UNIVERSITY OF BIRMINGHAM

Feature extraction for robust automatic target recognition

by

Guy Thomas Maskall

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Electronic, Electrical and Computer Engineering
The University of Birmingham
November 2006

Abstract

This thesis documents research into the use of feature extraction for achieving robust automatic target recognition (ATR). Classification experiments were performed on inverse synthetic aperture radar (ISAR) images of military vehicles. The issue of robustness was considered over differences between vehicles used for training and those used for testing a classifier. These differences were broken down into two categories: the vehicle used for testing was represented during training, but its configuration was different; and the vehicle used for testing was not represented during training. The specific approach adopted for performing the feature extraction was a flexible radial basis function network. The merit of a given set of extracted features was measured through the error rate obtained using the nearest-neighbour classification rule and compared with linear features. Comparison was also made with results from a support vector machine. A major result of this work has been to perform classification on an extensive and varied real-world data set to show that configuration changes to vehicles are not the major challenge previously thought. Instead it is recommended that future research tackle the challenge of training a classifier on simulated or ISAR data and testing on data from an operationally-representative sensor.

For Shannon.

love you . . . ever and ever

Acknowledgements

I would like to express my indebtedness to Dr Andrew Webb of QinetiQ for his years of forbearance and guidance. I would also like to sincerely thank Mr David Pycock of The University of Birmingham, whose input was invaluable in forging this thesis. I am grateful to Prof Chih-Jen Lin of National Taiwan University for his prompt and helpful answers to my queries about LIBSVM.

A big thanks goes to "the pod what was" — Mr Matthew (my driving licence is in perfect condition) Farrow, Dr Sunil (yes, your cabinet's locked) Manchanda, and Mr Andrew (watch out for that turtle head) May — for mutual moral support through some turbulent times.

Some of this work was initially funded by the UK MoD Corporate Research Programme. PhD fees were met by QinetiQ.

Contents

List of Figures

List of Tables

Glossary

A A matrix

$\mathbf{A}^{\mathbf{T}}$ Transpose of matrix **A**

$\langle \mathbf{x}, \mathbf{y} \rangle$ Inner-product of vectors **x** and **y**

$\triangleq$ equal to, by definition

$\mathcal{O}(.)$ Scales as the order of

1-NNR Special case of k-NNR with $k = 1$

ANN Artificial Neural Network

ADU Air Defence Unit

APC Armoured Personnel Carrier

ATR Automatic Target Recognition

$\text{argmax}_w(.)$ The w that maximizes the argument

$E(.)$ Expected value of the argument

GTM Generative Topographic Mapping

EM Expectation Maximization

ISAR Inverse Synthetic Aperture Radar

i.i.d. Independent and identically distributed

k-NNR k-Nearest Neighbour Rule

LDA Linear Discriminant Analysis

LOO Leave One Out

MARS Multivariate Adaptive Regression Splines

MBT Main Battle Tank

MDS Multi Dimensional Scaling

ML Maximum Likelihood

MLP Multilayer Perceptron

m**-D** m-dimensions, or m-dimensional

min(a,b) The minimum of a,b

NLM Nonlinear Mapping

NLPCA Nonlinear Principal Component Analysis

PCA Principal Component Analysis

$\mathbb{R}^p$ p-dimensional real space

RAM Radar Absorbant Material

RBF Radial Basis Function

RBFN Radial Basis Function Network

RDA Radar Depression Angle

SVC Support Vector Classifier

SVM Support Vector Machine

sgn(.) The sign, -1 or $+1$, of the argument

Tr. The Trace of a matrix

feature space the p-dimensional space in which a single point represents a vector of p features

measurement space the mn-dimensional space in which a single point represents an $m \times n$-pixel image, a.k.a. image space or full-dimension space

Chapter 1

Introduction

1.1 General

This thesis considers the problem of robust classification of radar data. Specifically, the data comprise inverse synthetic aperture radar (ISAR) images of military armoured vehicles. The challenges to achieving robustness in the classification arise from changes between the operating conditions in force when the classifier is trained and when the classifier is employed against novel data. The poorer the match between the two sets of operating conditions, the poorer will be the expected performance of the classifier. Thus the goal is to generalize over these differences, rather than merely over two sets of data drawn independently and identically distributed (i.i.d), as is often the case with research on classification. Even after some 50 years of research and development in the field of pattern recognition, the general problem of recognizing complex patterns with arbitrary orientation, location etc. remains unsolved [55]. The detection and recognition of faces is a research area that has much in common with the detection and recognition of radar images of vehicles: there is much within-class variability and objects can be viewed under a wide range of angles and illumination conditions. That problem, too, is still wanting a general solution [75, 131, 134].

An important aspect of the work reported in this thesis was the creation of an extensive and varied data set of ISAR images of military vehicles. This facilitated experiments that posed 'hard' questions on real-world data, without resorting to simulation or the 'judicious' re-use of training data for classifier testing.

1.2 Radar as an all-weather military sensor

Since the days of World War II, radar has played a vital military role. From large early-warning installations to small units fitted in aircraft or missiles, the ability of radar to

detect and locate objects day and night in all weathers is unsurpassed by any other sensor system. Increasingly, however, a new demand is being placed on radar as a sensor: target recognition.

In the early days of aerial warfare, for example, the reach of weapon systems was far shorter than the range at which the human eye could identify an object: an enemy aircraft would be shot at if it displayed the enemy insignia and ground targets would be bombed if similarly identified to be enemy vehicles. In modern warfare, the reach of weapon systems is vastly greater than the range at which the human eye can even detect an object. Thus, a pilot would either have to get close enough to the suspected target to make a visual identification or the weapon must be engaged and launched based only on information from a sensor such as radar. The former carries the risk of placing personnel and hardware at much greater risk of enemy counter-attack. The latter carries the increased risk of wrongly targeting a neutral or friendly entity.

With the improved resolution of modern radar sensors, finer features can be resolved on objects being imaged. This can help with the recognition of the objects, by providing more discriminatory information, but has the disadvantage that the volume of data to be processed increases. The increased resolution and the ability of radar to operate in all weathers, day and night, makes radar an "ideal" sensor for detecting and identifying objects at long range. The high data-rate, however, increasingly means that the interpretation of the imagery produced requires an automated solution. In this context, the task is that of assigning objects to predetermined classes; this is supervised classification and such a classifier requires training, where it is shown how to recognize the different classes. The problem then becomes one of training a classifier to identify objects by their radar signature.

1.3 Challenges to achieving robust classification of radar images

A particular problem with radar imagery is the case of a classifier being tested on data obtained under conditions poorly represented during training. There are a number of reasons why test conditions will differ from training conditions. One reason is the sheer range of potential configurations of military ground vehicles. By their nature, most vehicles have rotating turrets which may be set at an arbitrary angle with respect to the main body of the vehicle. Another reason is that different missions or different stages of a mission may see a vehicle set up with different combinations of open hatches or kit boxes, or with extra-range fuel tanks attached. For example, a relatively minor change such as the opening of a hatch at a particular angle to the radar may result in the appearance

of a bright pixel where there was none before. These different vehicle configurations and different equipment-fits lead to a large configuration space required to specify the state of a vehicle.

The size of this configuration space is a particularly acute problem for radar imagery as a radar views objects as an ensemble of specular surfaces. Two scatterers contributing to the brightness of a particular image pixel may interfere constructively when observed from one angle and destructively when that angle is changed only slightly. This has the potential to give rise to significant variations of pixel intensities for small changes in viewing angle even for a target in a fixed configuration. Thus, not only is the configuration space large, in terms of possible vehicle configurations, it also appears that it may be necessary to sample it very finely in order to capture all the variability present.

Another reason why test conditions can be inadequately represented during training is the difficulty of obtaining samples with which to train a classifier. The crux of this is that it requires the acquisition of a suitable range of vehicles in satisfactory working order. This is easy enough when dealing with allies, who would likely provide examples of their vehicles to help ensure that a weapon system could recognize their vehicles as 'friendly' and not attack. However, an army is hardly likely to volunteer samples of its vehicles to an opponent for training a classifier. Remote surveillance may well yield some measurements, but with vehicle configuration being out of the control of the observer, the range of conditions under which a vehicle could be observed would be extremely patchy. In this respect there really is no substitute for being in possession of a vehicle to facilitate a controlled series of measurements. Even then, if a wide variety of possible configurations must be sampled very finely, it could take a prohibitively long time both to gather the measurements and then to train a classifier.

1.4 Benefit of achieving robust classification

If the goal of robust classification of military vehicles in radar images were attained, the following illustrative benefits would be realized:

- A weapon could be launched without knowledge of the exact position of the target.
- Even if the exact position of a target were known at weapon-launch, a mobile target may well relocate during the weapon flyout time. Provided the target remained within the weapon's search area, it would be detected, recognized, and targeted.
- An ability to classify targets implies an ability to prioritize and attack the most valuable or dangerous target present in an area.

- The exact configuration of the target would not need to be known at weapon-launch.
- Even if the exact configuration at weapon-launch were known, the weapon would still recognize the target if its configuration had changed.
- If an opposing force fielded a new vehicle design, retraining the classifier would not be required. At best, a robust classifier may correctly infer the vehicle class from features (e.g. the presence of turret, gun barrel, and tracks suggests a Main Battle Tank). At worst, a robust classifier would be less susceptible to confusion arising from a previously-unseen vehicle.
- For surveillance systems gathering high-resolution imagery very quickly, an automated classifier could annotate the scene and greatly aid the human observer.

Realistically, perfect robustness and ability to generalize will not be achieved. However, the better the ability of a classifier to generalize across operating conditions, then the closer to the above ideals will be the benefits realized.

1.5 Approach adopted

This thesis adopts a multivariate analysis [32] view of radar images of military vehicles. In this manner an image comprising p pixels is viewed as a single point in $\mathbb{R}^p$. Successive images of a vehicle, gathered as the viewing angle changes, will thus be represented by multiple points in $\mathbb{R}^p$. Likewise, images of a different vehicle can be described as points in the same $\mathbb{R}^p$. There will be spatial relationships in $\mathbb{R}^p$ between the points of one vehicle, spatial relationships between the points of the other vehicle, and spatial relationships between the points of the two vehicles. The exact positioning of a point representing an image in $\mathbb{R}^p$ will be affected by the location of the vehicle within the image and by variability in individual pixels. This variability is caused both by noise and by the physical (specular) processes previously mentioned.

By performing feature extraction on this space we seek to obtain a subspace that retains desirable spatial relationships whilst rejecting unwanted variability. Performing classification in this subspace thus offers a route to achieving improved robustness to changes in vehicle configuration by transforming the input measurements into a reduced-dimension space in which the representations of the data are rendered less sensitive to unwanted variability whilst retaining structure that is important for classification.

Thus the aim of this work is to seek such a dimensionality-reducing transformation of the data, modelling the transformation by a radial basis function network because of its flexibility and ease of training. As the purpose of the feature extraction is for input to a

classifier, the success of any particular approach is assessed by performing classification, using the nearest neighbour rule, in the derived feature space and measuring the resulting error rate. The lower the error rate, particularly relative to a given baseline such as performance in the original measurement space ($\mathbb{R}^p$), the more successful the approach.

1.6 Outline

The outline of this thesis is as follows. Chapter 2 documents a review of techniques for feature extraction. The field is large and the treatment here concentrates on providing a logical progression from common linear techniques to nonlinear techniques. In this way the thesis attempts not just to present a catalogue of relevant research, but to show the flow of ideas that relate various approaches and culminate in the chosen nonlinear method. The detail behind the radial basis function approach to modelling the feature extraction is given in chapter 3. Modifications to the basic approach are discussed. The design of the experimental method is detailed in chapter 4. Also given in that chapter are further descriptions of the aim of the research and the data used. Baseline classification results using features calculated by two common linear methods of feature extraction are given in chapter 5 along with classification results obtained using the images. The results of pilot studies projecting the radar data to feature space, including the nonlinear features obtained using the radial basis function approach, are shown. The results of classification performed in the nonlinear feature spaces are presented in chapter 6. The class of algorithms known as support vector machines (SVMs) are introduced in chapter 7 along with an illustrative example of the SVM behaviour and results when applied to the radar data. Arguably the most important chapter of the thesis is chapter 8 which provides a full comparison of the various approaches: classification without any feature extraction, classification in linear and nonlinear feature spaces, and classification with the SVM. Finally, a summary of the achievements of the thesis is given in chapter 9, the conclusions in chapter 10, and the recommendations for further work in chapter 11.

Chapter 2

Literature review

2.1 Introduction

This chapter presents a literature review for techniques of feature extraction. The field is vast and such a review must be constrained. The intent of this chapter is to present a flow of ideas, from the simple linear to the complex nonlinear, with the goal of not merely presenting a catalogue listing of techniques but of demonstrating the inter-relations between them.

Section 2.2 introduces the linear techniques of Principal Components Analysis and Linear Discriminant Analysis. Principal Components Analysis (2.2.1), in particular, has long been a staple of researchers' toolboxes and has seen numerous attempts to generalize it to include nonlinearities. Some of these are described in section 2.3. The distinction made here between, for example, "Principal Curves and Surfaces" and "Latent Variable Models" is more one of emphasis than of separating wholly different techniques.

The progression of ideas then continues, via Support Vector Machines (section 2.4) where kernel functions are introduced, through Multidimensional Scaling (section 2.5), to Radial Basis Functions (section 2.6), which also use kernel functions. Finally, in sections 2.7 and 2.8 the strands are pulled together, culminating in the use of Radial Basis Function Networks to perform Multidimensional Scaling as a technique for feature extraction applied to data prior to classification — the technique adopted in this thesis.

2.2 Linear methods

2.2.1 Principal components analysis

Principal Components Analysis (PCA) [4, 36, 127] is a common technique for performing feature extraction and dimensionality reduction. The aim of PCA is to derive new, un-

correlated, variables that are linear combinations of the original variables. The procedure is unsupervised as it does not depend upon any class labels assigned to the training data. It merely takes the data as points in $\mathbb{R}^p$ and determines axes that account for proportions of the variance in the data. By projecting the p-dimensional data onto a subspace determined by the first m of these dimensions, assuming $m < p$ and the dimensions are ordered appropriately, a reduction in dimensionality of the data is achieved that is optimal in a sum-of-squares sense.

Bishop [4] writes a particularly lucid and succinct account, introducing PCA as an optimum unsupervised reduction of the dimensionality of a dataset before going on to show that the basis vectors of the projection are given by the eigenvectors of the scatter matrix $\mathbf{S}$ (2.1) of the N data vectors $\{\mathbf{x}_n : n = 1, \ldots, N\}$.

$$\mathbf{S} = \sum_n (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^{\mathrm{T}}, \tag{2.1}$$

where

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_n \mathbf{x}_n.$$

The scatter matrix is merely $(N - 1)$ times the sample covariance matrix (unbiased estimate). Indeed, multiplication of $\mathbf{S}$ by any scalar is unimportant in this context as it will not change the direction of the eigenvectors or their relative importance in accounting for variance in the data. The scatter matrix will always be real and symmetric thus guaranteeing that orthonormal basis vectors can always be obtained [110]. The number of useful basis vectors is at most $\min(p, N - 1)$. The error, E_m, in approximating the data using the eigenvectors with the m largest eigenvalues is proportional to the sum of the eigenvalues of the unused eigenvectors (2.2).

$$E_m = \frac{1}{2} \sum_{i=m+1}^{p} \lambda_i \tag{2.2}$$

PCA is also referred to as the Karhunen-Loève transform and the Hotelling transform. The latter being somewhat uncommon and mostly in the context of image processing [41].

As already stated, PCA is an unsupervised technique: it makes no use of the data labels (classes) and so can produce results that are significantly sub-optimal for the purposes of classification. An example of this is shown in figure 2.1. Projecting the data to one dimension usually uses the first principal component, $\mathbf{u}_1$, which accounts for maximum variance. However, in this case, this would lose any separation of the classes. Instead, it is the second principal component, $\mathbf{u}_2$, that defines the direction offering optimal class separation. This is, however, an extreme example, and such an unsupervised method can

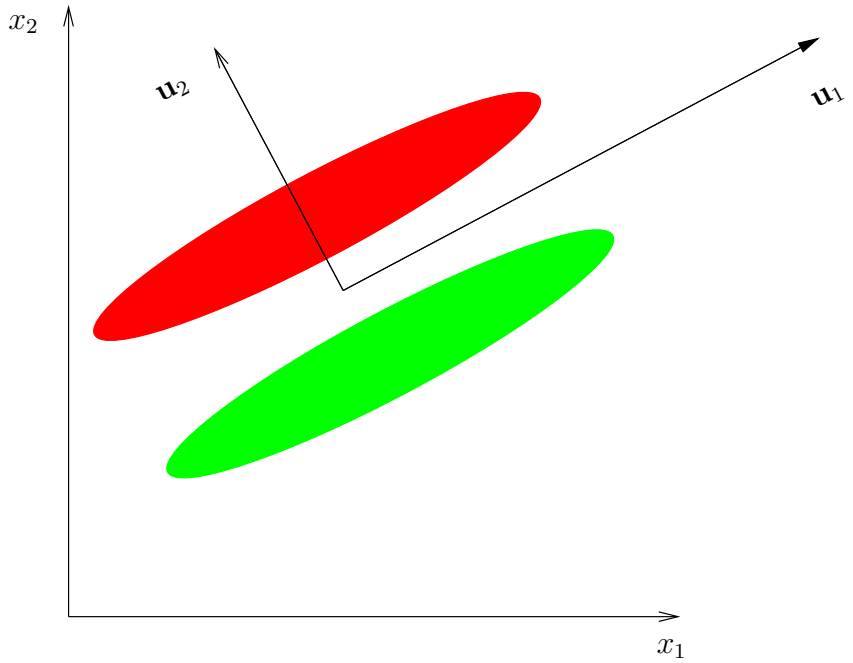


Figure 2.1: *Two classes, red and green, each have an elliptical distribution. The first principal component,* $\mathbf{u}_1$*, will tend to align in the direction of maximum variance, which loses separability between the classes. Here it is the second principal component,* $\mathbf{u}_2$*, that defines a direction in which the classes are separable.*

be useful in many applications.

2.2.2 Linear discriminant analysis

Linear discriminant analysis (LDA) projects the data values onto axes in a manner that is optimal for discrimination, rather than representation, as with PCA. Fisher's approach [36, 127] to the two-class discrimination problem was to find a projection onto a 1-dimensional line that optimally separated the two classes. The criterion proposed by Fisher was the ratio of between-class to within-class variances. The solution to Fisher's criterion is a line that joins the means of the two classes. LDA extends this approach to the C-class problem involving $C-1$ discriminant functions. An alternative name for the multiclass generalization of Fisher's linear criterion is Multiple Discriminant Analysis (MDA) [36]. Here we use the term LDA to refer to a C-class generalization of Fisher's criterion.

To generalize Fisher's criterion to C classes, following the approach of Duda, Hart, and Stork [36], we first must generalize the between-class and the within-class variances to C classes. We define the within-class scatter matrix

$$\mathbf{S}_W = \sum_{i=1}^{C} \mathbf{S}_i \tag{2.3}$$

where

$$\mathbf{S}_i = \sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \bar{\mathbf{x}}_i)(\mathbf{x} - \bar{\mathbf{x}}_i)^{\mathrm{T}} \tag{2.4}$$

and

$$\bar{\mathbf{x}}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}.$$

So the within-class scatter matrix $\mathbf{S}_W$ (2.3) is the sum of the individual scatter matrices for each class (2.4). Comparing (2.4) with (2.1) we see that $\mathbf{S}$ is the scatter matrix for the dataset as a whole, calculated using the global mean ($\bar{\mathbf{x}}$); and $\mathbf{S}_i$ is the scatter matrix for the subset, $\mathcal{D}_i$, of the dataset belonging to class i using the mean ($\bar{\mathbf{x}}_i$) of the n_i samples in $\mathcal{D}_i$.

