L DATA
LINEAR PCA
NLPCA-1 (Bottleneck Identity NN)
NLPCA-2 (Dimension reduction by Bottleneck NN classifier) (M-25-N-25-M)
CLASSIFICATION BY NN CLASSIFIER (M-25-10)
TRAINING AND TEST CLASSIFICATION RESULTS

Fig.4.1. The flowchart for experiments in section 4.3.

4.3.1 Experiment-1: Classification with 5-D original data

Here, the original features were classified using the neural network classifier. In Table 4.1, the training and testing results of classification are presented. Classification improves as the dimensionality of data increases.

TABLE 4.1

CLASSIFICATION RESULTS WITH 5-D ORIGINAL DATA.

Number of features	Training results (%)	Testing results (%)
1	30.42	30.45
2	35.84	36.63
3	52.63	51.39
4	59.93	58.67
5	64.45	59.98

4.3.2 Experiment -2: Dimensionality reduction of 5-D data using linear PCA

Here the original features were scaled and eigenvectors were computed. Features were then transformed to the new space using the PCA eigenvectors. Classification results for different dimension reduction values are shown in Table 4.2.

TABLE 4.2

CLASSIFICATION RESULTS FOR 5-D DATA WITH LINEAR PCA.

Dimension reduction (Initial-Final)	Training results (%)	Testing results (%)
5-1	41.28	43.03
5-2	50.51	48.06
5-3	55.62	52.65
5-4	57.24	55.82
5-5	64.07	60.42

4.3.3 Experiment-3: Dimensionality reduction of 5- D data using NLPCA-1

This experiment was performed using the Matlab 7.1 neural network toolbox for the dimensionality reduction. The same network parameters were selected as for the 3-D data example in chapter 3 but using network sizes as listed in Table 4.3. The training was performed using regularization. The hyperbolic tangent function was selected for the 1st and 3rd layers, while a linear function was chosen for the 2nd and 4th layers. The classification results are shown in Table 4.3.

TABLE 4.3

CLASSIFICATION RESULTS FOR 5-D DATA WITH NLPCA-1.

Dimension reduction (Initial-Final)	Neural network architecture	Training results (%)	Testing results (%)
5-1	5-10-1-10-5	40.93	39.42
5-2	5-10-2-10-5	44.89	40.95
5-3	5-10-3-10-5	56.21	52.1
5-4	5-10-4-10-5	62.1	59.1
5-5	5-10-5-10-5	61.34	59.32

4.3.4 Experiment-4: Dimensionality reduction of 5-D speech data using NLPCA-2

In this experiment, the neural network classifier was again used to reduce the dimensionality, but the same neural network was trained as a classifier. The vowel data was used as the input to a network and was classified at the output layer. The classification results are shown in Table 4.4.

TABLE 4.4

CLASSIFICATION RESULTS FOR 5-D DATA WITH NLPCA-2.

Dimension reduction (Initial-Final)	Neural network architecture	Training results (%)	Testing results (%)
5-1	5-6-1-6-5	49.37	47.13
5-2	5-6-2-6-5	57.66s	54.84
5-3	5-6-3-6-5	60.08	57.03
5-4	5-6-4-6-5	59.02	58.15
5-5	5-6-5-6-5	57.85	59.55

4.3.5 Comparison of the results obtained with 5-D data

Fig.4.2 shows the plot of classification of training results for 5-D data obtained with different methods, while Fig.4.3 shows the test results. Recall that NLPCA-1 represents the method in which the neural network with a bottleneck layer is used for dimensionality reduction, and another neural network is trained as a classifier using the reduced dimensionality data. NLPCA-2 represents the method whereby one neural network is used for both dimensionality reduction and classification. In comparing the results, it can be seen that the training results are higher than the testing results for all cases, which is expected. However in both cases, the NLPCA-2 works better for low dimensionality spaces (i.e. three or fewer dimensions), and there is not much difference between all four methods for four or more dimensions.