The general between-class scatter matrix is given by

$$\mathbf{S}_B = \sum_{i=1}^{C} n_i (\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})^{\mathrm{T}}. \tag{2.5}$$

The goal of LDA can then be realized by finding a transformation matrix $\mathbf{W}$ that in some sense maximizes the ratio of the between-class scatter (2.5) to the within-class scatter (2.3). A simple scalar measure of scatter is the determinant of the scatter matrix [36]. This leads to the criterion function

$$J(\mathbf{W}) = \frac{\left|\mathbf{W}^{\mathrm{T}} \mathbf{S}_B \mathbf{W}\right|}{\left|\mathbf{W}^{\mathrm{T}} \mathbf{S}_W \mathbf{W}\right|}. \tag{2.6}$$

The columns, $\mathbf{w}_i$, of an optimal $\mathbf{W}$ are the generalized eigenvectors that correspond to the largest eigenvalues in

$$\mathbf{S}_B \mathbf{w}_i = \lambda_i \mathbf{S}_W \mathbf{w}_i. \tag{2.7}$$

Because $\mathbf{S}_B$ is the sum of C matrices of rank one or less, and because only $C - 1$ of these are independent, $\mathbf{S}_B$ is of rank $C - 1$ or less. Thus, no more than $C - 1$ of the eigenvalues are nonzero, and the desired eigenvectors satisfying (2.7) correspond to those nonzero eigenvalues [36]. In consequence of this, LDA defines a projection from d-dimensional space down to at most $(C-1)$-dimensional space. This can be conceptualized geometrically by considering the subspace spanned by the class mean vectors. The mean vectors of two groups define a line, hence Fisher's criterion (for two groups) defines a line. The mean vectors of three groups define at most a plane, and so on. Thus, although LDA is a supervised technique and obtains a projection that optimally separates classes, it may define too drastic a reduction of dimensionality for small numbers of classes.

2.2.3 Is linear always enough?

Linear methods have their advantages. They are usually relatively straightforward to implement using linear algebra. Furthermore, as the models obtained are no more than linear in the observed variables, the solutions are generally interpretable in terms of the original observed variables. If any relationships between the observed variables are inherently linear then linear techniques, such as those discussed, are appropriate tools.

The relationships between the measured variables of many real-world processes contain nonlinearities [33, 34]. With multiple sensors monitoring processes, e.g. temperature sensors distributed along a chemical processing chain; multiple features being measured, e.g. temperature, pressure, acidity, and salinity within a reaction vessel; and high-resolution imaging sensors, e.g. modern high-bandwidth radar systems; the vectors of measured variables can have a large number of dimensions. The combination of high dimensionality with nonlinear relationships between variables means that the patterns will be scattered along nonlinear subspaces [55] within the high-dimensional measurement space.

The structure of this subspace will not be efficiently modelled by hyperplanes and the use of linear PCA in such circumstances can be inadequate [87]. The argument frequently used to support PCA is that the eigenvectors associated with the largest eigenvalues contain the information of most interest and the axes defined by the eigenvectors with the lowest eigenvalues (the minor components) consist of noise or unimportant variance. It has been shown that if PCA is used in nonlinear problems, then these minor components do not always consist of noise or unimportant variance [130]. Hence, there is great interest in developing techniques capable of capturing nonlinear structure in data.

However, whilst hyperplanes may be suboptimal for the global fitting of nonlinear surfaces, improved performance can be obtained by local fitting. Such a local PCA technique was described by Kambhatla and Leen [56]. The method comprised two stages: a partitioning of the input space followed by the determination of principal axes within each region. A disadvantage of this approach was that the two stages were distinct: the partitioning of the space into regions was not optimized for subsequent calculation of the principal components. The requirement to perform two distinct steps — partitioning and PCA — was obviated by Tipping and Bishop [114]. Local PCA models were combined within the framework of a probabilistic mixture in which all the parameters were determined from Maximum Likelihood (ML) using an Expectation-Maximization (EM) algorithm. The application of the approach to image compression and hand-written digit recognition was demonstrated, but one possible disadvantage was that it did not directly minimize squared reconstruction error.

2.3 Generalizing principal components analysis

2.3.1 Principal curves and surfaces

Just as the first linear principal component yields a straight-line summary of the data that satisfies an orthogonal distance criterion, Hastie and Stuetzle [43] proposed a principal curve that also minimizes an orthogonal distance. Such a curve passes through the middle of the data in a smooth manner such that any point on the curve coincides with the average of all data samples that project onto that point on the curve. Principal curves have their attractions as nonlinear generalizations of PCA: like PCA they minimize an orthogonal distance criterion, and if the curve is constrained to a straight line, the result is a linear principal component. However, unlike PCA, principal curves do not define a transformation of the data. This makes them unsuitable as a tool for performing feature extraction on new measurements.

Following from Hastie and Stuetzle's definition of self-consistency [43], Webb [124] constructed principal surfaces using radial basis functions with the goal of maximizing variance. Unlike Hastie and Stuetzle's curves, Webb's curves and surfaces could stretch, like fitting a rubber sheet rather than a sheet of paper to a nonlinear surface. The model parameters were determined by solving a simple generalized eigenvector equation. It was then shown, in [126], that a variance-maximizing transformation could be obtained via a loss function, with nonlinear PCA being modelled as a dimension-reducing transformation rather than a curve-fitting exercise.

2.3.2 Latent variable models

A further unsettling characteristic of Hastie and Stuetzle's approach, as noted by themselves [43], was that if data were generated from the model $\mathbf{X} = \mathbf{f}(s) + \boldsymbol{\epsilon}$, with $\mathbf{f}$ smooth and $\boldsymbol{\epsilon}$ being a noise term with $E(\boldsymbol{\epsilon}) = \mathbf{0}$, then $\mathbf{f}$ will not, in general, be a principal curve of the resulting data points. Tibshirani [112] proposed an alternative definition of a principal curve; one that does not share this limitation. That approach introduced a generative distribution based on a mixture of Gaussians, with model parameters determined through a likelihood function and EM. The number of Gaussian components was equal to the number of data points. Smoothing was imposed by penalizing the likelihood function with a derivative-based regularization term.

In a similar vein was the Generative Topographic Mapping (GTM) [5]. This was also a latent variable model utilizing a mixture of Gaussians with model parameters determined by ML and EM. The mapping was defined by specifying a set of points $\{\mathbf{x}_i\}$ on a uniform grid in latent space, together with a set of basis functions $\{\phi_j(\mathbf{x})\}$, typically radially

symmetric Gaussians. A common width parameter, along with the number and spacing of the basis functions, determined the smoothness of the manifold.

LeBlanc and Tibshirani [67] built upon ideas from Hastie and Stuetzle's principal curves [43] and Friedman's Multivariate Adaptive Regression Splines (MARS) [38] to develop a nonlinear generalization of principal components motivated from the perspective of function approximation. The approach, known as Adaptive Principal Surfaces, was also a nonlinear factor model built upon radially symmetric basis functions, with the centres, or knots, located at unique data samples. This model, however, was fitted through the minimization of a residual sum-of-squares error term. An important disadvantage of the approach, in common with principal curves, is that it only yielded a lower-dimension approximation of the data: it did not define a transformation for new measurements.

2.3.3 Neural networks for principal components analysis

Although Artificial Neural Networks (ANNs) can implement linear PCA [3, 86, 98], the eigenvectors can be computed efficiently using well-known numerical methods, e.g. [94]. It is for the inclusion of nonlinearities that the situation becomes much more favourable for a neural implementation. Neural networks with a finite number of hidden units are capable of universal approximation of nonlinear functions [27, 48, 49, 52].

Dong and McAvoy [33, 34] developed a neural network, trained via a conjugate gradient learning method [68], that learned a projection giving rise to the principal component scores obtained from principal curves [43]. The architecture of this network is shown in figure 2.2, where $X = [x_1, x_2, \ldots, x_p]^{\mathrm{T}}$ denotes the input data, $T = [t_1, t_2, \ldots, t_m]^{\mathrm{T}}$ denotes the projection of the data to feature space, and $X' = [x'_1, x'_2, \ldots, x'_p]^{\mathrm{T}}$ denotes the resconstructed (approximated) data space. By inserting the scores derived using principal curves into the $T = [t_1, t_2, \ldots, t_m]^{\mathrm{T}}$, the network learned a transformation that effected $X \rightarrow T$. This nonlinear PCA (NLPCA) method effectively comprised two three-layer networks. If all that is required is a projection of data (X) to feature space (T), only the mapping network $\mathcal{G}$ is needed.

The alternative to learning explicit principal scores is to join the output of $\mathcal{G}$ to the input of $\mathcal{H}$. This forms an autoassociative network with a bottleneck layer that seeks a compact representation T of the input data. Kramer [60] used such an approach and demonstrated its utility for preprocessing data to enable sensor-based calculations to be performed correctly even in the presence of large sensor biases and failures [61]. For performing NLPCA the network is split after training and the output of $\mathcal{G}$ used to define a projection to the NLPCA subspace. A disadvantage of Kramer's NLPCA is that it models the projection index with a continuous function. This has several undesirable consequences, for example curves and surfaces that intersect themselves cannot be

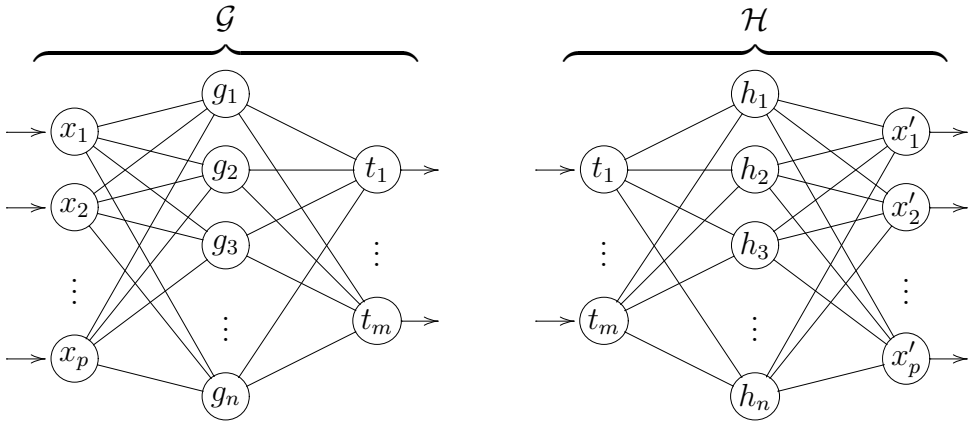


Figure 2.2: *Neural network for performing nonlinear principal component analysis.* $X = [x_1, x_2, \ldots, x_p]^T$ *and* $X' = [x'_1, x'_2, \ldots, x'_p]^T$ *are the input data and approximation to the input data, respectively. The* g_i *and* h_i *are the (nonlinear) hidden layer nodes in the mapping and demapping layers, respectively. The* t_i *are the projected data points, or 'principal scores', as output by the mapping layer and input by the demapping layer.*

approximated [73, 74].

2.3.4 Summary

Principal curves (surfaces) is one nonlinear generalization of linear Principal Component Analysis. The technique can capture nonlinear structure in data, but, because it obtains a mapping of data points rather than defining a transformation that is applied to data points, it does not readily admit the inclusion of new measurements; this is a requirement of any practical method of dimensionality reduction if it is to be a precursor to classification. Dong and McAvoy's approach was to treat the projection indices (distances along the curve of projected data points) as target nonlinear principal component scores, and trained a network to learn the effecting transformation. Thus, new measurements could be subjected to this 'nonlinear principal component' dimensionality reduction. This methodology was in contrast to that of Kramer, where a five-layer autoassociative network, with implicit reduced-dimensionality scores in the middle, was used. After training, the network was broken apart as only the first half, the mapping half, is of interest for performing dimensionality reduction.

Artificial Neural Networks can be used to perform linear PCA, but they struggle to compete with modern, efficient closed-form solutions. It is through their universal function approximation properties that they show their merit in learning nonlinear PCA mappings.

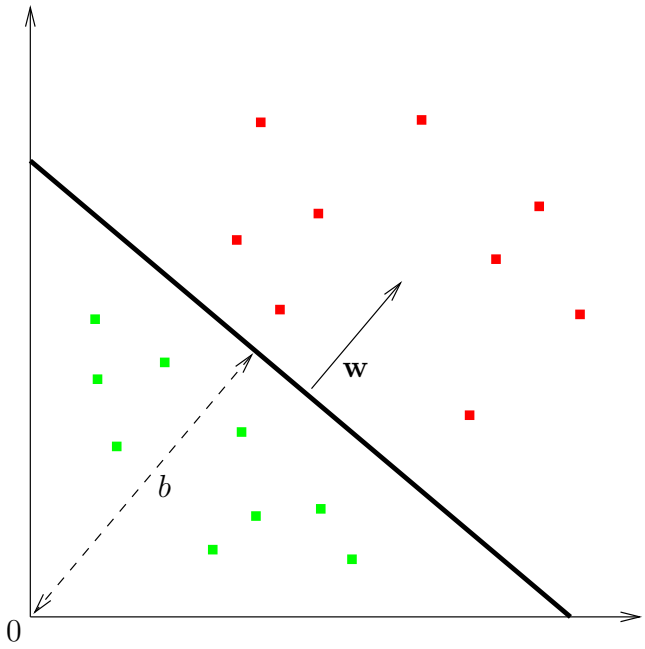


Figure 2.3: *A hyperplane,* $(\boldsymbol{w}, b)$*, separating two classes (red and green) in two dimensions.*

2.4 Support vector machines

Introduced in the early 1990's and based on a solid theoretical background [18, 119, 120], Support Vector Machines (SVMs) have received an enormous amount of interest [26, 36, 44, 101]. This section motivates and introduces SVMs following the progression of ideas in [26].

2.4.1 Linear learning methodology

Consider the two, linearly-separable classes in figure 2.3. Such binary classification can be performed using the real-valued function $f : \mathcal{X} \subseteq \mathbb{R}^p \to \mathbb{R}$, where a pattern $\mathbf{x}$ is assigned to one class if $f(\mathbf{x}) \geq 0$, and to the other class if $f(\mathbf{x}) < 0$. For such linearly-separable classes, the function f is a linear function of $\mathbf{x} \in \mathcal{X}$ and so can be written as:

$$\begin{aligned} f(\mathbf{x}) &= \langle \mathbf{w}, \mathbf{x} \rangle + b \\ &= \sum_{i=1}^{p} w_i x_i + b, \end{aligned} \tag{2.8}$$

where $(\mathbf{w}, b) \in \mathbb{R}^p \times \mathbb{R}$ are the parameters controlling the function and the decision rule is given by $\operatorname{sgn}(f(\mathbf{x}))$. The geometric interpretation of this decision rule is an input space split into two by the hyperplane $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$, with $\mathbf{w}$ defining the direction perpendicular to the hyperplane and b moving the hyperplane parallel to itself.

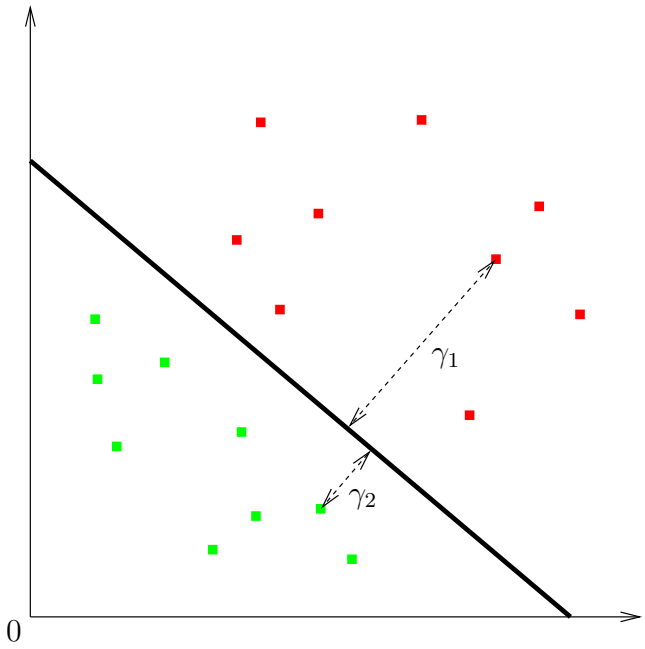


Figure 2.4: *The geometric margins, γ_1 and γ_2, of two points.*

The earliest iterative procedure for learning a linear classification rule was that of the perceptron. This learning methodology is 'on-line' and 'mistake-driven': training patterns are presented one-at-a-time and, if the current configuration of parameters misclassifies a pattern, the $\mathbf{w}$ and b are updated directly so as to move the hyperplane towards the desired orientation and position. The algorithm is guaranteed to converge provided the data are linearly separable. The solution obtained is not guaranteed to be unique: any hyperplane separating the two classes is a valid solution, and causes the algorithm to terminate. A more desirable solution is one which is unique, and this can be achieved by realizing the solution that maximizes a quantity called the margin.

The functional margin of an example $(\mathbf{x}_i, y_i)$ with respect to a hyperplane $(\mathbf{w}, b)$ is defined:

$$\gamma_i = y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b),$$

where y_i is the class label of pattern $\mathbf{x}_i$; being $+1$ for the 'positive' class and -1 for the 'negative' class. Thus $\gamma_i > 0$ implies correct classification of the example. The functional margin distribution of a hyperplane $(\mathbf{w}, b)$ with respect to a training set S is the distribution of the margins of the examples in S. The minimum of the margin distribution is sometimes referred to as the functional margin of hyperplane $(\mathbf{w}, b)$ with respect to S. If the parameters are normalized, so that the hyperplane is described by $(\mathbf{w}/\|\mathbf{w}\|, b/\|\mathbf{w}\|)$, the geometric margin is obtained, which measures the Euclidean distances of the points from the decision boundary. This is illustrated in figure 2.4. The margin, γ, of training set S is the maximum geometric margin over all hyperplanes. The

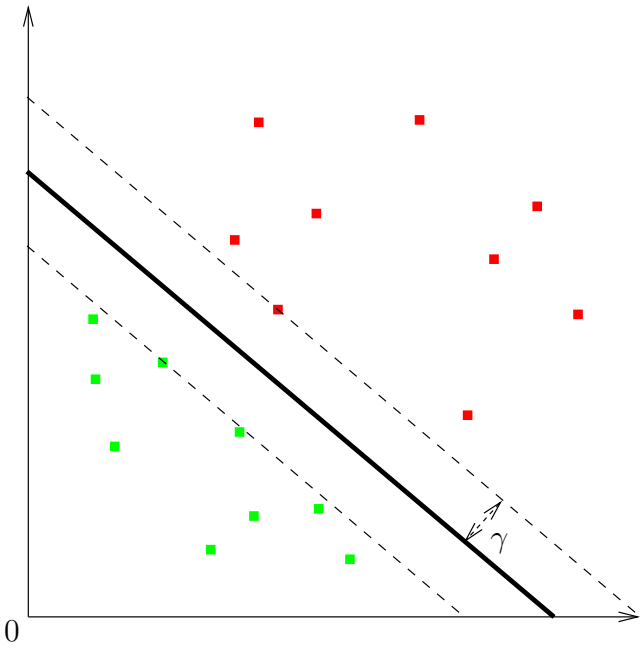


Figure 2.5: *The training set margin, γ, maximization of which yields a unique separating hyperplane.*

hyperplane realizing this maximum is the maximal margin hyperplane. It can be regarded as unique and optimal. This is shown in figure 2.5.

2.4.2 Dual representation

Because the perceptron algorithm works by adding misclassified examples to the weight vector, the final hypothesis will be a linear combination of the l training points: $\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$. The α_i are positive values proportional to the number of times misclassification of $\mathbf{x}_i$ caused the weight vector to be updated. This is sometimes referred to as the embedding strength of pattern $\mathbf{x}_i$.

For fixed samples S, $\boldsymbol{\alpha}$ can be thought of as an alternative representation of the hypothesis in dual coordinates. The decision function can thus be rewritten in terms of these dual coordinates:

$$\begin{aligned} h(\mathbf{x}) &= \operatorname{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) \\ &= \operatorname{sgn}\left(\left\langle \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i, \mathbf{x} \right\rangle + b\right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b\right) \end{aligned} \tag{2.9}$$

Notice in (2.9) that the decision function is now only expressed in terms of input patterns:

the normal to the hyperplane, $\mathbf{w}$, does not appear. Moreover, pattern $\mathbf{x}_i$ is weighted according to how important it is in defining the separating hyperplane.

The discussion, so far, of the simple perceptron has assumed the training data to be linearly separable. Already, however, many of the important concepts in the theory of SVMs have been encountered:

- the margin;
- the margin distribution;
- the dual representation.