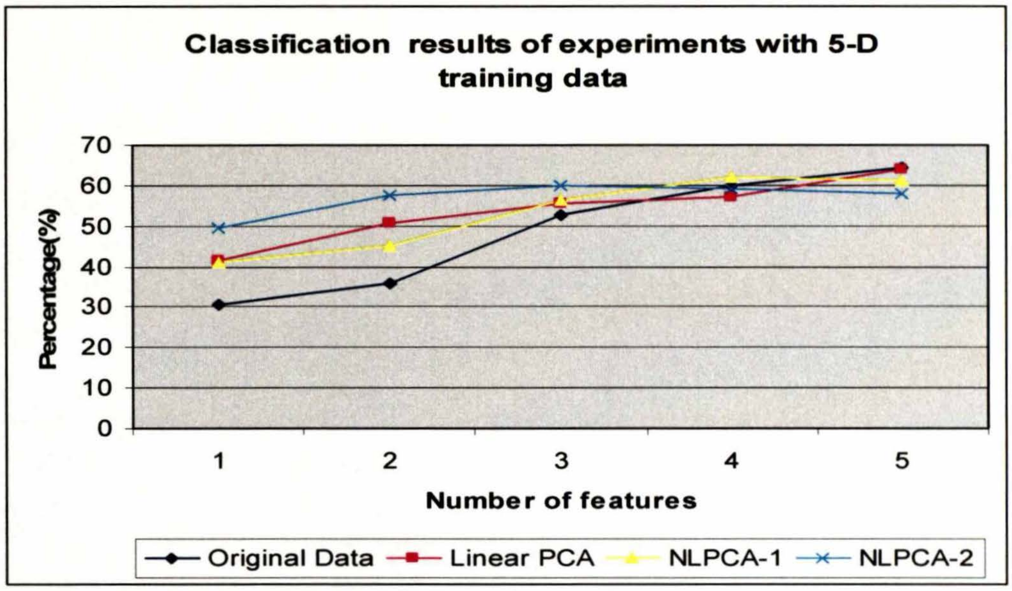

Fig.4.2. Comparison of classification results obtained with 5-D training data.

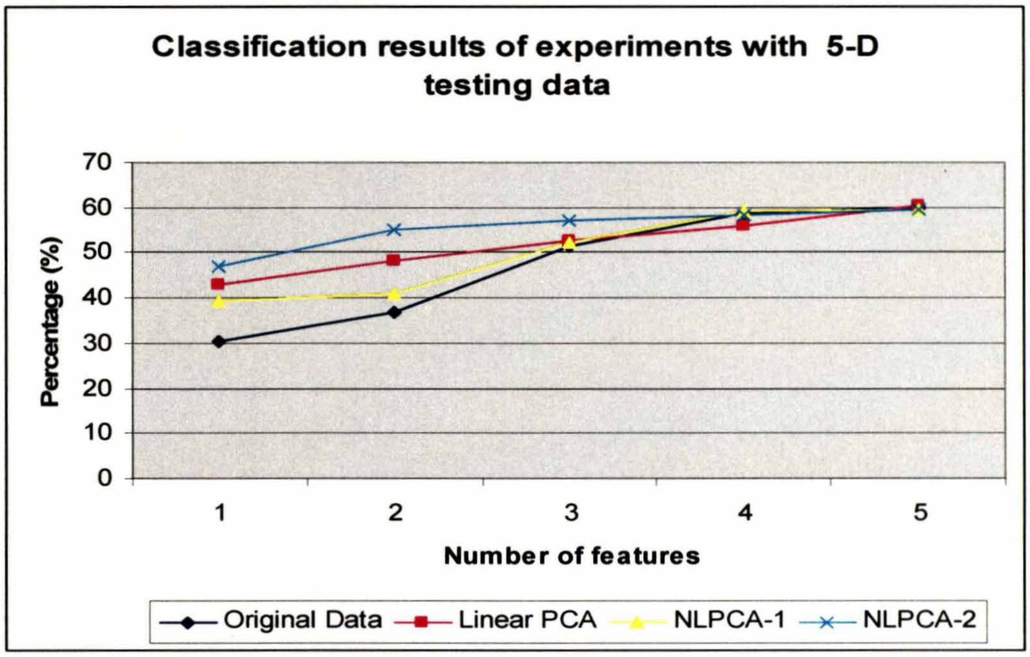

Fig.4.3. Comparison of classification results obtained with 5-D testing data.

4.4 Experiments based on 10-D speech data

In this section the experiments performed with 10 dimensional data are presented. All the parameters remain the same as for the case of 5-D data.

4.4.1 Experiment-5: Classification with 10-D original data

The results, as the numbers of features are varied from 1 to 10, are presented in Table 4.5. Note that the results based on 5 or fewer features are identical to those presented in Table 4.1.

TABLE 4.5

CLASSIFICATION RESULTS WITH ORIGINAL DATA.

Number of features	Training results (%)	Testing results (%)
1	30.42	30.45
2	35.84	36.63
3	52.63	51.39
4	59.93	58.67
5	64.45	59.98
6	65.27	61.13
7	66.18	61.67
8	66.79	63.48
9	67.94	62.99
10	68.05	63.42

4.4.2 Experiment -6: Dimensionality reduction of 10-D using linear PCA

Table 4.6 presents the classification results obtained with reduced dimensionality using the linear PCA method.

TABLE 4.6

CLASSIFICATION RESULTS FOR 10-D DATA WITH LINEAR PCA.

Dimension reduction (Initial-Final)	Training results (%)	Testing results (%)
10-1	42.58	41.06
10-2	49.9	47.27
10-3	54.38	51.34
10-4	58.06	56.26
10-5	60.48	56.37
10-6	63.17	59.7
10-7	64.01	60.52
10-8	66.13	63.09
10-9	67.59	63.48
10-10	68.32	63.84

4.4.3 Experiment -7: Dimensionality reduction of 10-D data using NLPCA-1

Table 4.7 shows the classification results obtained for 10-D training and testing data.

TABLE 4.7

CLASSIFICATION RESULTS FOR 10-D DATA WITH NLPCA-1.

Dimension reduction (Initial-Final)	Neural network architecture	Training results (%)	Testing results (%)
10-1	10-25-1-25-10	30.8	30.95
10-2	10-25-2-25-10	30.58	30.13
10-3	10-25-3-25-10	55.05	53.14
10-4	10-25-4-25-10	58.33	54.51
10-5	10-25-5-25-10	64.77	61.34
10-6	10-25-6-25-10	63.17	59.32
10-7	10-25-7-25-10	63.25	57.03
10-8	10-25-8-25-10	66.64	60.96
10-9	10-25-9-25-10	69.45	62.44
10-10	10-25-10-25-10	63.02	62.49

4.4.4 Experiment-8: Dimensionality reduction of 10-D speech data using NLPCA-2

TABLE 4.8

CLASSIFICATION RESULTS FOR 10-D DATA WITH NLPCA-2.

Dimension reduction (Initial-Final)	Neural network architecture	Training results (%)	Testing results (%)
10-1	10-10-1-10-10	53.96	52.65
10-2	10-10-2-10-10	59.65	55.99
10-3	10-10-3-10-10	60.04	58.61
10-4	10-10-4-10-10	63.06	59.65
10-5	10-10-5-10-10	63.74	61.07
10-6	10-10-6-10-10	62.35	58.88
10-7	10-10-7-10-10	63.97	60.05
10-8	10-10-8-10-10	64.14	60.52
10-9	10-10-9-10-10	63.63	58.56
10-10	10-10-10-10-10	65.19	60.62

4.4.5 Comparison of the results

Fig.4.4 and Fig.4.5 presents a graphical comparison of the results obtained with the various methods based on 10 original features. The trend remains the same as for 5-D data. The NLPCA-2 works the best for the low dimensionality spaces (four or fewer dimensions), and all methods perform similarly for five or more dimensions.

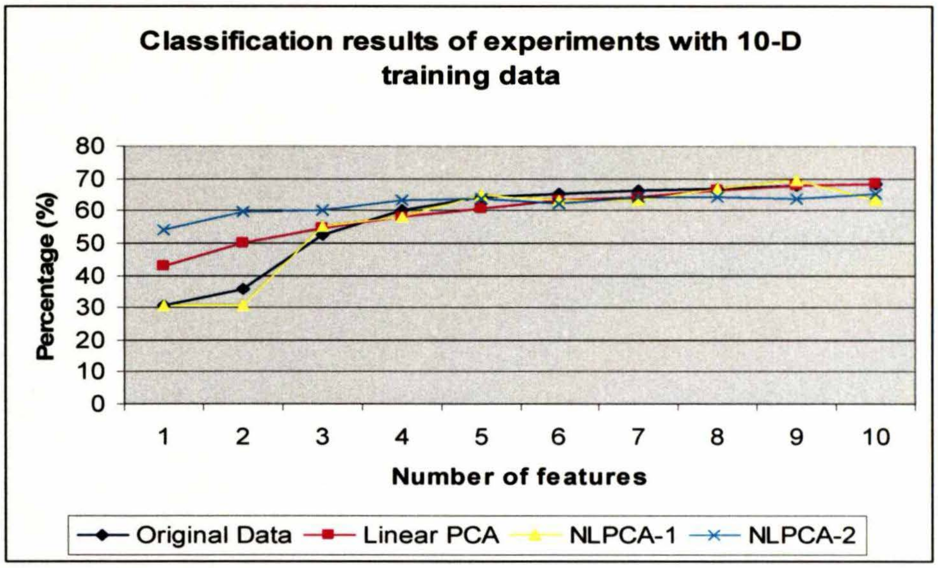

Fig.4.4. Comparison of classification results obtained with 10-D training data.