2.4.3 Kernel-induced feature space

If the two classes in the finite training set S are not linearly separable in the input space, they can always be projected into a higher dimensional feature space in which they are separable by a linear classifier (hyperplane). The equivalent of (2.8) in this feature space is:

$$f(\mathbf{x}) = \sum_{i=1}^{l} w_i \phi_i(\mathbf{x}) + b, \tag{2.10}$$

where $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{F}$ is a nonlinear map from the input space $\mathcal{X}$ to some feature space $\mathcal{F}$. Thus there are two steps in the construction of this nonlinear learning-machine:

1. a nonlinear mapping, $\boldsymbol{\phi}(\mathbf{x})$, maps input space, $\mathcal{X}$, to the feature space, $\mathcal{F}$;
2. a linear machine then classifies the patterns in $\mathcal{F}$.

Because such an approach involves working directly in the higher dimensional feature space, it would rapidly fall foul of the curse of dimensionality, as well as requiring the explicit calculation of the image, $\boldsymbol{\phi}(\mathbf{x})$, of each input pattern, in the feature space. The fact that linear learning machines can be expressed in the dual representation (2.9), means that the hypothesis can be expressed as a linear combination of the training points, and the decision rule can be evaluated just using inner products between a test point, $\mathbf{x}$, and the training points, $\mathbf{x}_i$:

$$h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i \left\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}) \right\rangle + b\right) \tag{2.11}$$

Equation (2.11) is the analogue of (2.9) for classification performed in feature space. If the inner product $\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}) \rangle$ in feature space can be computed implicitly, as a function of the original input points, then the two steps needed to build a nonlinear learning machine

can be merged. A kernel function [12, 84, 99, 101, 104] is just such a direct computation method.

A kernel is a function K, such that for all $\mathbf{x}, \mathbf{z} \in \mathcal{X}$:

$$K(\mathbf{x}, \mathbf{z}) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{z}) \rangle ,$$

where $\boldsymbol{\phi}$ is a mapping from $\mathcal{X}$ to an (inner product) feature space $\mathcal{F}$. The use of the dual representation, with a kernel implicitly computing the inner product between two points in feature space, means that the dimension of the feature space need not affect the computation. The key is to find a kernel function that can be evaluated efficiently: the decision function can then be evaluated by, at most, l evaluations of the kernel, K, as in:

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b.$$

As an illustration of the introduction of nonlinearity into the feature space map, consider:

$$\begin{aligned} \langle \mathbf{x}, \mathbf{z} \rangle^2 &= \left(\sum_{i=1}^{p} x_i z_i \right)^2 \\ &= \left(\sum_{i=1}^{p} x_i z_i \right) \left(\sum_{j=1}^{p} x_j z_j \right) \\ &= \sum_{i=1}^{p} \sum_{j=1}^{p} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{p} (x_i x_j)(z_i z_j), \end{aligned}$$

which is equivalent to the inner product between the feature vectors $\boldsymbol{\phi}(\mathbf{x}) = (x_i x_j)_{i,j=1}^{p} = (x_1^2, x_1 x_2, \ldots, x_1 x_p, x_2 x_1, x_2^2, \ldots, x_p^2)^{\mathrm{T}}$, the monomials of degree 2. Thus, calculation of the kernel function $K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$ achieves the same end as the calculation $\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{z}) \rangle$, but the calculations are performed in the original input space. Clearly, the explicit calculation of the monomials becomes prohibitive as the dimensionality of the input space, p, and the degree, d, of monomial, increase. Instead, they can be computed implicitly by $K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^d$.

So far it appears to be necessary to create a complicated feature space, compute the inner product in that space, and find a direct method of computing that value in terms of the original inputs. In practice, a kernel is defined directly; the feature space is implicitly defined by the kernel. Not just any function can be chosen for K: Mercer's theorem [79]

gives necessary and sufficient conditions for $K(\mathbf{x}, \mathbf{z})$ to be an inner product in feature space $\mathcal{F}$.

2.4.4 Controlling the generalization capability

The use of a kernel function greatly increases the expressive power of the learning machine, whilst retaining the underlying linearity that ensures the learning task remains tractable. This increased flexibility comes with an increased risk of overfitting. A powerful kernel, that successfully separates noisy training points, with zero training error, is likely to generalize less well than a less powerful (less complex) function that accepts some degree of training error.

The successful control of the increased flexibility of kernel-induced feature spaces requires a sophisticated theory of generalization, one which is able to precisely describe which factors have to be controlled in the learning machine in order to guarantee good generalization. The theory of Vapnik and Chervonenkis [119] is the most appropriate for Support Vector Machines, and, historically, it has driven their development. The theory places reliable bounds on the generalization of linear classifiers and controls the complexity of linear functions in kernel space.

The key to good generalization performance is to minimize a quantity that reflects both the empirical risk and the complexity of the kernel function. The empirical risk can be reduced to zero through the use of an appropriately complex kernel, but the complexity term will become large, implying poor generalization. Similarly, a kernel of insufficient complexity can have an arbitrarily low measure of complexity, but will be unable to produce a low value of empirical risk. By trading off the two terms an optimal kernel can be found.

2.4.5 Support vectors

The positive semi-definiteness requirement that Mercer's theorem places on the kernel matrix $K(\mathbf{x}_i, \mathbf{x}_j)$, has the consequence that the optimization problem is convex and does not suffer from local minima. Furthermore, it turns out that not all of the l training points, as in (2.9), are required to define the solution: only those points closest to the hyperplane have non-zero weight. This sparsity of the dual representation, depending on only a reduced number of 'support' vectors, leads to both efficient algorithms and the name of the resulting learning machines: Support Vector Machines.

2.4.6 Summary

Support Vector Machines are kernel-based algorithms that make use of implicit, but constrained, nonlinear high-dimension spaces. They have a solid theoretical background and attractive learning rules. By combining linear learning methodology with nonlinear projections to a higher-dimensional space, they appear to have the best of both the linear and nonlinear worlds. Their solid basis in Statistical Learning (Vapnik-Chervonenkis) Theory, allows them to work implicitly in high-dimensional spaces without the problems inherent in working directly in such spaces. Key to their success is a family of kernels that allows this implicit nonlinear transformation of data.

2.4.7 Epilogue

There are publications that focus on tutorials for support vector regression [105] and classification [11, 50], and ν-SVMs [15, 16, 17, 106]. SVMs have been applied in the domain of classifying military targets [57, 132, 133].

Having introduced kernel functions and the concept of working implicitly in some high-dimensional feature space, it is worth noting that both PCA and Fisher's Discriminant have been 'kernelized' e.g. [83] and [100] — effectively applying these linear techniques in some high (possibly infinite) dimension nonlinear feature space. Thus, in the case of kernel-PCA, one can model higher than second-order statistics within the data. Unlike SVMs, however, this results in a non-sparse matrix as the components are expressed in terms of kernels associated with every training vector. Tipping [113] addressed this issue by approximating the covariance matrix in feature space using maximum-likelihood to obtain a sparse form of kernel-PCA. Kernel-based learning, with applications including the kernelization of PCA and Fisher's Discriminant, is covered in, for example, [84] and [101].

2.5 Multidimensional scaling

2.5.1 Introduction

Whereas the techniques discussed so far in this chapter operate directly on the measured data, perhaps via covariance or correlation matrices, Multidimensional Scaling (MDS) [25, 65] operates on a matrix of dissimilarities or proximities between data points. The objective of an MDS technique is to take the $N \times N$ matrix of the dissimilarities between the N p-dimensional data vectors and obtain an m-dimensional ($m \leq p$) configuration of points that best reflects the given dissimilarities. The key difference between MDS and the previously discussed methods is that MDS does not require the original

$N \times p$ data matrix to be known, only the matrix of dissimilarities. For example, Dillon and Goldstein [32] used MDS to visualize structure in the perception of similarities between products such as soft drinks and newspapers. The nature of the dissimilarities, how they are to be related to the distances in the derived (m-dimensional) space, the criterion used to quantify how closely the distances in the derived space resemble the dissimilarities, and how such a criterion is optimized are the main factors that give rise to the variety of MDS techniques developed.

We have used the term dissimilarities, although we could equally well use similarities. The key property of the input to MDS is that it represents how closely one entity is related to another; whether the numbers are large and decreasing, or small and increasing, for increasing similarity between pairs, matters little. Shepard [102] suggested the characterization of the (dis)similarities as substitutability: if A is more similar to B than to C, then the consequences are greater when C, rather than B, is substituted for A. In order to cover all specific concepts of closeness or nearness, in a psychological or associative sense, Shepard adopted the term 'proximity measure' to describe the number representing the closeness of the relation between a pair of entities in his analysis of proximities [102, 103].

Two basic types of MDS are nonmetric and metric MDS [65, 127]. Nonmetric MDS seeks only to preserve the rank order of proximities. Representing the proximity between pair (i, j) by δ_{ij} and the resulting derived distance by d_{ij}, then if $\delta_{ij} < \delta_{kl}$ nonmetric scaling attempts to obtain $d_{ij} < d_{kl}$. Metric scaling assumes that the values assigned to the δ_{ij} are meaningful, i.e. the proximities are quantitative, and seeks to obtain a functional relationship between the δ_{ij} and the d_{ij}. In other words, both types of MDS seek to obtain a functional relationship $d = f(\delta)$ between the proximities and the derived distances. In nonmetric MDS $f(.)$ is restricted to the class of monotone functions, whereas in metric MDS it could, for example, be a linear function $d = b\delta$ so that doubling δ would mean doubling d.

In the remainder of this section we shall discuss nonmetric scaling, starting with Shepard's [102, 103] seminal papers, as it provides some valuable insight into MDS. This is followed by a section on metric MDS.

2.5.2 Nonmetric multidimensional scaling

Shepard [102] described an algorithm designed to reconstruct the metric configuration of a set of points in Euclidean space using only nonmetric information about the proximities. The actual values of the proximities themselves would not be known, only the rank order of them. Two conditions were imposed upon the solution, namely monotonicity and minimum dimensionality. It is most interesting that these two conditions are generally sufficient to lead to a unique and quantitative solution. Given an initial starting config-

uration of points, the twin desires of monotonicity and minimum dimensionality can be thought of as forces acting on all of the points pulling them towards a new configuration until the opposing forces balance.

The monotonicity condition requires that the rank order of the $N(N-1)/2$ distances between the N points should be the inverse of the rank order of the $N(N-1)/2$ proximity measures. In other words, the two points corresponding to the pair of entities with the largest proximity measure should ideally end up with the smallest distance of separation in Euclidean space. Any desired rank order of distances between N points can be achieved in a space of at least $N-1$ dimensions [102]. For any particular arrangement of the points in $(N-1)$-dimensional space that does not meet the requirement of monotonicity, a better configuration can be obtained by moving the points in such a way as to stretch those distances that are too small and compress those distances that are too large. Thus the procedure defined was:

1. Start with an initial configuration of points.
2. Compute the $N(N-1)/2$ Euclidean distances between the points.
3. Rank order these distances from smallest to largest.
4. Compare this ordering with the ranking already obtained for the proximity measures.
5. Modify the coordinates of the points as required to improve the fit.
6. Repeat from step 2 until the relative improvement in the fit falls below some threshold.

This procedure alone, however, leads to no reduction of the dimensionality of the data. Neither does it yield a unique solution. Shepard's requirement for a solution of minimum dimensionality addressed these problems. By supplementing the monotonicity requirement with an additional requirement that the final configuration should be of the smallest possible dimensionality, a nontrivial and unique solution can be secured.

Consider a straight line of points and a curved segment of points. A choice cannot be made between them on the basis of the rank order of the points alone. However, the reconstruction of the distances from Cartesian coordinates can be effected with only one coordinate axis in the case of the straight line, whereas two such axes are required for the curved segment. In other words, the curve has a higher dimensionality in Cartesian coordinates. The key observation is that the curved segment can be straightened by systematically stretching the larger distances and shrinking the smaller distances. This

suggests that the way to flatten a configuration of points into a space of smaller dimensionality is to increase the variance of the distances by further stretching those distances that are already large and by further shrinking those distances that are already small. This was the method of analysis proposed by Shepard in [102]. After the monotonicity and dimensionality-reducing forces were in equilibrium the points still lived in the original $(N-1)$-dimensionality space, albeit within a subspace of this space. Hence, PCA was then performed to identify the subspace.

Shepard demonstrated the recovery of metric configurations from rank-ordered proximities [103]. The metric determinacy increases with the number of points. Specifically, if the number of points is sufficiently large with respect to the number of dimensions of the configuration then a rank ordering of proximities is enough to fix the relative locations of the points in a metric space. The greater the number of points, the more localized the fix.

Coxon [25] illustrated this same property on an example of Scottish mileages. Sixteen towns on the Scottish mainland were selected and the distances between them measured with a ruler, with a small error added. The resulting 120 distances were reduced to rank-order only and these rank-orders subjected to an MDS analysis. The derived 2-D configuration of the towns were then overlaid onto a map and shown to be visually in the correct geographic positions.

Kruskal [62, 63] built upon Shepard's work, but improved it in a number of important regards. Kruskal's approach still sought a monotonic function of rank-ordered proximities, but central to the endeavour was a numerical measure of the goodness-of-fit of the solution and a systematic method for optimizing it. Kruskal's advances on Shepard's "analysis of proximities" were subsequently incorporated into standard MDS computer programs [65].

For a given configuration of points, the calculated distance between the ith and jth points is denoted by d_{ij} and the corresponding distance required to actually satisfy monotonicity of the d_{ij} and δ_{ij} is denoted by $\hat{d}_{ij}$. The deviation of the single point (i,j) from monotonicity is thus given by $d_{ij} - \hat{d}_{ij}$ and summarized as the sum of squares of the deviations. This results in Kruskal's [62] "raw stress", which is made invariant to coordinate scalings (and the number of data points) by scaling with $1/\sum_{i<j} d_{ij}^2$. This results in a goodness-of-fit measure, Kruskal's stress, S:

$$S = \sqrt{\frac{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2}{\sum_{i<j} d_{ij}^2}} \tag{2.12}$$

By definition, the desired (best-fitting) configuration is the one that minimizes S. Kruskal adopted the method of "gradients" or "steepest descent" [62, 63] to minimize S,

a function of many variables.

In summary, the primary computation is to find the configuration that minimizes the stress (2.12) and the secondary problem is to find the values of $\hat{d}_{ij}$ from the values of d_{ij}. Kruskal's algorithmic architecture, as detailed in [63], was:

1. Initialize the configuration (of points).
2. Normalize the configuration.
3. Calculate the distances d_{ij}.
4. Fit the $\hat{d}_{ij}$ utilizing the rank order of the dissimilarities.
5. Calculate the stress and gradient of the present configuration.
6. If the criterion for a (local) minimum is satisfied then halt; else calculate the new configuration by the method of gradient descent and repeat from step 2.

2.5.3 Metric multidimensional scaling

The use of metric MDS for the analysis of nonlinear data structure was reported by Sammon [97]. This appears to mark the introduction of metric MDS into the pattern recognition literature [127]. Sammon used a loss function almost identical to Kruskal's stress (2.12) (also see Kruskal's comment in [64]). Sammon's form of the normalization or weighting of the stress gives a slight preference to small interpoint distances [85], i.e. local structure is emphasized, whereas Kruskal's form gives no preference, which is to be preferred when only the rank order of the input distances is known. The key difference between Sammon's and Kruskal's approaches is that Sammon was considering feature or measurement vectors as inputs. The difference (or dissimilarity) between input vectors could thus be calculated by a suitable metric (e.g. Euclidean) and so quantified, whereas previously in nonmetric MDS we have only seen qualitative dissimilarities. Again, the method of steepest descent was used for optimization.

Sammon's primary motivation was the nonlinear mapping (NLM) of the input vectors to 2- and 3-dimensions for the visualization of any inherent structure. The NLM was demonstrated to be effective at revealing, in 2-dimensions, structure that a (linear) PCA projection could not. Note that the NLM was still just a mapping: the final configuration was reached by iteratively adjusting the positions of the output points in configuration space and no functional transformation was defined on the input data vectors.

A mapping, such as Sammon's NLM, may be of merit for visualizing relationships within multivariate data, but once the map is determined for a finite number of samples

it is far from straightforward to place new, unclassified samples within the derived configuration space. Such mapping approaches are awkward (at best) for use in classification applications, where novel measurements are to be related to previous measurements (exemplars) of known class. Furthermore, a configuration in m-dimensions comprising N points involves Nm adjustable parameters, which rapidly becomes a challenging computational problem as dataset size N increases.

Koontz and Fukunaga [59] defined a criterion that combined both a structure preservation and a class separability term. A low order polynomial function was employed to transform the original distances into distances in configuration space and optimize this criterion. Thus, given any two points in the input space, the distance between their images in the output space could be computed. The problem then became that of finding an output configuration that (best) yielded these distances. In this regard Koontz and Fukunaga's [59] goal was that of metric MDS. Their advance, however, was a function on the input vectors that effected this desired distance transformation. The optimization problem was then one of finding a small number of parameters defining the required transformation function (in contrast to the task of finding Nm coordinates) and, by acting directly on the vectors, rather than the distances between them, the projection of novel measurements was accommodated.

2.5.4 Summary

With origins in fields such as psychology, MDS attempts to represent (usually ordinal) data in (usually) two dimensions. The representation is such that it conveys as much of the similarity between data points as possible: data points that are similar in measurement space lie close together in a two-dimensional plot after MDS.

Early work by Shepard and Kruskal on nonmetric MDS sought to obtain solutions that best preserved monotonicity of rank ordered data, along with a requirement for minimum dimensionality. Clearly, the practical use of MDS for dimensionality reduction of new measurements requires a technique that can work with the data themselves, rather than just measures of similarity between data points. Access to the actual data values then allows metric MDS, wherein quantitative similarities between data points are preserved, rather than monotonic relationships.

Sammon was the first to use metric MDS in the pattern recognition field, but this was an iterative technique that sought a low-dimensional configuration of points that captured (nonlinear) structure in the original data. Like principal curves, Sammon's map did not result in a function that could be used to map future measurements to the derived low-dimensional space, making it of little use as a preprocessor for classification tasks. Koontz and Fukunaga's advance was the derivation of a function on the input data that effected

a metric MDS transformation. A limitation, however, was that the parametric form of the function had to be specified in advance and so could be highly suboptimal. This is in contrast to artificial neural networks, with their data-driven, universal approximation property.

2.6 Radial basis functions

2.6.1 Introduction

Radial Basis Function Networks (RBFNs) can be motivated from several perspectives, including, and originally, function approximation. Generally comprising three layers, with one hidden layer, they can be viewed as a curve-fitting problem in a high-dimensional space [45]. The hidden layer contains the nonlinearity, whereas the output layer is a linear combination of the hidden nodes. As such, RBFNs owe much to the results of Cover [23], who established that a complex pattern classification problem is more likely to be linearly separable if cast nonlinearly into a high-dimensional space. This provides a justification for having a nonlinear hidden layer of high-dimension followed by a linear output layer. This particular architecture can lead to computationally attractive methods of obtaining solutions.

2.6.2 Radial basis functions for interpolation

The problem of exact interpolation is defined [8, 9, 45]: for a given set of N unique points $\{\mathbf{x}_i \in \mathbb{R}^p | i = 1, 2, \ldots, N\}$, and corresponding set of N real numbers $\{d_i \in \mathbb{R}^1 | i = 1, 2, \ldots, N\}$, find a function $F : \mathbb{R}^p \to \mathbb{R}^1$ that satisfies the interpolation condition:

$$F(\mathbf{x}_i) = d_i. \tag{2.13}$$

The surface is thus required to pass exactly through all of the training data points.