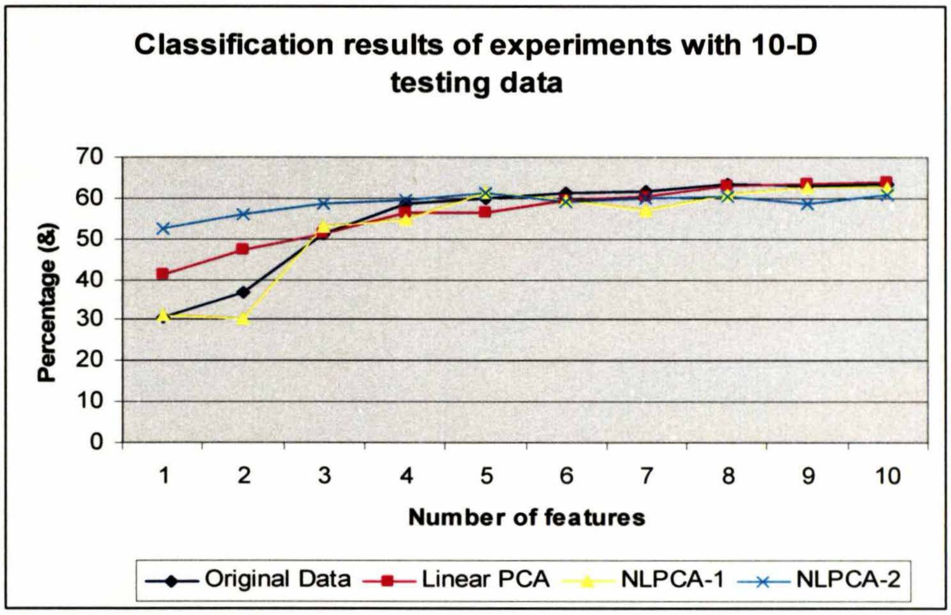

Fig.4.5. Comparison of classification results obtained with 10-D testing data.

4.5 Experiments based on 15-D speech data

This section presents the final experiments performed with 15-D speech data. Here the problem of dimensionality reduction becomes significantly more computationally complex than for 5-D or 10-D data.

4.5.1 Experiment-9: Classification with 15-D original data

Table 4.9 shows the classification results for original 15-D data. This contains the result from the 10-D data, but extended to 15-D.

TABLE 4.9

CLASSIFICATION RESULTS WITH 15-D ORIGINAL DATA.

Number of features	Training results (%)	Testing results (%)
1	30.42	30.45
2	35.84	36.63
3	52.63	51.39
4	59.93	58.67
5	64.45	59.98
6	65.27	61.13
7	66.18	61.67
8	66.79	63.48
9	67.94	62.99
10	68.05	63.42
11	68.85	64.68
12	69.5	65.17
13	69.81	65.17
14	70.26	64.46
15	70.57	65.77

4.5.2 Experiment-10: Dimensionality reduction of 15-D data using linear PCA

Table 4.10 presents the classification results of 15-D data with linear PCA.

TABLE 4.10

CLASSIFICATION RESULTS FOR 15-D DATA WITH LINEAR PCA.

Dimension reduction (Initial-Final)	Training results (%)	Testing results (%)
15-1	41.27	39.42
15-2	46.95	43.74
15-3	50.86	47.9
15-4	56.1	52.27
15-5	57.74	54.24
15-6	61.08	56.91
15-7	62.31	57.96
15-8	63.23	60.03
15-9	65.9	61.24
15-10	67.5	63.26
15-11	68.3	64.08
15-12	68.85	63.42
15-13	69.75	65.17
15-14	70.53	65.17
15-15	70.99	65.06

4.5.3 Experiment-11: Dimensionality reduction of 15-D data with NLPCA-1

Table 4.11 presents the classification results obtained using the NLPCA-1 approach. Here the node nonlinearities and the training method remains the same as for the 5-D and 10-D data. However, the number of neurons in the first and third hidden layers is set to 30, while the desired dimension reduction is represented by the bottleneck layer.

TABLE 4.11

CLASSIFICATION RESULTS FOR 15-D DATA WITH NLPCA -1.

Dimension reduction (Initial-Final)	Neural network architecture	Training results (%)	Testing results (%)
15-1	15-30-1-30-15	31.88	31.15
15-2	15-30-2-30-15	48.04	44.29
15-3	15-30-3-30-15	49.16	45.54
15-4	15-30-4-30-15	56.88	51.65
15-5	15-30-5-30-15	61.8	58.67
15-6	15-30-6-30-15	61.8	56.59
15-7	15-30-7-30-15	64.87	58.06
15-8	15-30-8-30-15	66.49	61.45
15-9	15-30-9-30-15	67.82	62.17
15-10	15-30-10-30-15	68.28	63.09
15-11	15-30-11-30-15	70.09	62.99
15-12	15-30-12-30-15	68.95	60.96
15-13	15-30-13-30-15	70.4	63.2
15-14	15-30-14-30-15	66.36	60.74
15-15	15-30-15-30-15	63.73	63.42

4.5.4 Experiment-12: Dimensionality reduction of 15-D speech data using NLPCA-2

Table 4.12 shows the classification results obtained using NLPCA-2 approach. The numbers of neurons in the hidden layers were selected by empirical methods and the final architecture selected, as mentioned in the table, seemed to perform the best among all the combinations tested.

TABLE 4.12

CLASSIFICATION RESULTS FOR 15-D DATA WITH NLPCA-2.

Dimension reduction (Initial-Final)	Neural network architecture	Training results (%)	Testing results (%)
15-1	15-10-1-10-10	59.38	55.88
15-2	15-10-2-10-10	66.2	60.2
15-3	15-10-3-10-10	66.55	62.49
15-4	15-10-4-10-10	67.21	62.49
15-5	15-10-5-10-10	67.58	62.49
15-6	15-10-6-10-10	68.26	62.55
15-7	15-10-7-10-10	68.53	62
15-8	15-10-8-10-10	68.38	63
15-9	15-10-9-10-10	68.85	63.26
15-10	15-10-10-10-10	67.86	62.49
15-11	15-10-11-10-10	69.94	63.31
15-12	15-10-12-10-10	68.7	62.17
15-13	15-10-13-10-10	68.87	63
15-14	15-10-14-10-10	69.5	63.26
15-15	15-10-15-10-10	71.58	61.73

4.5.5 Comparison of the results

Here, the graphical comparison of the results obtained in these sections is presented for both training and test results. Here also, as we have seen in previous experiments with 5-D and 10-D data, NLPCA-2 seems to perform the best particularly for lower dimensionality spaces. Fig.4.6 and Fig.4.7 shows performance plots for training and testing data respectively.

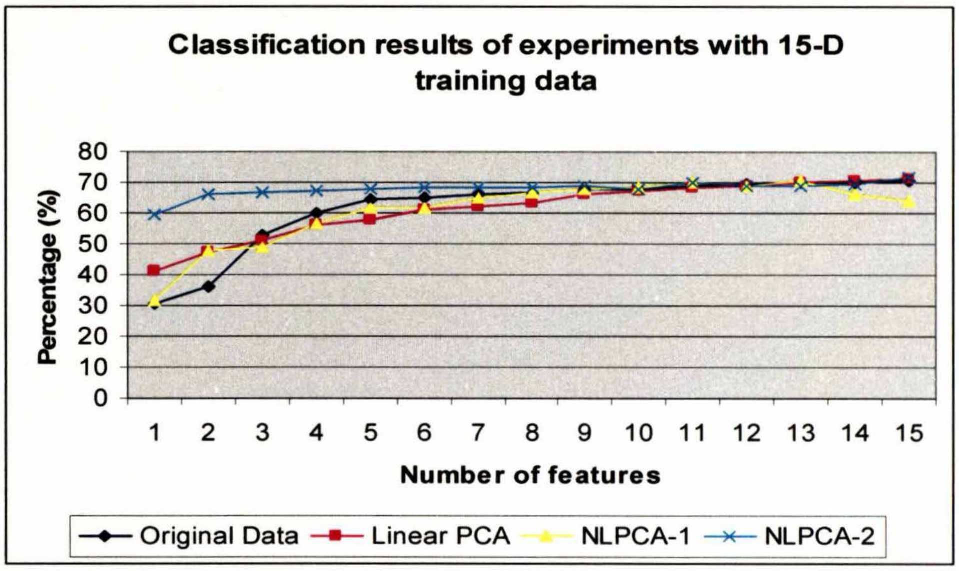

Fig.4.6. Comparison of classification results obtained with 15-D training data.