Introducing a set of N radial basis functions $\phi(\|\mathbf{x} - \mathbf{x}_i\|)$, each centred on a training data point $\mathbf{x}_i$, results in interpolating functions for F, of the form:

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) \tag{2.14}$$

and inserting the interpolation conditions (2.13) into (2.14), gives the set of linear equa-

tions:

$$\begin{pmatrix} d_1 \\ \vdots \\ d_N \end{pmatrix} = \begin{pmatrix} \phi_{11} & \dots & \phi_{1N} \\ \vdots & \ddots & \vdots \\ \phi_{N1} & \dots & \phi_{NN} \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_N \end{pmatrix}, \tag{2.15}$$

where $\phi_{ij} \triangleq \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$. This has the compact form:

$$\mathbf{d} = \Phi\mathbf{w}. \tag{2.16}$$

Thus, provided the inverse of Φ exists, the exact interpolation problem as specified has the unique solution given by:

$$\mathbf{w} = \Phi^{-1}\mathbf{d}. \tag{2.17}$$

It has been proven by Micchelli [81], with the results being built upon by Powell [92], that, provided the N training data points are distinct, there is a large class of functions ϕ that do, indeed, always yield a non-singular matrix Φ. However, notwithstanding these highly satisfying theoretical results regarding the invertibility of Φ, (2.17) is not without its practical limitations. Exact interpolation is generally undesirable in practice because of the danger of fitting unwanted variations, such as noise, in the data. The cost of having a basis function at every data point scales poorly with N, as the cost of matrix inversion grows $\mathcal{O}(N^3)$ [94].

Broomhead and Lowe [8, 9] introduced a distinction between data points, $\mathbf{x}$, and basis centres, $\boldsymbol{\mu}$ so that the basis centres no longer needed to coincide with the data points. By divorcing the basis centres from the training data points, and having fewer centres, l, than data points, N, the problem becomes overspecified; matrix Φ is no longer square and the previous exact problem becomes one of linear optimization. By defining a sum-of-squares optimization criterion, $\|\Phi\mathbf{w} - \mathbf{d}\|^2$, a unique solution that minimizes this criterion is given by:

$$\mathbf{w} = \Phi^{\dagger}\mathbf{d}, \tag{2.18}$$

where $\Phi^{\dagger}$ denotes the pseudo-inverse of Φ, and can be obtained via Singular Value Decomposition [94].

This discussion can readily be generalized to the map $F : \mathbb{R}^p \to \mathbb{R}^{p'}$, in which case the N distinct data points in $\mathbb{R}^p$ are associated with N vectors $\mathbf{d}_i \in \mathbb{R}^{p'}$, as opposed to N scalars $d_i \in \mathbb{R}^1$. With the further addition of constant offsets $\{w_{0k}\}$, the form of the interpolation functions is:

$$F_k(\mathbf{x}) = w_{0k} + \sum_{j=1}^{l} w_{jk}\phi(\|\mathbf{x} - \boldsymbol{\mu}_j\|), \tag{2.19}$$

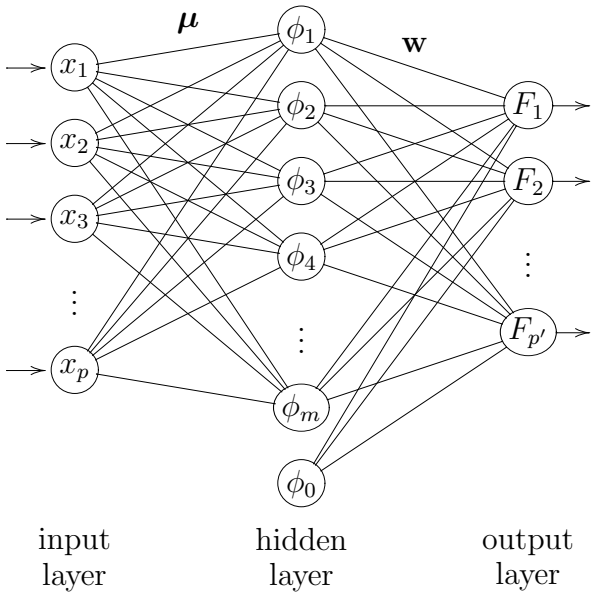


Figure 2.6: *Radial basis function network for implementing (2.19), where hidden node ϕ_j is synonymous with $\phi(\|\mathbf{x} - \boldsymbol{\mu}_j\|)$ and $\phi_0 \triangleq 1$ is associated with the offsets w_{0k}.*

where $\mathbf{x} \in \mathbb{R}^p$ and $k = 1, 2, \ldots, p'$. Importantly for obtaining solutions, the nonlinear transformation $\mathbf{F}(\mathbf{x})$ is formulated in terms of linear algebra.

2.6.3 Radial basis function networks

The form of the expansion, (2.19), as a weighted sum over nonlinear functions suggests a feedforward-network architecture comprising three fully-connected layers, with no interconnections within the layers. This is shown in figure 2.6. From the neural network perspective; the RBF centres, $\boldsymbol{\mu}$, are the input-to-hidden-layer weights; the RBFs, $\boldsymbol{\phi}(.)$, are the (nonlinear) transfer functions at the hidden layer; and the weights, $\mathbf{w}$, are the hidden-to-output-layer weights [71].

Broomhead and Lowe [8, 9] showed that a guaranteed learning rule for RBFNs existed, but did not specify the optimality of the result. Park and Sandberg established that RBFNs having a single hidden layer are capable of universal approximation [88, 89]. Furthermore, it has been shown that RBFNs possess the property of "best approximation" [40], such that within the set of functions arising from all possible combinations of all possible parameter values, there is one function which has minimum approximation error for any given target function [4, 40].

2.6.4 Model selection and optimization

The development of RBFNs following Broomhead and Lowe [8, 9], with a guaranteed linear learning rule, assumes that the hidden-to-output-layer weights are the only parameters to solve for: all other parameters, such as the RBF centres, are fixed. To achieve this, the locations of the RBF centres etc. can be obtained via unsupervised training on the data. Possible methods include random selection from the dataset and random selection followed by clustering, e.g. k-means [4, 127]. The number of centres to use can be determined by, for example, cross-validation and performance on a separate test set [127].

Empirically, this appears to be a satisfactory approach. Broomhead and Lowe [8, 9] observed, for the exact interpolation problem, that the placing of the centres was not critical. Further work, in the classification domain, concluded that the positions of the centres do not appear to be critical provided they are spread throughout the input data space [54]. This conclusion was echoed again, by Lowe, in [69].

The network can be optimized as a whole, but this requires a nonlinear optimization technique, and the attraction of a guaranteed linear learning rule is lost. The usual problems with nonlinear optimizations, such as local minima, then need to be guarded against. A supervised algorithm, whereby Gaussian nodes are iteratively added in the training process, has been described [13] and results suggested a reduced susceptibility to local-minima. However, Lowe concluded that a nonlinear optimization, including the first layer parameters, is only beneficial when a minimal network is required: the same generalization performance can be achieved simply by using more centres and performing linear optimization on the final-layer weights [69].

Regarding the form of the nonlinearity $\phi(.)$, there are many suitable candidates [90], with the Gaussian being a popular choice. Network performance is generally found not to be crucially dependent on the choice made [69, 92]. However, the use of unbounded and nonpositive basis functions has been considered, motivated from the domains of functional interpolation and kernel-based density estimation, with encouraging results [70].

2.6.5 Summary

RBFNs are well motivated for function approximation. They share a common philosophy with SVMs, namely that of projecting a problem nonlinearly and then finding a linear solution. They are relatively insensitive to the placement of the basis-function centres, which need not coincide with data points and can be chosen by an unsupervised method. For a wider development of RBFNs see the relevant chapters in [4, 45] as well as the appropriate section in [127] and the review article by Powell [93].

2.7 Neural networks for multidimensional scaling

Koontz and Fukunaga's [59] parametric transformation of section 2.5.3 was unattractive because the form of the transformation must be pre-defined. Furthermore, the number of parameters to be determined increases rapidly as the specified order of the polynomial increases. The universal-approximation properties of feedforward Artificial Neural Networks (ANNs), with a single hidden layer [27, 48, 49, 109], and their data-driven learning, make them attractive tools for effecting an MDS transformation: if there is a nonlinear function, $\mathbf{f}$, that effects an optimum transformation of the input data to a lower-dimensional space, then ANNs offer the potential of a transformation, $\hat{\mathbf{f}}$, which arbitrarily-accurately approximates $\mathbf{f}$, and they can learn this $\hat{\mathbf{f}}$ from the data.

Webb [121] proposed the use of a MultiLayer Perceptron (MLP) [4, 45] feedforward network to effect a nonmetric MDS transformation. The criterion optimized was Kruskal's stress (2.12), and this was achieved by differentiating the stress with respect to the network parameters and applying a gradient-based learning algorithm. However, the use of stress (2.12) as the optimization criterion for a feedforward network presents a heavy computational burden for very large datasets [121].

An alternative to the MLP ANN, is the Radial Basis Function Network (RBFN) [4, 45, 71]. With their established functional approximation properties, RBFNs are good candidates for learning an MDS dimensionality-reducing transformation and such an approach was described by Webb [122]. The dimension-reducing transformation, $\mathbf{f} : \mathbb{R}^p \rightarrow \mathbb{R}^m$, from p-dimensional input space to m-dimensional output (configuration) space, where $\mathbf{f} = (f_1, f_2, \ldots, f_m)^{\mathrm{T}}$ was taken to be a sum of l basis functions of the form:

$$f_i(\mathbf{x}; \mathbf{W}) = \sum_{j=1}^{l} w_{ji} \phi_j(\|\mathbf{x} - \boldsymbol{\mu}_j\|) \tag{2.20}$$

where $\phi_j, j = 1, \ldots, l$ are the basis functions and $w_{ji}, j = 1, \ldots, l, i = 1, \ldots, m$ are weights. The main difference from an MLP is that the output of a hidden node is a radial function of the distance between the input vectors, $\mathbf{x}$, and the hidden-node weight-vectors, $\boldsymbol{\mu}$, rather than a scalar product [71]. The first-layer weights, $\boldsymbol{\mu}$, can generally be determined from the data by an unsupervised algorithm, and are not required to be found as part of the network optimization process [71]. The major task for training the network then becomes that of the determination of the final-layer weights.

Webb's approach [122] was to define a least-squares loss function:

$$\sigma^2(\mathbf{W}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{ij} (q_{ij} - d_{ij}(\mathbf{X}))^2 \tag{2.21}$$

where $\alpha_{ij}, i, j = 1, \ldots, N$ are positive weights, $d_{ij} = |\mathbf{x}_i - \mathbf{x}_j|$ is the distance (dissimilarity) between the ith and jth input pattern, and $q_{ij} = |\mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_j)|$ is the distance between the ith and jth pattern in the output (configuration) space[1]. By seeking to minimize σ^2 with respect to the weights, $\mathbf{W}$, a nonlinear transformation of the input vectors to a reduced-dimension space was defined, such that the distances between the points in the input space were similar to the distances between the points in the output space. This loss function was optimized via the process of iterative majorization [28, 29, 30, 46, 47]. The distances were calculated using the Euclidean metric, and the RBFs were of Gaussian form.

Extensions of this method have used different forms for the weightings α_{ij} in (2.21) [124], and Minkowski metrics with order parameter $1 \leq p \leq 2$ for the calculation of the distances [125].

2.8 Summary justifying the selected approach

For obtaining reduced-dimensionality descriptions of data, a linear method such as PCA certainly has its attractions. However, globally approximating the distributions with hyperplanes is often inefficient and cannot define arbitrary transformations of the input space. Such transformations may be required to simultaneously cancel out unwanted differences between data belonging to the same class whilst separating data belonging to different classes. The approximating capabilities of ANNs are very appealing for learning transformations that generalize linear techniques via the inclusion of nonlinearities.

Multidimensional Scaling (MDS) is attractive when the goal is feature extraction for classification as MDS deals primarily with dissimilarities. It is dissimilarities between data points that are of interest to a classifier. ANNs are capable of learning a transformation that will effect any desired MDS mapping. In particular, RBFNs can learn arbitrary mappings using only a single hidden-layer; the initial weights can be set *a priori* and do not need to be learned as a part of the optimization process, leading to efficient linear optimization procedures; and RBFNs have the satisfying property of best approximation, meaning there will be a unique optimal solution.

Thus, the use of an RBFN for performing MDS, using a loss function such as (2.21), is herein presented as a promising technique for performing feature extraction prior to classification when the goal of the feature extraction is to reduce the influence of undesirable variability within the data. This approach offers great flexibility within the optimization criterion; for example, the contribution of different data pairs to the solution can be varied, and the calculation of the desired dissimilarities can be modified through the use of

[1] c.f. Kruskal's stress, (2.12).

different distance metrics.

Chapter 3

Radial basis function approach

3.1 Introduction

Chapter 2 presented a progression of techniques for data-driven feature extraction. Such feature extraction takes the form of a mapping of data points to a reduced-dimension space, although SVMs generally perform an implicit projection to a higher-dimension space. The techniques described in that chapter started with simple linear transformations of the data, introduced nonlinear generalizations, and culminated in the application of a Radial Basis Function Network for Multidimensional Scaling as described by Webb [122].

This chapter describes the detail of an approach based on that of Webb [122], as introduced in section 2.7, and modifications to the approach that are novel to this thesis. The data-transformation and, in particular, the method used to optimize its determining parameters are described in section 3.2. The key aspects are that the transformation is defined to be a sum of basis functions, and the network weights are determined by a procedure that does not rely on gradient calculations.

Certain degrees-of-freedom must be fixed to fully specify the algorithm and control its training. The form of the basis functions must be chosen. Both the number and locations of centres must be selected, along with any other basis-function-specific parameters. The optimization of the network weights is an iterative process and so a stopping criterion for deeming the solution to have converged must be specified. These issues are discussed in section 3.2.3.

The algorithm allows considerable flexibility in the specification of the problem to be solved, whilst still utilizing the same basic learning methodology. For example, not all pairs of data points need contribute equally to the solution; changing the relative contributions of the distances between pairs of points that belong to the same class and pairs that belong to different classes will alter the importance of different aspects of the structure being modelled within the data, variously emphasizing intra-class structure or

inter-class structure. Alternatively, instead of adjusting the contributions from certain inter-point distances, the distances themselves can be modified, changing the nature of the relationships between the data points. This will modify the data structure being modelled. Such flexibility allows variations in the specification of the desired relationships between the data points, within the same algorithmic framework. Some such variations are described in section 3.3.

Finally, section 3.4 summarizes the important concepts of this chapter and highlights the modifications to the radial basis function approach that are novel to this thesis.

3.2 Determining the nonlinear transformation

3.2.1 Defining the loss function

The original measurement space is denoted $\mathbb{R}^p$, and the feature space $\mathbb{R}^m$, with $m < p$. The desired transformation can thus be denoted by $\mathbf{f} : \mathbb{R}^p \rightarrow \mathbb{R}^m$. Taking $\mathbf{f} = (f_1, \ldots, f_m)^{\mathrm{T}}$ to be a sum of l basis functions of the form

$$f_i(\mathbf{x}, \mathbf{W}) = \sum_{j=1}^{l} w_{ji}\phi_j(\mathbf{x}) \tag{3.1}$$

where $\phi_j, j = 1, \ldots, l$ are nonlinear functions of $\mathbf{x} \in \mathbb{R}^p$ and $w_{ji}, j = 1, \ldots, l, i = 1, \ldots, m$ are weights to be determined. Thus, the transformations $f_i, i = 1, \ldots, m$ share a common set of basis functions $\phi_j, j = 1, \ldots, l$. Equation 3.1 can be written as

$$\mathbf{f}(\mathbf{x}) = \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) \tag{3.2}$$

where $\mathbf{W}$ is the $l \times m$ matrix $[W]_{ij} = w_{ij}$ and $\boldsymbol{\phi}$ is the vector with jth component $\phi_j(\mathbf{x})$.

Obtaining a solution for $\mathbf{f}$ can thus be regarded as a two-stage process:

1. determination or specification of the basis functions ϕ_i, and
2. determination of the weights $\mathbf{W}$.

Methods for optimizing the weights, $\mathbf{W}$, include:

- variance maximization subject to constraints [123, 126], and
- stress minimization [122, 125].

The second method is the one adopted here, which requires the definition of a suitable stress function.

Let $\mathbf{X}$ represent the $N \times p$ multivariate data matrix of N data samples (the rows) and p variables (the columns). The sample vectors (rows) are denoted by $\{\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^p\}$. Denoting the distance (dissimilarity) between the ith and jth data vectors in input space ($\mathbb{R}^p$) by d_{ij}, this can be expressed as

$$d_{ij}(\mathbf{X}) = |\mathbf{x}_i - \mathbf{x}_j|. \tag{3.3}$$

Similarly, the distance (dissimilarity) between those points in feature space ($\mathbb{R}^m$), can be denoted by q_{ij}, and can be expressed as

$$\begin{aligned} q_{ij}(\mathbf{W}) &= |\mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_j)| && (3.4) \\ &= |\mathbf{W}^{\mathrm{T}}(\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}(\mathbf{x}_j))|. && (3.5) \end{aligned}$$

Thus, a least-squares loss function can be defined:

$$\sigma^2(\mathbf{W}) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_{ij}(q_{ij}(\mathbf{W}) - d_{ij}(\mathbf{X}))^2}{\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^2}, \tag{3.6}$$

where $\alpha_{ij}, i, j = 1, \ldots, N$ are positive weights. The denominator renders the stress dimensionless [62, 63]. The task, then, is one of minimizing σ with respect to the weights $\mathbf{W}$.

3.2.2 Iterative majorization

The method used here for minimizing σ is iterative majorization [28, 29, 30, 46]. This method derives a majorizing function that is everywhere greater-than or equal-to the function being minimized. Iteratively minimizing the majorizing function leads towards a solution that also minimizes the desired function.

The numerator of the loss, or stress, function (3.6) can be expanded into the three terms:

$$\sigma^2(\mathbf{W}) = \sum_i \sum_j \alpha_{ij} q_{ij}^2(\mathbf{W}) + \sum_i \sum_j \alpha_{ij} d_{ij}^2(\mathbf{X}) - 2\sum_i \sum_j \alpha_{ij} q_{ij}(\mathbf{W}) d_{ij}(\mathbf{X}), \tag{3.7}$$

where $q_{ij}^2(\mathbf{W}) = (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)$ is quadratic in $\mathbf{W}$ and $\boldsymbol{\phi}_i$ denotes $\boldsymbol{\phi}(\mathbf{x}_i)$.

Dealing with the first term, the summation of $q_{ij}^2(\mathbf{W})$, and introducing the matrix $\mathbf{A} = \sum_i \sum_j \alpha_{ij}(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^{\mathrm{T}}$, it can be written:

$$\sum_i \sum_j \alpha_{ij} q_{ij}^2(\mathbf{W}) = \mathrm{Tr}\{\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{W}\} \tag{3.8}$$

The second term in (3.7), the summation of $d_{ij}^2(\mathbf{X})$, is independent of $\mathbf{W}$. It is the third, cross-product, term in (3.7):

$$\sum_i \sum_j \alpha_{ij} q_{ij}(\mathbf{W}) d_{ij}(\mathbf{X}) \tag{3.9}$$

that facilitates the procedure of iterative majorization. The explanation is as follows.

The Cauchy-Schwartz inequality states:

$$|\mathbf{w}||\mathbf{v}| \geq \mathbf{w}^{\mathrm{T}}\mathbf{v}. \tag{3.10}$$

Let

$$\mathbf{w} = \mathbf{W}^{\mathrm{T}}(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j), \tag{3.11}$$

and

$$\mathbf{v} = \mathbf{V}^{\mathrm{T}}(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j), \tag{3.12}$$

where $\mathbf{V}$ are weights like $\mathbf{W}$. Substituting (3.11) and (3.12) into (3.10) and using the expansion of $q_{ij}(\mathbf{W})$ from (3.5) yields:

$$q_{ij}(\mathbf{W}) q_{ij}(\mathbf{V}) \geq (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^{\mathrm{T}} \mathbf{W}\mathbf{V}^{\mathrm{T}} (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j). \tag{3.13}$$

The third term (3.9) may itself be split into the sum of two, following Heiser [46]:

$$\sum_i \sum_j \alpha_{ij} q_{ij}(\mathbf{W}) d_{ij}(\mathbf{X}) = \sum_{(i,j)\in S_+} \alpha_{ij} q_{ij}(\mathbf{W}) d_{ij}(\mathbf{X}) + \sum_{(i,j)\in S_0} \alpha_{ij} q_{ij}(\mathbf{W}) d_{ij}(\mathbf{X}), \tag{3.14}$$

where S_+ is the set of all pairs of patterns (i, j) for which $q_{ij}(\mathbf{V}) > 0$, and S_0 the set for which $q_{ij}(\mathbf{V}) = 0$.