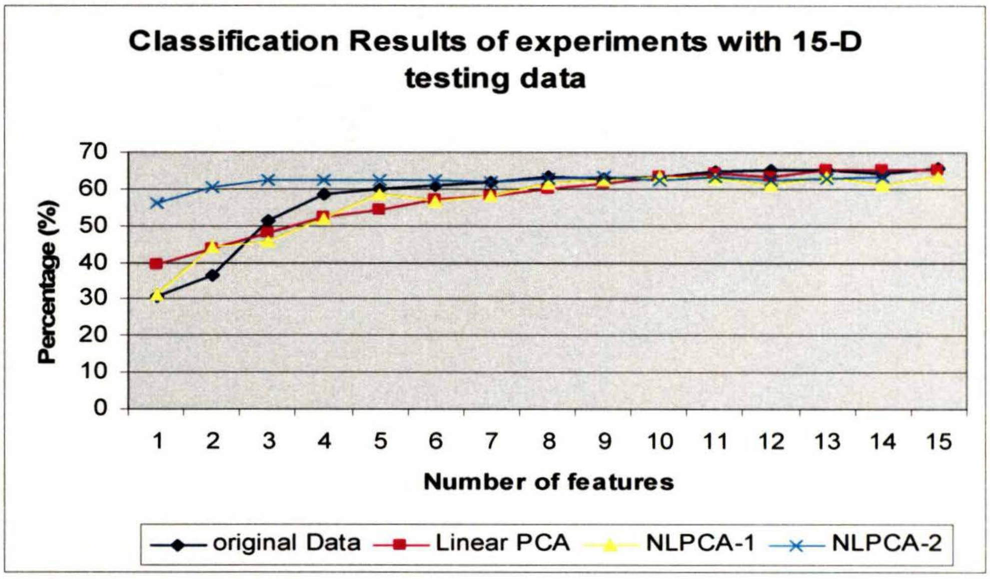

Fig.4.7. Comparison of classification results obtained with 15-D testing data.

4.6 Summary

In this chapter, the experiments with real speech data with 5, 10, and 15 original dimensions were conducted. In nearly all the experiments, the NLPCA-2 approach works best for the lower dimensionality spaces (three or fewer), but it does not result in higher accuracy when a large number of dimensions (six or more) is used. Linear PCA fairs poorly as compared to NLPCA-2 but slightly better than the other two methods at lower dimensionalities. However all the methods perform approximately the same when used for higher dimensionality data. In the next chapter, the results are discussed in more detail and the scope for future work is presented.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

In this thesis, linear and nonlinear approaches to dimensionality reduction were presented and their performance compared. This work was motivated by the concern that high-dimensional data presents many challenges with regard to interpretation, computational complexity, and analysis. In the field of pattern classification, such as automatic speech recognition, high dimensionality spaces cause 'curse of dimensionality' problems with classifier training and poor generalization. A good dimensionality reduction technique would greatly help to overcome these challenges. The main aim of dimensionality reduction technique is to reduce the dimensionality without loss of important information. Linear PCA is a commonly used and simple technique for data reduction, but, while it works well for some kinds of data, it does not perform desirably when the data follows curved subspaces. Linear methods currently used are often inadequate to represent complex real world systems, which often have data better suited to representation by nonlinear basis vectors.

Some of the limitations of linear dimensionality reduction are expected to be removed using nonlinear methods. Hence the concept of Nonlinear PCA was introduced. The artificial neural network was used as the implementation of NLPCA for data in a nonlinear system. Initial experiments were performed using pseudo-random 2-D and 3-D data and then extended to high-dimensional real time speech data. For the case of the speech data, classification experiments were performed with reduced dimensionality data and classification performance was used to judge the effectiveness of each method of data reduction. NLPCA was first tested with 2-D and 3-D data, created so that all data lies on a nonlinear curve or nonlinear surface. A four layer (3 hidden layers) network was trained as an identity map. This network was configured as a "bottleneck" to force dimensionality reduction. After training, the data was passed through the network and reconstructed and plotted. For the case of 2-D data, the data at the output of the neural network was mapped to a single curved line passing through the original scattered data points. For the case of 3-D data, with a bottleneck of 2 dimensions, the output data was almost identical to the original. From this, we concluded that neural network-based NLPCA can be used to "discover" curved basis vectors suitable for representing data located on curved subspaces in 2-D and 3-D spaces. No linear set of basis vectors, such as those that might be obtained by linear PCA methods, could have been used to represent this data accurately in reduced dimensionality spaces. Therefore, we concluded that there are indeed cases for which NLPCA is advantageous over linear PCA, and we anticipated this advantage would extend to higher dimensions and to real data.

For higher dimensional testing, we used speech data for 10 vowels extracted from the NTIMIT database, with 5, 10, and 15 features. For 5-D to 1-D dimension reduction, we obtained approximately 30% classification accuracy when the original data (using the first dimension only) was classified, while with linear PCA, the corresponding accuracy improves to approximately 40% for both training and testing data. Results with NLPCA-1 (using a bottleneck neural network for dimensionality reduction and a separate neural network for classification) are approximately the same as for linear PCA. Using NLPCA-2, which uses one neural network for dimensionality reduction and classification, the NLPCA results are considerably improved to 49% for training data and 47% for testing data, respectively. As dimensionality increases, the results generally improve for all methods but the degree of improvement for NLPCA is less than for other methods. For example, for 5D-3D dimension reduction, the results are improved from 40% to 58% for linear PCA, but results for NLPCA improved from 47% to 59%. For 10-D and 15-D data the trend remains the same as for 5-D data. However, the important thing to note here is that for a large number of dimensions, the classification results are nearly the same for all the methods used. Thus NLPCA methods show potential for classification improvement in very low dimensionality subspaces (five or fewer dimensions for the cases examined in this study). For larger dimensionality subspaces, there is no significant advantage of NLPCA over linear PCA.