Considering the first sum on the right hand side of (3.14), for which $q_{ij}(\mathbf{V})$ is nonzero. From (3.13), for $(i, j) \in S_+$:

$$\begin{aligned} \alpha_{ij} q_{ij}(\mathbf{W}) d_{ij}(\mathbf{X}) &\geq \alpha_{ij} \frac{d_{ij}(\mathbf{X})}{q_{ij}(\mathbf{V})} (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^{\mathrm{T}} \mathbf{W}\mathbf{V}^{\mathrm{T}} (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j) & (3.15) \\ &\geq c_{ij}(\mathbf{V}) (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^{\mathrm{T}} \mathbf{W}\mathbf{V}^{\mathrm{T}} (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j), & (3.16) \end{aligned}$$

where $c_{ij}(\mathbf{V}) = \alpha_{ij} d_{ij}(\mathbf{X}) / q_{ij}(\mathbf{V})$. For the second term of (3.14), for which $q_{ij}(\mathbf{V}) = 0$, it can simply be stated:

$$\alpha_{ij} q_{ij}(\mathbf{W}) d_{ij}(\mathbf{X}) \geq 0 \tag{3.17}$$

due to the non-negativity of the distances q_{ij} and d_{ij} and α_{ij} being positive by definition.

Therefore, $c_{ij}(\mathbf{V})$ may be defined as:

$$c_{ij}(\mathbf{V}) = \begin{cases} \alpha_{ij} d_{ij}(\mathbf{X})/q_{ij}(\mathbf{V}) & (i,j) \in S_+ \\ 0 & (i,j) \in S_0 \end{cases}$$

This definition of c_{ij} leads to the general expression, for all (i, j):

$$\alpha_{ij} q_{ij}(\mathbf{W}) d_{ij}(\mathbf{X}) \geq c_{ij}(\mathbf{V})(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^{\mathrm{T}} \mathbf{W}\mathbf{V}^{\mathrm{T}}(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j) \tag{3.18}$$

As (3.18) gives the inequality for each ijth element, the inequality must also apply to the sum. Introducing the matrix $\mathbf{D}(\mathbf{V}) = \sum_i \sum_j c_{ij}(\mathbf{V})(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^{\mathrm{T}}$, rearranging the right hand side of (3.18), summing over (i, j), and substituting in $\mathbf{D}(\mathbf{V})$ yields:

$$\sum_i \sum_j \alpha_{ij} q_{ij}(\mathbf{W}) d_{ij}(\mathbf{X}) \geq \mathrm{Tr}\{\mathbf{V}^{\mathrm{T}} \mathbf{D}(\mathbf{V}) \mathbf{W}\} \tag{3.19}$$

The inequality of (3.19), when applied in the context of the stress as given in (3.7), gives a bound on the loss function such that $\sigma^2(\mathbf{W}) \leq \sigma_m^2(\mathbf{W}, \mathbf{V})$, where (bringing in (3.7) and (3.8)):

$$\sigma_m^2(\mathbf{W}, \mathbf{V}) = \mathrm{Tr}\{\mathbf{W}^{\mathrm{T}} \mathbf{A} \mathbf{W}\} + \sum_i \sum_j \alpha_{ij} d_{ij}^2(\mathbf{X}) - 2\mathrm{Tr}\{\mathbf{V}^{\mathrm{T}} \mathbf{D}(\mathbf{V}) \mathbf{W}\}. \tag{3.20}$$

The quadratic $\sigma_m^2(\mathbf{W}, \mathbf{V})$ is the desired majorizing function and has the property that the equality holds when $\mathbf{V} = \mathbf{W}$, that is $\sigma^2(\mathbf{W}) = \sigma_m^2(\mathbf{W}, \mathbf{W})$. Crucially, from differentiating (3.20) with respect to $\mathbf{W}$ and equating to zero, the solution for $\mathbf{W}$ that minimizes $\sigma_m^2(\mathbf{W}, \mathbf{V})$ satisfies

$$\mathbf{A}\mathbf{W} = \mathbf{D}(\mathbf{V})\mathbf{V}. \tag{3.21}$$

This leads to the following straightforward steps for the minimization procedure:

1. Initialize $\mathbf{W}$. This was done uniformly over the interval $[-1, 1]$.
2. Set $\mathbf{V} = \mathbf{W}$
3. Update $\mathbf{W}$:
$$\mathbf{W} \leftarrow \underset{\mathbf{W}}{\mathrm{argmax}}\ \sigma_m^2(\mathbf{W}, \mathbf{V})$$
4. Check for convergence. If convergence not satisfied, return to step 2.

The update in step 3 minimizes the majorizing function σ_m^2 and thereby leads to a decrease

in the value of σ^2 because:

$$\sigma^2(\mathbf{W}) \leq \sigma_m^2(\mathbf{W}, \mathbf{V}) \leq \sigma_m^2(\mathbf{V}, \mathbf{V}) = \sigma^2(\mathbf{V}).$$

The first inequality follows from the construction of the majorizing function, the second inequality follows from the minimization of $\sigma_m^2(\mathbf{W}, \mathbf{V})$, and the equality follows again from the construction of the majorizing function.

Thus, the stress function, σ^2, is minimized by proxy, without requiring the calculation of its local gradient.

3.2.3 Model selection

One of the first aspects of the RBF model that must be selected is the form of the basis functions $\phi_i(\mathbf{x})$. Gaussian basis functions,

$$\phi_i(\mathbf{x}) = \exp\left\{-\frac{|\mathbf{x} - \mathbf{c}_i|^2}{h^2}\right\}, \tag{3.22}$$

are a popular choice [4, 70], although the case has been made [70, 129] for the benefits of unbounded, non-positive-definite basis functions such as $\phi(z) = z^2 \log(z)$. For this work, we stay with the Gaussian form (3.22). This leaves centres $\mathbf{c}_i$ and the bandwidth h to be determined.

For a given number of centres, there are several methods commonly used for selecting their locations [4]. The method adopted here is that of randomly selecting the desired number of points from the input data and then optimizing their positions through a k-means clustering algorithm to ensure they best represent the distribution of the data. This is essentially an unsupervised method for setting the input-to-hidden-layer weights.

The number of points to use as centres is more an issue of model order selection [4] than an explicit step in the optimization of the RBF network. The number was obtained here by training a series of networks using a range of values and choosing that number that lead to a small value of stress for a separate validation group of data. The same strategy was adopted for determining a suitable basis-function bandwidth h.

During training, the network was defined to have converged to a solution after each iteration when the relative change in the stress fell below one part in 10^4.

3.3 Customizing the projection

Within the basic framework so far outlined (formulation of the stress, optimization via iterative majorization etc.) there are a variety of customizations that can modify the

behaviour of the RBFN. This section outlines those used in this work.

3.3.1 Modifying the weights α_{ij} in the loss function

One aspect of specifying the model that has yet to be discussed is that of the positive weights α in (3.6). These parameters, like the number of centres, have to be specified before a network can be trained. A simple implementation is just to set $\alpha_{ij} = 1 \, \forall i, j$. In this way all distances between pairs of points contribute equally to the stress. As far as class membership of the points is concerned, the technique is unsupervised: it captures wholesale data structure, irrespective of the class labels of the points.

The formulation of the stress in (3.6) can be made sensitive to the class labels of each pair of points (i, j) via α_{ij}:

$$\alpha_{ij} = \begin{cases} \rho & \omega_i = \omega_j \\ (1 - \rho) & \omega_i \neq \omega_j \end{cases} , \tag{3.23}$$

where $0 \leq \rho \leq 1$ and ω_i is the class label of point i. Thus, varying the parameter ρ varies the degree of inter-class and intra-class structure captured: $\rho = 0$ means distances between points belonging to the same class contribute nothing to the stress, $\rho = 1$ means that only distances between points belonging to the same class contribute, values of ρ in-between lead to superpositions of the cases.

3.3.2 Modifying the calculated distances d_{ij}

Rather than differentially weight the contributions of distances to the stress, another approach is to directly modify the distances themselves. After all, the distances between points, as calculated using the Euclidean metric, have only really been used as estimates of the dissimilarities between data samples. For feature extraction as applied to the task of discrimination rather than discovering data-structure, and where labelled training data are available, the projection might be better trained by providing it with dissimilarities that better reflect *a priori* knowledge of the relationships between samples.

The method adopted here is similar to that described by Cox and Ferry [24]. Cox and Ferry applied a scaling factor $\gamma_r \geq 1.0$ to inter-class dissimilarities, leaving intra-class dissimilarities unchanged. These dissimilarities were then input into a non-metric MDS algorithm which sought to preserve the rank-ordered dissimilarities. Their choice of value for γ_r was *ad hoc* and a range of values were tried in the search for optimum class separability. Cox and Ferry did offer a recipe for calculating a minimum value of γ_r required to achieve separability, but it appeared not to be used. This was possibly borne of the desire to avoid degenerate solutions [24], although this was not made clear.

Here we modify the inter-point distances d_{ij}, initially calculated as before (3.3), according to:

$$d_{ij} \leftarrow \begin{cases} \gamma_r d_{ij} & \omega_i \neq \omega_j \\ d_{ij} & \omega_i = \omega_j \end{cases}, \tag{3.24}$$

where, again ω_i is the class label of point i, $\gamma_r = d_{Smax}/d_{Bmin}$, d_{Smax} is the largest intra-class distance, and d_{Bmin} is the smallest inter-class distance. In this manner, the input distances are transformed such that the smallest inter-class distance is equal to the largest intra-class distance, thus teaching the RBFN that samples belonging to different classes should be regarded as more dissimilar to one another than samples belonging to the same class.

3.3.3 Box-Cox: an alternative distance metric

Instead of calculating the distances according to the Euclidean metric

$$d_{ij}^2 = \left\langle (\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}, (\mathbf{x}_i - \mathbf{x}_j) \right\rangle, \tag{3.25}$$

a different distance metric may be used. The Box-Cox metric [6, 96] has previously been found to be effective for radar data [66, 116, 117, 118]. Specifically, the metric comprises the Box-Cox transformation:

$$x^{(t)} = \begin{cases} (x^t - 1)/t & 0 < t \leq 1 \\ \log x & t = 0 \end{cases} \tag{3.26}$$

followed by the Euclidean distance metric:

$$d_{ij}^2 = \left\langle (\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)})^{\mathrm{T}}, (\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}) \right\rangle. \tag{3.27}$$

t is thus a parameter that shifts the transformation (3.26) between linear ($t = 1$) and log-scale ($t = 0$). For $t = 1$ the Euclidean metric is recovered. Work on classifying radar range profiles of aircraft found $t \approx 0.1$ to be optimal [116, 117, 118], and work on classifying ISAR images of military vehicles found $t = 0$ to be optimal [66]. For the present work on ISAR images of military vehicles we will thus use $t = 0$, i.e. log-scaling the data.

It is worthwhile here to clarify the role of this log-scaling in the dimension-reducing projection. It is only applied as part of the calculation of the d_{ij}. The RBFN is still trained on the non-log-scaled data, as before, only now it seeks to capture structure as defined by the Box-Cox metric (with $t = 0$), rather than the Euclidean.

This leads to a comparison between learning the Box-Cox transformation as part of the nonlinear feature extraction and performing feature extraction after the Box-Cox

transformation has been explicitly applied to the data. The latter case can be thought of as applying the Box-Cox transformation to data as a pre-processing step prior to the RBFN stage. In which case, the transformation is applied globally to all data-space, $\mathbb{R}^p$:

$$\mathbf{x} \leftarrow \log \mathbf{x} \quad \forall \mathbf{x} \in \mathbb{R}^p. \tag{3.28}$$

The RBFN no longer attempts to learn the Box-Cox transformation as part of the data projection, instead it simply attempts to capture the Euclidean structure in data that resides in a data-space already transformed by (3.28).

3.4 Summary

This chapter presented the detailed theory behind the RBFN and its optimization. The goal of the RBFN was to perform MDS. A novel contribution of this thesis is the application of this RBFN to perform feature extraction on radar data prior to classification. Further novel contributions are modifications to the training of the RBFN to effect features that are optimized for different goals:

- the inter-point distances are weighted differently, depending upon whether a pair of points belong to the same class or different classes;
- the inter-point distances are increased by a scaling factor if a pair of points belong to different classes, this is motivated by Cox and Ferry's work [24] but is novel in its application in this context;
- the inter-point distances are calculated using the Box-Cox metric as opposed to the Euclidean, so the RBFN is trained to implicitly perform the Box-Cox transformation on the data.

MDS models dissimilarities and classification depends upon dissimilarities between data samples. Thus, the modifications to the training of the RBFN were targeted at either the calculation of the dissimilarities or the importance assigned to particular dissimilarities. Changing the weighting of the distances in the optimization of the network attempts to teach the network to concentrate on features that represent what a particular class is, or on features that represent the differences between classes, or a combination of the two. Rescaling the inter-class distances attempts to teach the network the intuitive notion that two images of vehicles from different classes should be more different, whatever the aspect angle, than two images of vehicles from the same class. Training the RBFN using Box-Cox distances attempts to teach the network to base its idea of dissimilarity on the Box-Cox distance rather than the Euclidean distance.

Chapter 4

Experimental design

4.1 Introduction

This chapter outlines the goal of the research, the data used, the experiments performed, and the criterion used for assessing the utility of a particular approach. Pre-processing of the radar images is also discussed.

Section 4.2 outlines the goal of the research. Included in this section is the motivation for adopting the approach taken. Section 4.3 then describes the radar data used for this work. An important aspect of this research is that it has been conducted on a rich and diverse set of real-world radar data and the data used for testing are always from a set of observations different from those used for training. Crucially, this allows the testing of classifier performance under conditions that differ from those present during classifier training. Section 4.4 presents an outline of the experiments described in this thesis and section 4.5 then details the steps in the experimental method.

4.2 Aim of the research

The aim of this research was to investigate a technique for data-driven feature extraction. Specifically, the desire was to obtain features that gave improved classification performance when the test conditions deviated from the training conditions. The application domain was that of radar images of military vehicles. Deviations in operating conditions comprised testing the classifier on vehicles that were present during training but with configurational changes, and vehicles that were not present during training but still belonged to a defined class of the training data. As well as looking for improvements in classification performance across configuration changes and vehicles, the classification performance across such differing operating conditions will help to answer the question "How hard is the problem?" Because of the tendency of researchers to use i.i.d. data, this

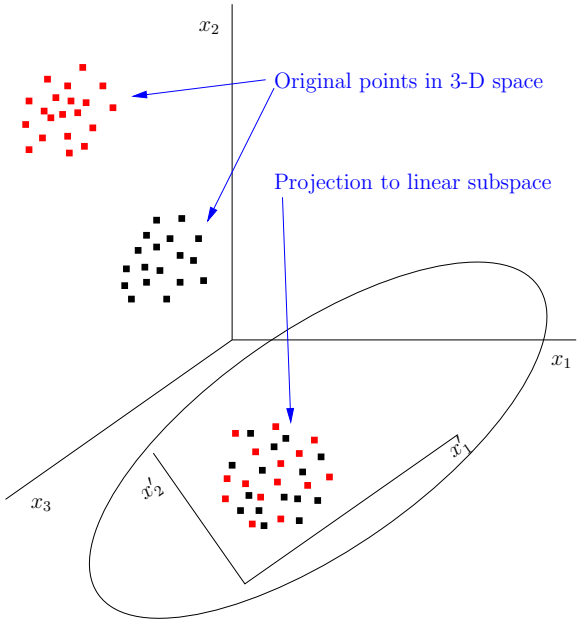


Figure 4.1: *Example illustrating two groups of points (red and black) in 3-D space denoted by* $\mathbf{x} = [x_1, x_2, x_3]^T$. *The two groups belong to the same class and a simple linear projection to a 2-D subspace (inset ellipse) denoted by* $\mathbf{x}' = [x'_1, x'_2]^T$ *projects out the unwanted variance between the groups.*

question in this application domain is unanswered.

The motivation for approaching the problem from a dimension-reducing feature extraction perspective is illustrated in the following figures. Figure 4.1 represents two data clusters in 3-D space. The two clusters belong to the same defined class, yet are separated. A simple linear projection can be seen to remove the unwanted difference between the two clusters. The addition of a third data set belonging to a different class, shown in green in figure 4.2, creates a situation where the simple linear projection can no longer remove the unwanted variability without also projecting the different classes onto the same region of subspace. Only a nonlinear projection can simultaneously separate the two classes whilst removing the unwanted difference between the red and black data points.

4.3 Inverse synthetic aperture radar data

4.3.1 Description

The data used for this work were inverse synthetic aperture radar (ISAR) images of military vehicles. By measuring a vehicle as it rotates on a turntable in front of a radar, a full 360°-worth of images can rapidly be constructed. The radar was operated using circularly polarized radiation at a centre frequency of 94 GHz. One of the 'odd-bounce' polarization (trihedral) channels was used to construct the radar images. The images were

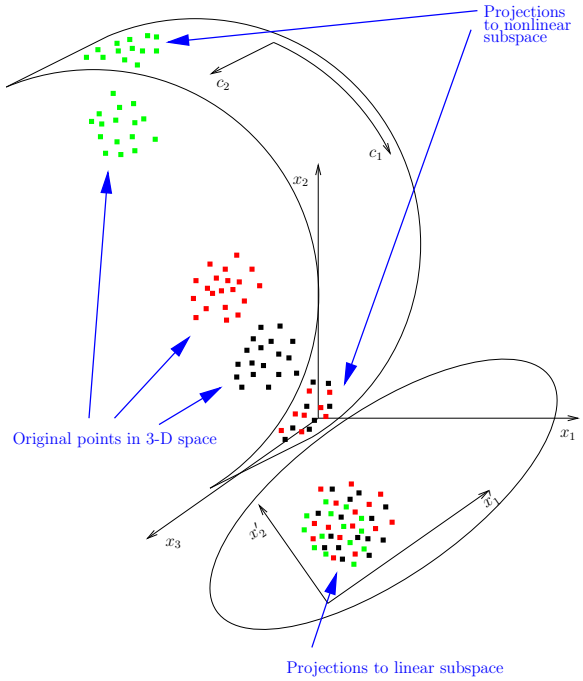


Figure 4.2: *Example illustrating the addition of a new group of points (green). The green points belong to a different group than the red and black points. The linear projection (inset ellipse) collapses the green points onto the red and black points. A nonlinear projection to a 2-D subspace (inset curved space) denoted by* $\mathbf{c} = [c_1, c_2]^T$ *is required to merge both the red and black points* and *keep the green points separate.*

processed to a nominal resolution of 30 cm down-range and 35 cm cross-range. Unless otherwise stated, all vehicles were imaged at a radar depression angle (RDA) of 10°.

The vehicles belonged to three basic classes: Main Battle Tank (MBT), Armoured Personnel Carrier (APC), and Air Defence Unit (ADU). It was at this level that the class labels were assigned; successful classification would be the correct designation of a vehicle as an MBT, for example, rather than to identify the specific variety of MBT. Pictures of most of the vehicles used in this study are provided in appendix A.

A sample ISAR image of an MBT is shown in figure 4.3. The main body of the vehicle extends down-range to approximately 35–40 on that axis. Because of self-shadowing, most of the responses from the main body of the vehicle were from the lower-right quadrant, with the upper-left being undiscernable from the background level. Typical of radar imagery, a small number of the pixels were the brightest by far. Approximately four of these dominant scatterers, coloured red through yellow, are visible in this image. Towards the bottom of the image, protruding from the main body of the vehicle, the gun barrel can be seen. As the vehicle rotates, however, the barrel is rarely seen except mostly from aspect angles close to head-on, as was the case here.

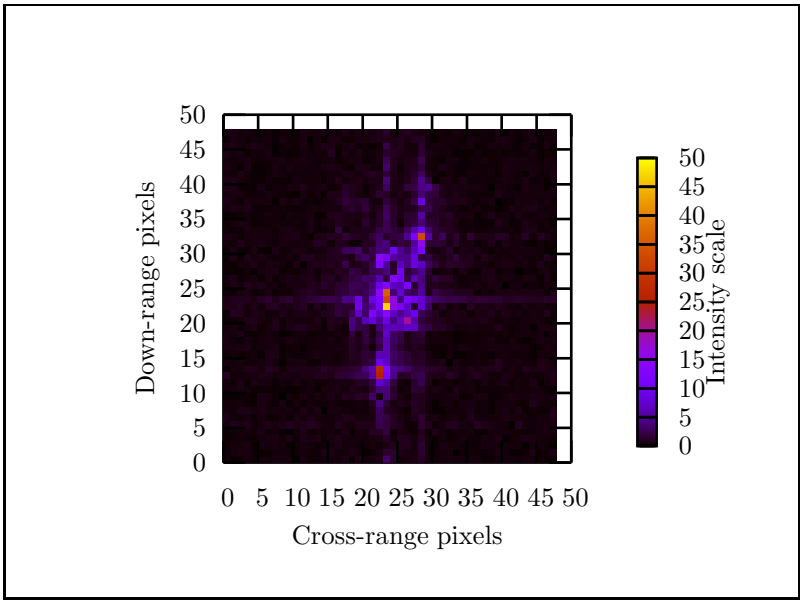


Figure 4.3: *Sample ISAR image of MBT vehicle.*

Table 4.1: *Small training data set. Unless specified otherwise, all hatches and toolboxes were closed with gun pointing forwards (0°). Numbers of unaveraged and averaged images, respectively, are given in parentheses.*

Vehicle name	Vehicle label	Comments
T-62	MBT1-1	Snorkel down. (1074, 72)
BMP	APC1-1	(1244, 72)
ZSU-23-4	ADU1-1	Radar up. (1360, 72)

4.3.2 Partitioning

As a major interest of this research was in cases where the test data were not well represented by the training data, the test data were sub-partitioned to graduate the degree by which they differed from the training data. A small training set was formed, where each class was represented by a single vehicle: MBT1-1 represented the MBT class, APC1-1 represented the APC class, and ADU1-1 represented the ADU class. This group is listed in table 4.1. The vehicle labels code for vehicle class, vehicle number, and rotation number in the format *Cm-n*, where *C* denotes the class ('MBT', 'APC', or 'ADU'), *m* the vehicle number, and *n* the rotation number, e.g. MBT1-1 and MBT1-2 refer to two different rotations of the same vehicle (MBT1). Different rotations of a vehicle generally represent different configurations of the vehicle. Likewise, a small validation set was created for selecting model parameters. This small validation set comprised the individual data sets MBT1-8, APC1-4, and ADU1-8, as listed in table 4.2.

The test data sets were assigned to one of five groups. Broadly, a group either comprised only vehicles used in training (MBT1, APC1, and ADU1) or none of the vehicles

Table 4.2: *Small validation data set. Unless specified otherwise, all hatches and toolboxes were closed with gun pointing forwards (0°). Numbers of unaveraged and averaged images, respectively, are given in parentheses.*

Vehicle name	Vehicle label	Comments
T-62	MBT1-8	Snorkel down, hatch 1 opened, toolboxes 2 and 3 opened. (1083, 72)
BMP	APC1-4	Gun at 180° azimuth. (1213, 72)
ZSU-23-4	ADU1-8	Radar up, gun at 180° azimuth. (1111, 71)

used in training. There were then further differences between the groups in terms of the degree of configuration-change relative to the training data. The five groups used for testing were:

Group 1 contained the exact same vehicles present in the small training set, but with minor configuration changes, such as a hatch opened or closed. A summary of this group is given in table 4.3;

Group 2 contained the exact same vehicles present in the small training set but with greater configuration changes, such as a 90° turret rotation. A summary of this group is given in table 4.4;

Group 3 contained vehicles not present in the small training set. A summary of this group is given in table 4.5;

Group 4 contained vehicles not present in the small training set combined with greater configuration changes beyond those present in the previous group. A summary of this group is given in table 4.6;

Group 5 contained datasets of two vehicles imaged over a range of different RDAs. Neither of the vehicles were represented in the training data, at any RDA. A summary of this group is given in table 4.7.

In terms of configuration changes relative to the training data, Groups 3 and 4 are thus rough analogues of Groups 1 and 2 for vehicles not present in the small training set. The sample size for each data set is greater than 1000. Thus, when contemplating the error rate for a data set, a decrease of 1 percentage point means the correct classification of at least a further 10 test samples that were previously misclassified.

To investigate the effect of representing a wider variety of targets or conditions during training, additional data sets were appended to the small training and validation groups for some experiments. New vehicles that were added to the training data were MBT2 (MBT2-1) and APC2 (APC2-1), but limited members of the ADU class lead to another

Table 4.3: *Group 1 test data set. Unless specified otherwise, all hatches and toolboxes were closed with gun pointing forwards (0°). Numbers of unaveraged and averaged images, respectively, are given in parentheses.*

Vehicle name	Vehicle label	Comments
T-62	MBT1-3	Snorkel down, toolbox 1 opened. (1085, 72)
T-62	MBT1-4	Snorkel down, toolboxes opened. (1068, 72)
T-62	MBT1-5	Snorkel down, hatches opened. (1084, 72)
T-62	MBT1-7	Snorkel down, engine running. (1102, 72)
BMP	APC1-2	Hatches opened. (1257, 72)
ZSU-23-4	ADU1-3	Radar up, gun at 45° azimuth. (1120, 72)
ZSU-23-4	ADU1-5	Radar up, gun at 45° elevation. (1104, 72)

Table 4.4: *Group 2 test data set. Unless specified otherwise, all hatches and toolboxes were closed with gun pointing forwards (0°). Numbers of unaveraged and averaged images, respectively, are given in parentheses.*

Vehicle name	Vehicle label	Comments
T-62	MBT1-2	Snorkel up, hatches opened, toolboxes 2 & 3 opened. (1114, 72)
T-62	MBT1-6	Snorkel down, Russian cam netting draped over. (1054, 72)
BMP	APC1-3	Gun at 90° azimuth. (1206, 72)
ZSU-23-4	ADU1-4	Radar up, gun at 90° azimuth. (1116, 72)
ZSU-23-4	ADU1-6	Radar down, engine running. (1131, 72)
ZSU-23-4	ADU1-7	Radar down, engine running, hatches opened. (1095, 71)

Table 4.5: *Group 3 test data set. Unless specified otherwise, all hatches and toolboxes were closed with gun pointing forwards (0°). Numbers of unaveraged and averaged images, respectively, are given in parentheses.*

Vehicle name	Vehicle label	Comments
T-55	MBT2-2	Hatches opened. (1198, 72)
T-55	MBT2-3	Toolboxes opened. (1072, 72)
T-72	MBT3-3	(1341, 72)
T-72	MBT3-6	Engine running. (1335, 71)
432	APC2-2	Engine running. (1220, 72)
432	APC2-3	Engine running, hatches opened. (1279, 72)
Challenger 1	MBT4-3	Reflectors either side of vehicle. (1335, 72)
(classified)	MBT5-3	Hatches opened, engine running. (1254, 72)
Stormer	ADU2-1	(1276, 72)
ZSU-23-4	ADU3-1	(1188, 72)

Table 4.6: *Group 4 test data set. Unless specified otherwise, all hatches and toolboxes were closed with gun pointing forwards (0°). Numbers of unaveraged and averaged images, respectively, are given in parentheses.*

Vehicle name	Vehicle label	Comments
T-55	MBT2-4	Gun at 45° azimuth. (1078, 72)
T-55	MBT2-5	Gun at 180° azimuth. (1132, 72)
T-55	MBT2-6	Gun at 270° azimuth. (1076, 72)
Challenger 1	MBT4-1	Gun at 180° azimuth in crutch, hatches opened. (1311, 72)
Challenger 1	MBT4-2	Gun at 90° azimuth, hatches opened, reflectors either side of vehicle. (1334, 72)
(classified)	MBT5-4	Gun at 90° azimuth. (1275, 72)
(classified)	MBT5-5	Gun at 180° azimuth (transit configuration), engine running. (1235, 72)
(classified)	MBT5-6	Gun at 180° azimuth (transit configuration). (1197, 72)
Challenger 1	MBT4-4	Gun at 180° azimuth, turret under cover. (1269, 72)
ZSU-23-4	ADU3-2	Gun at 90° azimuth. (1242, 72)
ZSU-23-4	ADU3-3	Gun at 90° azimuth, hatches opened. (1243, 72)
ZSU-23-4	ADU3-4	Gun at 180° azimuth, hatches opened. (1263, 72)
ZSU-23-4	ADU3-5	Gun at 270° azimuth, reflectors either side of vehicle. (1253, 72)
T-72	MBT3-7	Gun at 15° azimuth, engine running, reflectors either side of vehicle. (1247, 72)
T-72	MBT3-8	Gun at 15° azimuth, engine running, reflectors either side of vehicle, hatches opened. (1141, 72)
T-72	MBT3-9	Gun at 90° azimuth, engine running, reflectors either side of vehicle. (1153, 72)
T-72	MBT3-10	Gun at 180° azimuth, engine running, reflectors either side of vehicle. (1160, 72)

Table 4.7: *Group 5 test data set. Unless specified otherwise, all hatches and toolboxes were closed with gun pointing forwards (0°). This grouping exhibited varying radar depression angles (RDA). Numbers of unaveraged and averaged images, respectively, are given in parentheses.*

Vehicle name	Vehicle label	Comments
T-72	MBT3-1	12° RDA. (1150, 72)
T-72	MBT3-2	11° RDA. (1340, 72)
T-72	MBT3-4	9° RDA. (1337, 72)
T-72	MBT3-5	8° RDA. (1328, 72)
(classified)	MBT5-1	12° RDA. (1154, 72)
(classified)	MBT5-2	8° RDA. (1135, 72)

Table 4.8: *Extra training data set. Unless specified otherwise, all hatches and toolboxes were closed with gun pointing forwards (0°). Numbers of unaveraged and averaged images, respectively, are given in parentheses.*

Vehicle name	Vehicle label	Comments
T-55	MBT2-1	(1103, 72)
ZSU-23-4	ADU1-2	Radar down. (1058, 72)
432	APC2-1	Hatches open. (1179, 72)

Table 4.9: *Extra validation data set. Unless specified otherwise, all hatches and toolboxes were closed with gun pointing forwards (0°). Numbers of unaveraged and averaged images, respectively, are given in parentheses.*

Vehicle name	Vehicle label	Comments
T-55	MBT2-7	Driver's hatch opened, engine running. (1069, 72)
T-72	MBT3-11	(1340, 72)
(classified)	MBT5-7	Gun at 180° azimuth. (1162, 72)

example of ADU1 (ADU1-2) being used. These additions to the training set are listed in table 4.8. Similarly, additional data sets for augmenting the small validation data set were MBT2-7, MBT3-11, and MBT5-7, as listed in table 4.9. The extended validation data set was used to perform model selection when the extended training data set was used for training.

4.3.3 Pre-processing

Normalization of the image pixels

Absolute calibration of the radar returns from targets is not something that can be relied upon. Battlefield smoke and rain are both factors that contribute to an uncertain attenuation of the radar signal strength. To correct for variations in radar power, a consistent process of normalizing the image pixels to unit sum was applied to all the radar images.

This was applied to all data sets.

Centering the vehicle within the image

In real-world applications a radar would typically image a wide area of ground when searching for a target. When a possible target is found, further processing would be necessary to confirm the identity of the target. The goal will be to classify the target, not the background clutter, and so the pixels associated only with the target need to be extracted from the image. A small region containing these target pixels would then be extracted from the scene for further processing, including classification. The position of the target vehicle will be subject to uncertainty within the extracted image region; the

nominal centre of the vehicle could be a little to the left, right, up, or down of the centre of the image region. In particular, bright returns from the leading edges of the target and dim returns from the far edges of the target result in a non-trivial relationship between the centre of the extracted pixels and the physical centre of the target.

For ISAR turntable measurements, the position of the turntable-centre relative to the radar can be accurately measured. However, the ISAR processing may yield an image containing either an odd or even number of cross-range pixels because of variations in the rotation rate of the turntable. This means that extracting a region out of the ISAR image can result in the cross-range position of the centre of the turntable differing by ± 1 pixel between datasets.

A further complication is that the physical centre of the vehicle may not coincide with the centre of the turntable. The relationship between the two depends upon how the vehicle was driven onto the turntable. The turntable had to be well balanced to distribute the weight evenly over the bearings. The relative position of the centre of gravity to the physical centre of the vehicle varies from one vehicle to another. An allowance therefore must be made for a bias in the position of each vehicle upon the turntable. Failure to do so may unfairly boost same-vehicle classification results because of the vehicles' positions being optimally aligned within the image regions.

Thus, the extracted image regions were subjected to a recentering algorithm. This addressed uncertainty over the exact position of a vehicle within an image region and removed any bias arising from inadvertant alignment of vehicles between images of the same vehicle. A characteristic of images from turntable ISAR measurements is that the image pixels surrounding the turntable are dim and attributable only to noise. A simple but effective scheme for segmenting the target from this background was to threshold the pixels at $\mu + 2\sigma$, where μ is the mean pixel value and σ the standard deviation. This yielded a binary image of the target and the extracted image-region was shifted so the centre of mass of the binary image was centred within the image region. This determined the amount of shifting that was then applied to the greyscale target-images. The shifting was cyclical, with pixels wrapping around if they were shifted off the edge of the region. This was acceptable because these pixels were merely random background noise containing no other structure.

After this shift was applied, the location of the vehicle within the extracted image region should be consistent with that obtained within a real-world scenario. In such scenarios there are no reliable measurements of the location of the target and its position must be estimated from the detected pixels. Importantly for the present context of turntable ISAR measurements, such shifting eliminates any systematic bias in the position of vehicles.

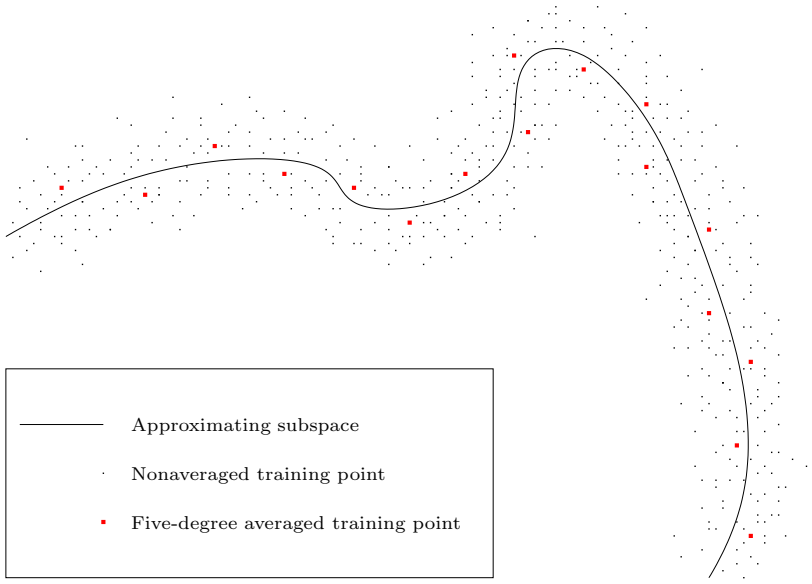


Figure 4.4: *Illustration showing non-averaged data points and their averages both being approximated by the same subspace. The non-averaged data are represented by black points; their averages by red points. The subspace marked by the solid curve provides an approximation to both.*

This was applied to all data sets.

Angular averaging to reduce size of training set

The ultimate goal of this work was to apply the dimension-reducing transformation to real-world radar data. Each radar image comprised 2401 pixels, and each separate data set comprised over 1000 individual images. Successive images within a given data set corresponded to increasing the aspect angle at which the vehicle was being measured by the radar; as the turntable upon which the vehicle was mounted rotated, an image was formed, then another, until a full 360° measurement of the vehicle was completed.

To ease the computational burden and speed up model selection these images were taken and incoherently averaged over 5° sectors. The averaging was performed prior to the vehicle being centred within the image in order to make use of the fact that the individual images falling within that angular segment will be optimally aligned. The numbers of unaveraged and averaged images in each data set over the 360° are given in parentheses in the data-set-partition tables. The angular-width for the sectors was chosen so that the maximum distance traversed by any one radar scatterer, as the vehicle rotates through 5°, does not exceed the dimensions of a radar resolution cell. This averaging can be regarded as merely performing a piecewise-linear smoothing of the data and the data points should be centred on the same subspace as the non-averaged originals: the overall nonlinear structure should be preserved. This is illustrated graphically in figure 4.4.

This was applied only to the training data.

4.4 Outline of experiments

The scheme proposed in this thesis for feature extraction applied to radar images of military vehicles is very flexible. The learning stage is trained using distances, or dissimilarities, between input vectors. The algorithm allows for some distances between input vectors to be regarded as more important than others, for example depending upon whether the two vectors belong to the same or different classes. The distances themselves can be modified to incorporate *a priori* knowledge about the dissimilarities between the data samples, for example to reflect the intuitive notion that two samples belonging to the same class should always be more similar to one another than if they belonged to different classes, regardless of aspect angle. Alternatively, the metric used to calculate the distances between data samples can be changed, for example replacing the Euclidean metric with the Box-Cox.

The simplest experiment was to calculate the dissimilarities between the samples using the Euclidean metric and make no use of the class labels associated with the data. This implementation was thus unsupervised and merely treated the data as a single ensemble, like PCA.

Supervision was then added in the form of varying the ratio of the contribution of inter- and intra-class distances during training. The inter-class distances are the most crucial for classification. This form of supervision allowed an investigation of the effect of training using inter-class distances, intra-class distances, and combinations of the two.

Another approach to incorporate supervision was to rescale the input distances. Specifically, the inter-class distances were magnified such that they were at least as large as the intra-class distances. The goal of the feature extraction was thus to maximize the separation of data samples belonging to different classes (subject to some constraint) and thus invites comparison with LDA.

Previous studies [66, 117, 118] have shown a benefit in the use of the Box-Cox metric when calculating distances (dissimilarities) between radar data samples. However, these conclusions were in the context of the metric used in a k-nearest-neighbour (k-NNR) classifier. This raises the interesting question of whether the feature extraction algorithm can learn a projection wherein Euclidean distances in feature space approximate Box-Cox distances in measurement space. After all, the network to be used for performing the feature extraction has the property of universal approximation which suggests that it should be possible to build in the Box-Cox transformation 'for free' as part of the nonlinear feature extraction. A natural comparison for this approach was then to perform an explicit

Box-Cox transformation on the data space globally. This was achieved simply by applying the Box-Cox transformation as a preprocessing step to all input data samples.

Finally, the use of kernel functions for performing the feature extraction suggests a comparison with Support Vector Machines (SVMs). SVMs also use kernel functions and offer theoretically very well motivated classification rules. SVMs thus provide a related, and 'state of the art', method that is optimized for classification, against which the adopted method of feature extraction plus 'bolted on' classifier can be compared.

4.5 Experimental method

4.5.1 Classification

As the focus of the research was on feature extraction with a view to improving classification performance, an obvious measure of success of a particular method of feature extraction was the subsequent error rate obtained by inputting the features into a classifier. Error rate is by far the most popular measure of classification performance [42] and it has the advantage of being easily interpreted. This then raises the issue of choosing a classification rule.

A suitable classifier should have the properties of not being so complex as to mask effects of the feature extraction, being easy to implement, being straightforward to interpret, yet offering reasonable performance. One such classifier is the k-NNR [31, 36, 127].

In a comparison of different values of k in the context of classifying radar images of military vehicles, previous research [7, 66] found $k = 1$ to be optimal. The resulting first-nearest-neighbour rule (1-NNR) has also been found to be effective in the classification of images [82]. In any case, Hand [42] comments that the choice of k is often not critical.

As 1-NNR classifies according to some measure of proximity between data points, it offers a simple and intuitive assessment of methods of feature extraction that seek to preserve or manipulate such proximities. Precisely because the 1-NNR decision boundaries are less smooth than those resulting from k-NNR with $k > 1$, the 1-NNR should be most sensitive to changes in the data distributions. For these reasons the first-nearest-neighbour rule (1-NNR) was chosen as the classifier for assessing the performances of feature extraction techniques.

4.5.2 Estimated error rate

Using the proportion of test samples misclassified (error rate) as the evaluation criterion raises two questions: "What error rate might be expected from random chance alone?" and "How statistically significant is a change in the observed error rate?"