One main disadvantage of Neural Network methods is instability or nonuniqueness. For the NN approach, optimization which starts with different initial parameters will typically result with different minima. A number of optimization runs starting from several random initial parameters is needed, and the best run is chosen as solution. While performing several optimization runs, there is no certainty that a global minimum (of an error function) is found [1]. Proper scaling of the data is essential to avoid the algorithm searching for parameters with a wide range of magnitudes. However, training the network with regularization generally improves the stability. For the case of short data with noise, nonlinear methods may not be able to find a reliable solution. The advantage of nonlinear methods over linear methods also depends upon the data set. Nonlinear approach works well in the case of data with little noise but is generally ineffective if data is short and noisy and properties are generally linear in nature. There might be other reasons for the limited performance of the NLPCA. One of the reasons might be poor training of the network.

One of the most interesting results of the work done in this thesis is that the first approach to NLPCA, NLPCA-1 based on the bottleneck neural network trained as an identify map was in fact generally no better (and sometimes worse) than linear PCA for dimensionality reduction followed by classification. However, the second NLPCA method, NLPCA-2, whereby the dimensionality reduction and classification were incorporated in a single network, was generally superior in performance to either linear or PCA or NLPCA-1. One possible explanation for this is that both PCA and NLPCA-1 implicitly assume that mean square error is a good measure of distance for making category decisions, whereas no such assumption is made for NLPCA-2. Rather, for NLPCA-2, the criteria for reducing dimensions are directly linked to minimizing classification error.

The natural extension of this thesis would be to develop a suitable method for proper training of the network. Presently the numbers of hidden neurons in the NN and weight parameters are determined by a trial and error approach. There are certain techniques such as generalized cross validation [13] and information criterion [16], which may be of great help in the future to provide more guidance in order to choose the best architecture for neural networks. Another drawback may be use of appropriate nonlinear transfer function. Apart from hyperbolic tangent transfer function which was used for the experiments in this thesis, there might be some transfer functions which may work better for certain types of data sets. While the Neural Network method is currently widely used for nonlinear analysis, new emerging techniques such as Kernel based methods [4] may play a significant role for the future development.

In terms of using NLPCA for automatic speech recognition applications, this technique could be used to reduce dimensionality, and then the reduced dimensionality space could be used as the features for an HMM-based speech recognition system. Presumably, the NLPCA networks could be trained with a relatively small database and then the trained network could be applied to other data, which is used for training an HMM system. However, this use of dimensionality reduction remains to be investigated. For the small databases used in this study, and the relatively limited classification experiments (vowel classification with labeled data), highest classification accuracy was obtained using all the original dimensions.

REFERENCES

[1] William W. Hsieh, "Nonlinear Multivariate and Time Series Analysis by Neural Network Methods," Review of Geophysics, 42, RG1003, Paper number- 2002RG000112, March 2004.

[2] Dong d. and Thomas J. McAvoy, "Nonlinear Principal Component Analysis-Based on Principal Curves and Neural Networks," Proceedings of the American Control Conference, Baltimore, Maryland, pp. 1284-1288, June 1994.

[3] E.B.Martin, A.J.Morris, and J.Zhang, "Process Performance Monitoring using Multivariate Statistical Process Control," IEE proceedings, Control theory applications, Vol. 143, No.2, pp.132-144, March 1996.

[4] Fernando Perez-Cruz, and Olivier Bousquet, "Kernel Methods and their Potential use in Signal Processing," IEEE signal processing magazine, Volume 21, Number 3, pp. 57-64, May 2004.

[5] David L. Donoho, "High-Dimensional Data Analysis: The Curses and Blessings of Dimensionality," Lecture presented in Department of Statistics, Stanford University, Stanford, http://www-stat.stanford.edu/~donoho/Lectures/AMS2000/Curses.pdf, August 8, 2000.

[6] Leonard Chapel, "Dimension Reduction by Non Linear Methods," Guest lecture, University of North Carolina, Chapel Hill, http://www.cs.unc.edu/Courses/comp290-90-f03/DimReduction2.pdf, Fall 2003 (last accessed in July 2006).

[7] Website Link, "Principal Component Analysis and Its Extensions (Y.M., S.S)," Chapter 2, decision.csl.uiuc.edu/ ~yima/psfile/ECE598/GPCA-notes.pdf. (Last accessed in July 2006).

[8] Matthias Scholz, Fatma Kaplan, Charles L. Guy, Joachim Kopka, and Joachim Selbig, "Non-linear PCA: a Missing Data Approach," Bioinformatics, Vol. 21 no. 20 2005, pp. 3887–3895, August 2005.

[9] Lindsay I Smith, "A Tutorial on Principal Components Analysis," http://csnet.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf, Feb. 26, 2002, (last accessed in July 2006).

[10] Kai Huang, Meel Velliste, and Robert F. Murphy, "Feature Reduction for Improved Recognition of Subcellular Location Patterns in Fluorescence Microscope Images," Proceedings of SPIE Vol. 4962, SPIE, 1605-7422/03, pp. 307-318, 2003.

[11] Imola K. Fodor, "A Survey of Dimension Reduction Techniques," Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, California, http://www.llnl.gov/CASC/sapphire/pubs/148494.pdf, June 2002 (last accessed in July 2006).

[12] Edward C. Malthouse, "Limitations of Nonlinear PCA as Performed with Generic Neural Networks," IEEE Transactions on Neural Networks, Vol. 9, No. 1, pp.165-173, January 1998.

[13] Burnham, K. P., and D. R. Anderson, "Model Selection and Inference: a practical information-theoretic approach," 1st Edition, Springer-Verlag, New York, ISBN 0-387-98504-2, pp. 353, 1998.

[14] Robert P.W. Duin, Marco Loog, and R. Haeb-Umbach, "Multi-class Linear Feature Extraction by Nonlinear PCA," IEEE, http://ieeexplore.ieee.org/iel5/7237/ 19583/00906096. pdf ?isnumber= 19583&arnumber=906096, pp. 398-401, 2000.

[15] Hornik K., Stinchcombe M., and White H., "Multilayer Feedforward Neural Networks are Universal Approximators," Neural Networks, Vol. 2, pp. 359-366, 1989.

[16] Yuval, "Neural network training for prediction of climatological time series, regularized by minimization of the generalized cross validation function," Monthly Weather Review, 128, pp. 1456–1473, 2000.

[17] Website Link, "Nonlinear Reduction of High-dimensional Data," http://www.quantlet.com/mdstat/scripts/csa/html/node159.html, (last accessed in July 2006).

[18] Jon Shlens, "A Tutorial on Principal Component Analysis-Derivation, Discussion and Singular Value Decomposition," University of California, San Diego, http://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf, March 2003.

[19] Ella Bingham and Heikki Mannila, "Random Projection in Dimensionality Reduction: Applications to Image and Text Data," Conference on Knowledge Discovery in Data, Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, California, pp. 245-250, 2001.

[20] Website Link, "Principal Components Analysis," http://www.cis.hut.fi/jhollmen/dippa/node30.html, (last accessed in July 2006).

[21] Software Help File, "Regularization," Matlab Version 7.1.0.246 (R14) Service Pack 3, Neural Network Toolbox, August 2005.

[22] Ashutosh Mishra, "Automatic Speaker Identification using Reusable and Retrainable Binary–Pair Partitioned Neural Networks," M.S. Thesis, E.C.E Dept., Old Dominion University, pp. 23-24, May 2003.

VITAE

TARA SINGH 757.309.3882
234 Kaufman Hall, Old Dominion University, Norfolk, VA 23508 tarasingh02@gmail.com

EDUCATION

Old Dominion University, Norfolk, VA **Aug '06**
Master of Science in Electrical Engineering
GPA: 3.76/4.00
Areas of Concentration: Digital Signal Processing

College of Tech.,G.B. Pant Univ of Ag. & Tech., India **June'01**
Bachelor of Technology in Electrical Engineering
GPA: 4.23/5.00, First Class with Distinction

WORK EXPERIENCE

Research Assistant – Dept. of Electrical and Computer Engineering **Aug '04 – Aug'06**

Application of Nonlinear principal component analysis (NLPCA) in Automatic speaker identification. Tools used include Matlab, C, Cooledit, Fortran

Teaching Assistant – Dept. of Electrical and Computer Engineering **Aug '04 – Aug'06**

Teaching assistant and grader for Digital System Design and Explore Engineering courses

Academic Assistant-Student Support Services **Aug '04 – Aug'06**

Tutor for undergraduate level Mathematics and Physics courses

Power Plant Engineer- Samtel Color Limited, Delhi, India **Oct'02-Aug'04**

Operation of 5.5 MW capacity Gas turbine. Operation and maintenance of Electrical distribution system, Involved in energy conservation project for reduction in the power consumption of plant using six sigma approach

Application Engineer-Trident Techlabs, Delhi, India **June'01-Oct'02**

Application and customization of power system softwares (CYME, SKM, PSAF, CDEGS, ETAP). Provided the technical support for analyzing the system parameters, formulation of the project report for recommending the measures to be implemented for system performance improvement