Table 4.10: *Error rates expected from random-chance alone for the small and extended training sets.*

Class	Error rate (%)	
	Small set	Extended set
MBT	71	69
APC	66	65
ADU	63	66

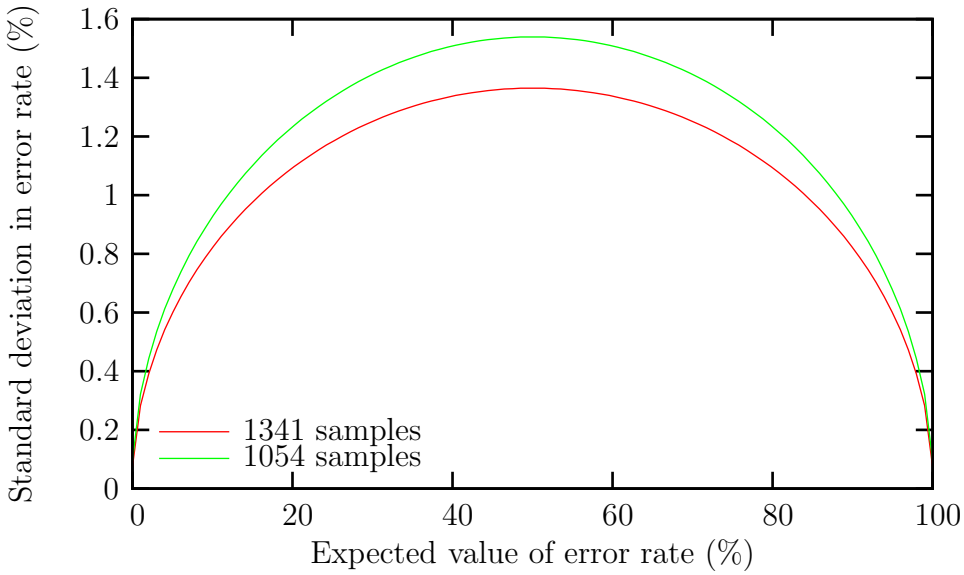


Figure 4.5: *Standard deviation of estimated error rate.*

Calculating random-chance error rates based on the relative proportions of the training data population sizes results in the error rates given in table 4.10. The populations are not exactly equal and so the random-chance error rates differ slightly from 67%.

If n_{mis} out of N samples are misclassified, it can be assumed that n_{mis} is binomially distributed with mean N times the true error rate. The posterior distribution of the true error rate (given n_{mis}) can then be derived [128] as a Beta distribution with mean

$$\mu = \frac{n_{mis} + 1}{N + 2},$$

and variance

$$s^2 = \frac{(n_{mis} + 1)(N - n_{mis} + 1)}{(N + 2)^2(N + 1)}.$$

For large n_{mis},N the mean is thus approximated by n_{mis}/N. For a given N and observed n_{mis}, the true error rate can be taken to be $\mu \pm s$. The smallest test set size is 1054 and the largest is 1341. The standard deviation, s, for each case is plotted as a function of the estimated error rate in figure 4.5. Thus, changes in error rate of $\gtrapprox 3$ percentage points

can reasonably be taken to be statistically significant.

4.5.3 Image shifting

The recentering of vehicles within the extracted image regions does not guarantee that any two vehicles will be optimally aligned. This is simply a recognition of the fact that all localization algorithms have an associated error: they cannot localize every vehicle with pixel precision [75]. The recentering acts to remove any systematic biases that would skew the results. The match between two vehicles could potentially be improved by shifting one of the images relative to the other to find a better match (smaller distance). This is simply a generalization to two dimensions of the sliding metric, as used in [117, 118] for one-dimensional radar range profiles. To shift images in both axes places a substantial processing overhead on the classification procedure: shifting by ± 1 pixel in each axis requires nine passes of the classifier and shifting by ± 2 pixels in each axis requires 25 passes.

To constrain this overhead, the amount of shifting performed was usually limited to ± 1 pixel. The reasons for this limit are twofold. Firstly, in the case of classifying the images directly, the distance calculations are most expensive because of the high dimensionality of the vectors representing the images. So, although the shifting can readily be interpreted as sliding one image over another, classification becomes very time consuming. Secondly, the shifting is performed on images prior to the calculation of features and an easy interpretation of the effect of the shifting is lost when classification is performed in feature space. So, although classification in low dimensionality feature space is far more rapid than using the images, it is unclear what effect the shifting has on the calculation of the features and thus what the results may mean. This latter problem can be illustrated by considering the feature extraction that is performed to represent the Earth's surface on a 2-dimensional map. A small shift in the position of an object located in the vicinity of the international date-line could result in the object disappearing off the right-hand-side of the map and appearing on the left. Alternatively, a short trip across one of the polar caps would result in the object re-appearing in a location very far from its starting point.

The inclusion of some shifting in the experiments will, however, provide useful information. If a shift of ± 1 pixel improves classification results using images then it indicates there is room for improvement in the alignment of the images. Whether the alignment of the images could be improved, and how this carries over into feature space, is a valid question for this study. The amount of shifting required in order to obtain the best results is a lesser concern, mostly of interest if the main focus of the work were on the recentering algorithm. Obtaining an answer to this latter question would require classification results

obtained using shifting of $\pm n$ pixels for a wide range of n, and would therefore be very time consuming. Thus, allowing a shift of ± 1 pixel should be sufficient to provide as much information as is required here.

4.5.4 Procedure

The experimental method was thus:

1. Train the feature extraction algorithm using specified training data;
2. Project the class exemplars onto the feature space;
3. Project the test images onto the feature space;
4. Classify the test images according to the 1-NNR;
5. Once all images for a given test data set are classified, calculate the error rate for that data set.

When shifting was applied to the test images, the classification decision for each test image was simply based on the minimum distance between the shifted images and the class exemplars (or their respective feature vectors).

Chapter 5

Baseline results

5.1 Introduction

This chapter gives baseline classification results obtained using the ISAR data both without any feature extraction and with some standard linear feature extraction techniques (PCA and LDA) applied prior to classification. These classification experiments provide results against which the subsequent classification experiments can be compared. Section 5.2 describes an implementation of PCA and the resultant visualization of the data. Section 5.3 presents a visualization of the data using LDA. Classification results obtained in the original measurement space, PCA-subspace, and LDA-subspace are then compared in section 5.4. Section 5.5 considers the application of the radial basis function network to the radar data. Practical issues are discussed and an approach to model selection described. Theoretical and practical considerations allow predictions to be made concerning the model parameters. These are illustrated and supported by pilot studies using real data. Finally, the chapter is summarized in section 5.6.

5.2 Principal Components Analysis

5.2.1 Introduction

The radar images used in this study were 49-by-49 pixels. This resulted in data vectors of dimension 2401. Conventional calculation of the data covariance matrix would result in a 2401-by-2401 matrix which, when training on averaged images (72 per dataset), would be far from full rank. This represents a high computational burden to compute a small number of eigenvectors.

In recognition of this, the computationally efficient approach described fully in [115] was adopted for the calculation of the principal components. In essence this technique

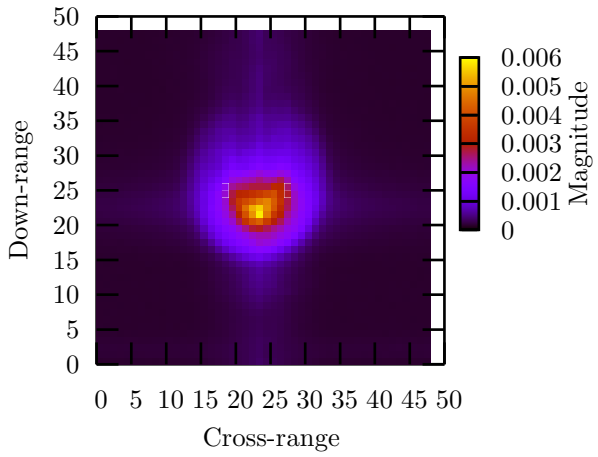


Figure 5.1: *Mean training image. This is subtracted from the training data prior to calculation of a covariance matrix.*

uses the covariance matrix formed from the transpose of the data matrix. The eigenvectors of this matrix then define weightings for a sum of the training images to form the principal axes, or eigenimages. These eigenimages in turn define projections onto a reduced-dimension feature space. This feature space can be used to visualize structure in the data. Classification can also be performed in this feature space.

5.2.2 Projecting to feature space

The averaged small training-data-set, comprising three sets of 72 images, was used to form the eigenimages. The first stage, prior to the calculation of the covariance matrix, was to obtain the mean image and subtract it from each of the training images. This mean image is shown in figure 5.1. The total number of training images was 216 and this yielded a 216-by-216 covariance matrix. Eigen decomposition of the 216-by-216 matrix resulted in 216 eigenvectors with the eigenvalues shown cumulatively in figure 5.2. From figure 5.2 it can be seen that 80% of the variance was contained in the first 20 components, and 90% of the variance in the first 40 components.

For data visualization purposes the first three principal components are of most interest. The first two components account for approximately 30% of the total variance and the first three account for approximately 39%. The corresponding eigenimages are shown in figure 5.3. The eigenimages heavily weight image pixels associated with the target area, with the first eigenimage particularly interpretable as a measure of how 'head on' the measured vehicle appears. Together these three eigenimages can be used to project

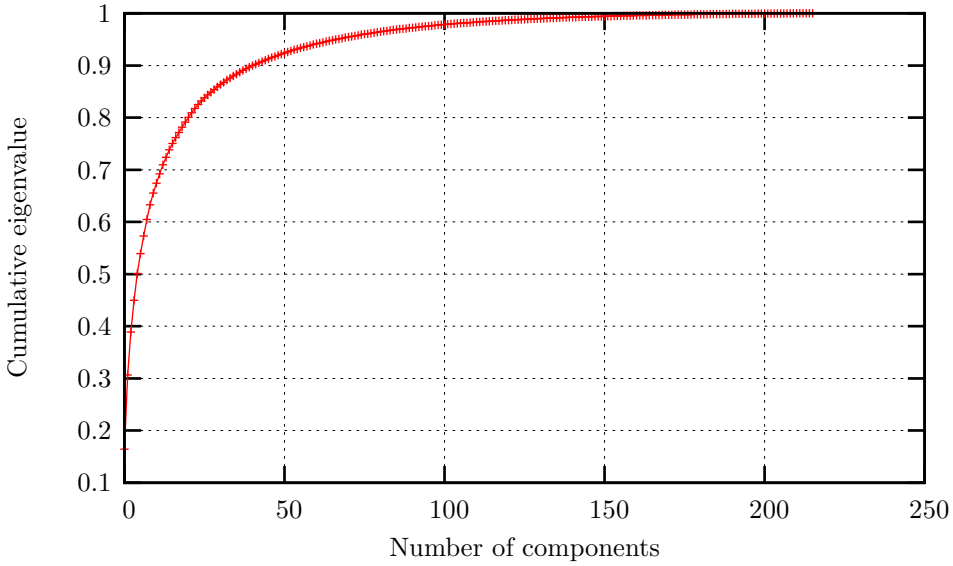


Figure 5.2: *Normalized cumulative eigenvalues for the 216 averaged training images. The eigenvalues are summed starting with the largest and including successively smaller values.*

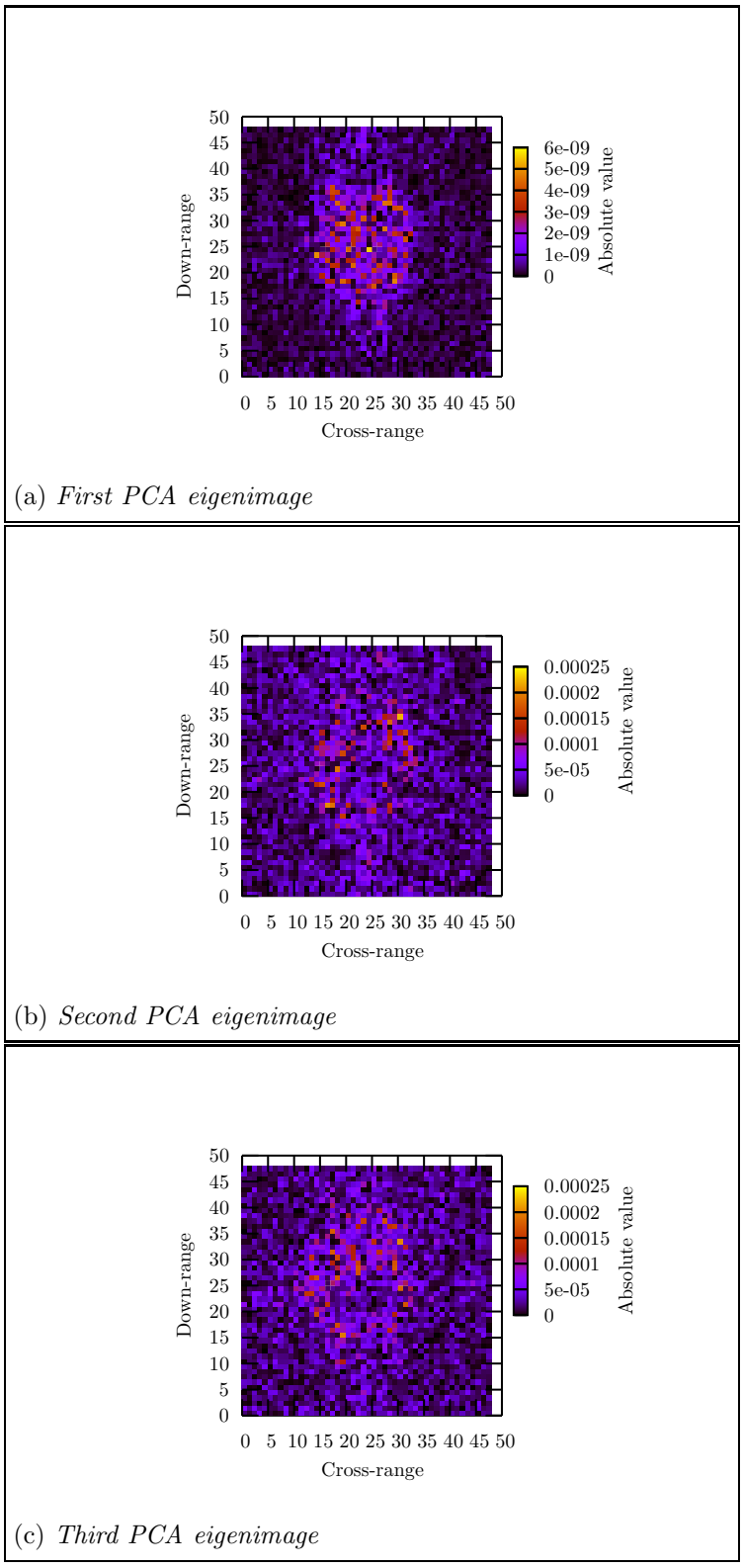
the data to a subspace of up to 3 dimensions.

Using the first two eigenimages to project the training data to a two-dimensional feature space produced the traces shown in figure 5.4. Successive points joined by lines correspond to successive average images as the target rotates through 360° in five-degree steps. Such traces logically must be closed paths as the target completes a full rotation: as the target completes a full rotation and tends towards its initial aspect angle, the corresponding image tends towards the initial image.

This is most clearly seen for the trace belonging to the APC — a particularly symmetric vehicle with a simple 'clean' shape — shown alone in figure 5.4c along with the projections of the nonaveraged images. The trace for this vehicle completes two closed paths, taking into account the fact that the last and first images are not joined. This is equivalent to a closed path every 180°, reflecting the fact that the vehicle has significant 180° symmetry.

Whilst the projections plausibly capture structure present as the vehicles rotate, there is very little separation between the classes as demonstrated in figure 5.4a. For each set of projections in figures 5.4b, 5.4c, and 5.4d, the projection of the averaged images appears to follow a smoothed path through the projected nonaveraged images. This is consistent with the averaged images being smoothed versions of the nonaveraged images.

Adding the third eigenimage to project to a 3-D feature space produced the traces shown in figure 5.5. The third principal component appears as the third axis. Again, lines join successive images in a vehicle's rotation through 360°. As only a 2-D projection of this 3-D space can be represented, we have attempted to select a slice that best

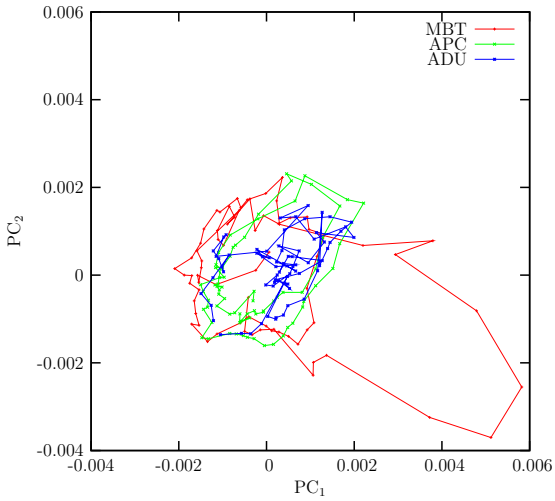


(a) *First PCA eigenimage*

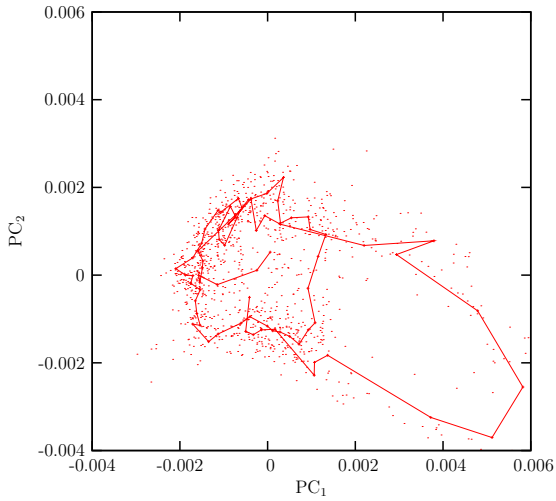
(b) *Second PCA eigenimage*

(c) *Third PCA eigenimage*

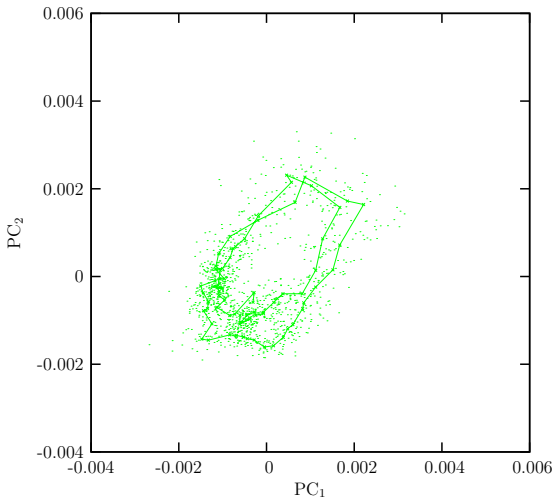
Figure 5.3: *First three PCA eigenimages derived from the averaged training images.*



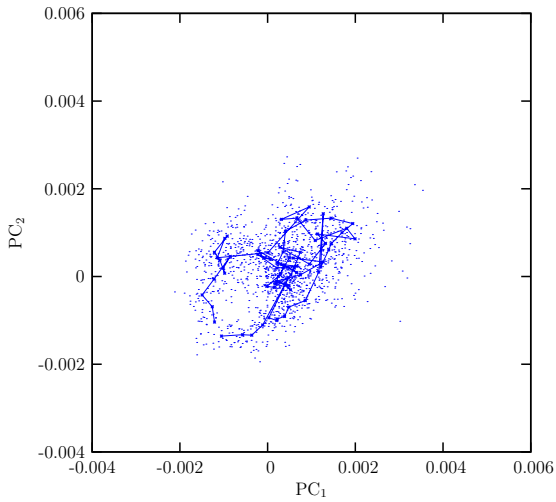
(a) *Averaged training data, all three classes.*



(b) *MBT training data, averaged and nonaveraged.*



(c) *APC training data, averaged and nonaveraged.*



(d) *ADU training data, averaged and nonaveraged.*

Figure 5.4: *Training images projected onto their first two principal components (PC_1 & PC_2). Successive averaged images are joined by lines demonstrating the vehicles' trajectories in $\mathbb{R}^2$ as the vehicles rotate.*

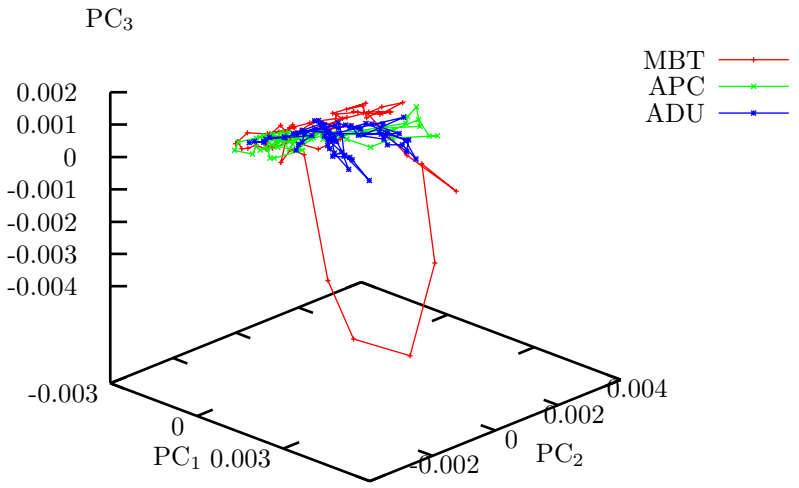


Figure 5.5: *Averaged training images projected onto their first three principal components (PC_1, PC_2, & PC_3). Successive images are joined by lines demonstrating the vehicles' trajectories in $\mathbb{R}^3$ as the vehicles rotate.*

demonstrates the additional structure present in the third dimension. The inclusion of this extra dimension does not seem to add anything noticeable in the way of inter-class separability. In an attempt to make any class separability clearer, the data for only the MBT and APC classes are shown in figure 5.6. These two classes appeared to be the most separable of the three in these dimensions and the figure presents the projection from an orientation that best conveys any separability between them. It is hard to be convinced that there is any worthwhile separation at this dimensionality.

5.2.3 Dimensionality of feature space for classification

For classifying test data in feature space, class exemplars for the 1-NNR classifier were constructed by taking the unaveraged training data, subtracting the previously computed mean image (figure 5.1), and projecting onto the first k principal components. The same steps were performed on the test images. When shifting of the test images was performed, this was done prior to the projection to feature space.

Classification results for each of the data sets in Group 1 are shown as a function of feature space dimension (number of principal components used) in figure 5.7. In general, all data sets show an initial rapid fall in error rate as the number of principal components used increases. This levels off at around 20–30 dimensions. The banding together of scores, particularly evident in figure 5.7a, is not attributable to grouping by vehicle type. For example, ADU1-3 and ADU1-5 — both representing the same ZSU vehicle — are amongst the two most widely differing scores. The inclusion of shifting in the process

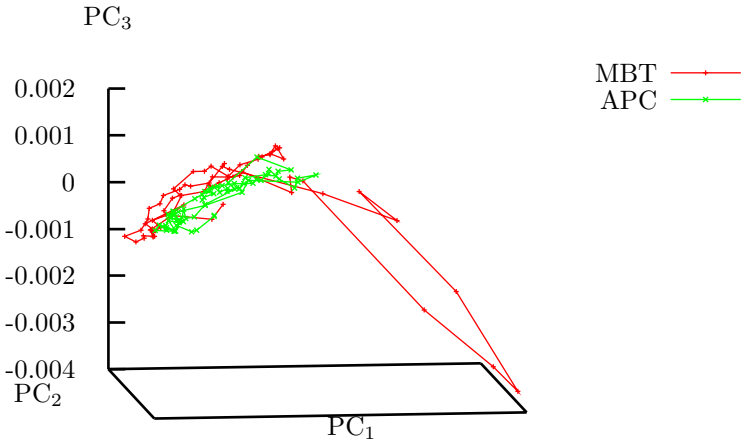
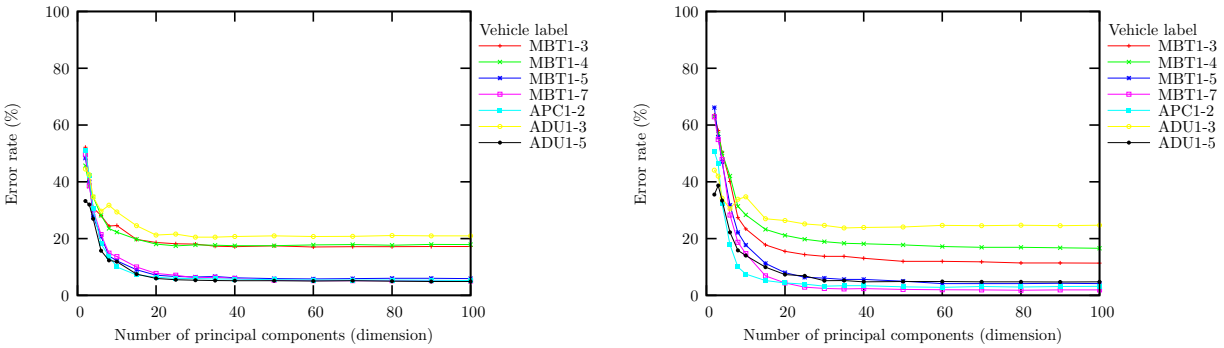


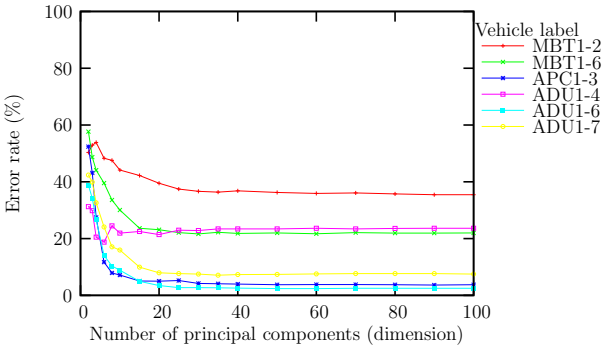
Figure 5.6: *Averaged training images for only the MBT and APC projected to their first three principal components (PC_1, PC_2, & PC_3). Successive images are joined by lines demonstrating the vehicles' trajectories in $\mathbb{R}^3$ as the vehicles rotate.*



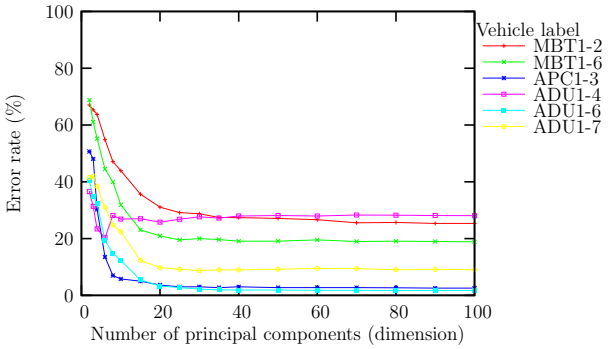
(a) *No shifting of input images prior to projection.*

(b) *Shifting of* ± 1 *pixels performed on input images prior to projection.*

Figure 5.7: *Classification results on Group 1 data sets after projection onto principal components.*

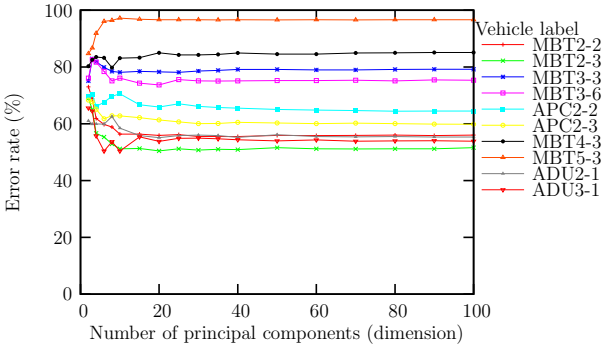


(a) *No shifting of input images prior to projection.*

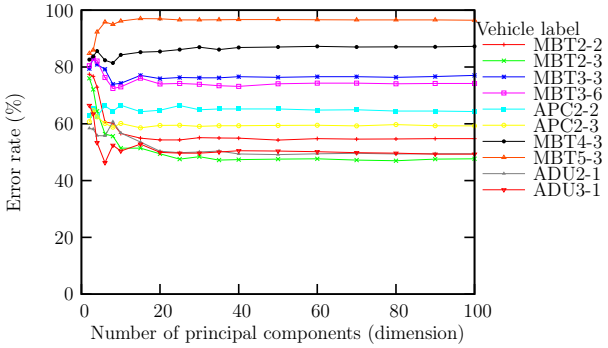


(b) *Shifting of* ± 1 *pixels performed on input images prior to projection.*

Figure 5.8: *Classification results on Group 2 data sets after projection onto principal components.*



(a) *No shifting of input images prior to projection.*



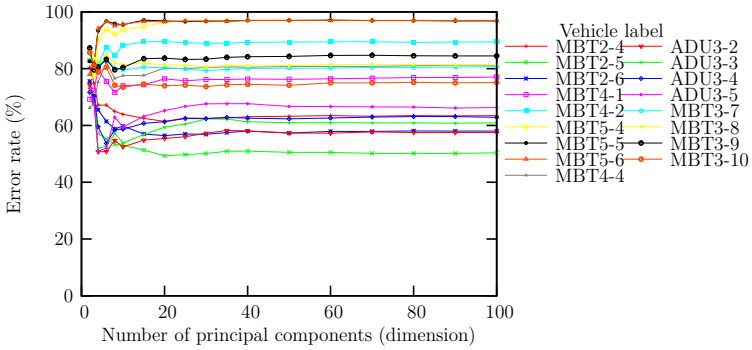
(b) *Shifting of* ± 1 *pixels performed on input images prior to projection.*

Figure 5.9: *Classification results on Group 3 data sets after projection onto principal components.*

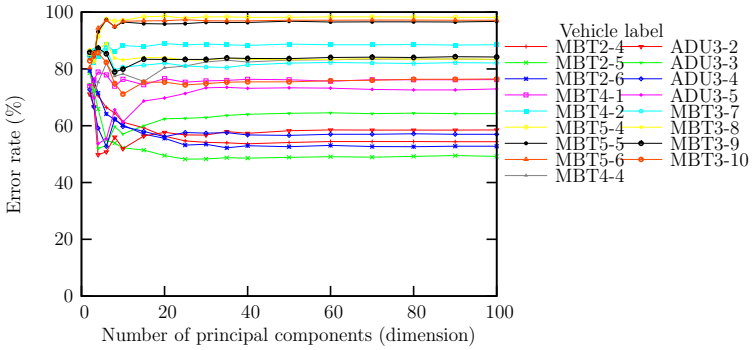
gave rise to the classification results given in figure 5.7b. Overall the effect is one of increasing the spread of results: the lowest error rate in that figure is lower than when shifting was not performed, but the highest score is higher.

Results for Group 2 data sets are given in figure 5.8. Again, an initial rapid fall in error rate is seen, with little change observed above 30 dimensions. In this group, the representations of the ZSU vehicle — ADU1-6 and ADU1-7 — are not so widely separated as they are in Group 1. The inclusion of shifting has the effect of lowering the highest score with little change in the lowest.

Group 3 comprises vehicle types not represented in training. Consequently, error rates are much higher against the data sets of this group, as shown in figure 5.9. For both no shifting, figure 5.9a, and shifting, figure 5.9b; there are increases and decreases as the number of principal components increases. Regardless of whether the error rate against a particular data set suffers an initial rise or fall with increasing dimension, performance



(a) *No shifting of input images prior to projection.*



(b) *Shifting of ±1 pixels performed on input images prior to projection.*

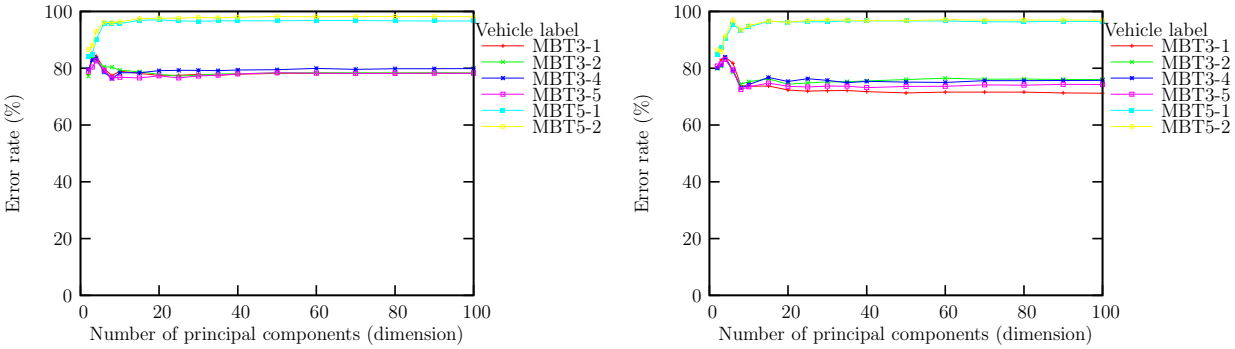
Figure 5.10: *Classification results on Group 4 data sets after projection onto principal components.*

again remains quite stable after 20–30 dimensions.

The results for Group 4, shown in figure 5.10, show the same trends as those for Group 3. The best performance was only about 50%. Some data sets demonstrate an initial rise with increasing feature space dimension, others an initial fall. The scores are reasonably stable after 20–30 dimensions.

All classification results for data sets belonging to Group 5, shown in figure 5.11, are worse than would be obtained by assigning class labels at random. There was, however, the pattern in the results of an initial change in performance as the number of principal components increased followed by a levelling off.

In line with the initial rapid rise in cumulative eigenvalue, shown in figure 5.2, the classification results demonstrate an initial, usually rapid, change with increasing number of principal components. The change in classification performance is minimal by the time the number of principal components used reaches 20–30. Any further increase in the number of components seems to have little effect on classification performance. This is consistent with the notion that the first components capture the gross structure that contains most of the information and the inclusion of further components, which do not alter the components (eigenvectors) already used, serves only to add less and less useful



(a) *No shifting of input images prior to projection.*

(b) *Shifting of* ± 1 *pixels performed on input images prior to projection.*

Figure 5.11: *Classification results on Group 5 data sets after projection onto principal components.*

structure until eventually subsequent components only add noise to the captured data structure. Thus, it seems reasonable to quote classification performances using the first 30 principal components as being reasonably indicative of the level of performance achieved in PCA feature space.

5.3 Linear Discriminant Analysis

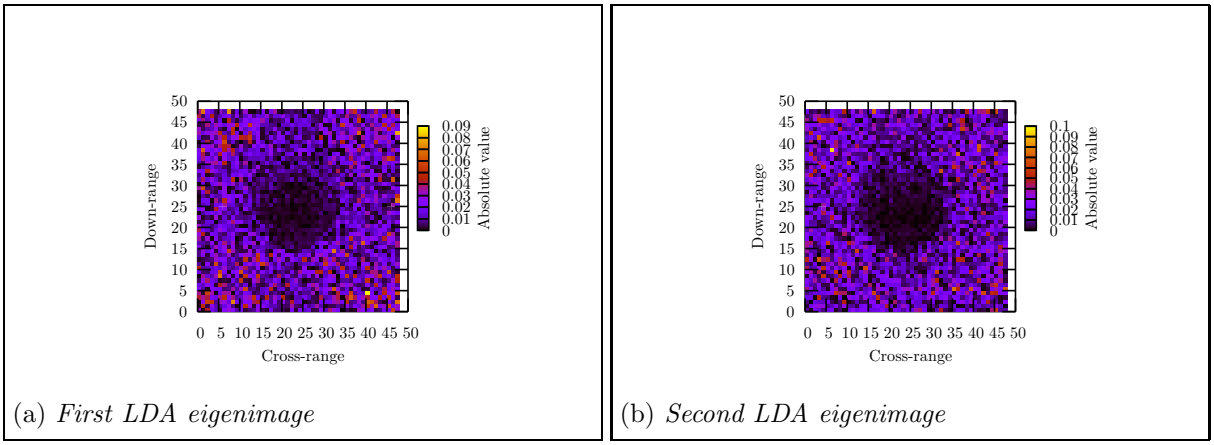
5.3.1 Introduction

The solution of the general symmetric eigenvector equation $\mathbf{Bw} = \lambda\mathbf{Sw}$, where $\mathbf{B}$ is the between-class scatter matrix and $\mathbf{S}$ is the assumed-common covariance matrix, to be solved for LDA, is a problem independant of the number of training images used. For this work the non-averaged training data were used for determining the eigenvectors. The resulting eigenvectors (or eigenimages) provide the directions $\mathbf{w}$ that maximize the separation between the sample means of the classes.

Because there are three classes defined, the number of significant eigenvectors, and hence the dimensionality of the feature space, is limited to two. The two dimensional feature space is defined by a projection of input images onto the directions $\mathbf{w}$. This allows visualization of the data in two dimensions and, by comparing distances from the exemplar training data in this feature space, the 1-NNR can be used to make classification predictions.

5.3.2 Projecting to feature space

The two eigenimages obtained from the small training set are shown in figure 5.12. In



(a) *First LDA eigenimage* (b) *Second LDA eigenimage*

Figure 5.12: *Eigenimages obtained by performing LDA on the small training set.*

contrast to the eigenimages from PCA, these eigenimages do not heavily weight pixels that can be attributed to target structure such as vehicle orientation. Instead, the eigenimages have a central region of low weight magnitudes. This region is smaller for the first eigenimage and larger for the second. The eigenimage weights have an approximately radial dependence about the centre of the image. This suggests that the radial extent (size) of the vehicles is the dominant physical feature that separates the classes.

Using these two eigenimages to project the training data to two dimensions yields the plot shown in figure 5.13. The classes appear separated to a degree not realized by the PCA projection to this low a dimension. There is not, however, anything like the structure evident within a class that was observed with the PCA projections: The trajectories described by successive points derived from the averaged training images appear to 'bounce around' fairly randomly within their cluster, rather than describe a perceivable track that exhibits some dependency on aspect angle. This highlights a key difference between the two linear techniques of PCA and LDA. PCA attempts to capture structure within the data, caring nothing for distinctions between different classes; whilst LDA attempts to capture (or enhance) distinctions between different classes, caring nothing for the structure within the classes.

Using the projections onto the two eigenimages to define a feature space, as visualized in figure 5.13, the unaveraged training samples were projected to form the exemplars for the 1-NNR classifier. The test data were likewise projected and subsequently classified in feature space using 1-NNR.

5.4 Benchmark classification results

In this section we present results for classification performed in the original measurement space using the full dimensionality images, in the PCA feature space using 30 dimensions,

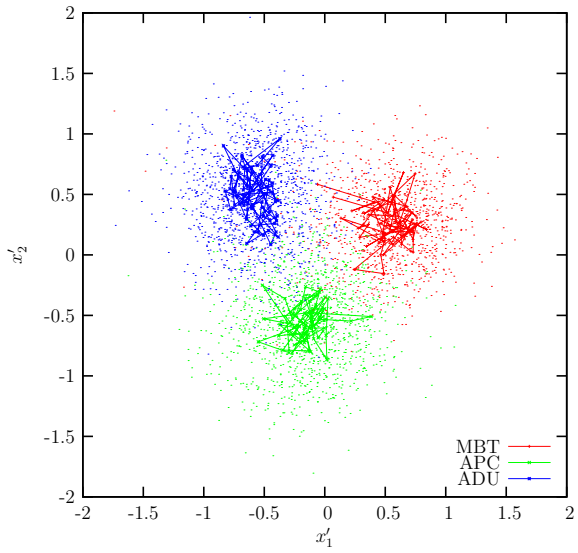
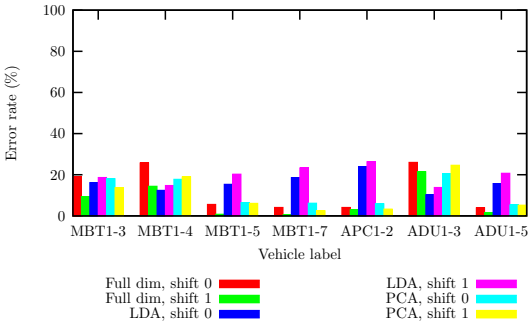


Figure 5.13: *Projection of both averaged and unaveraged training data sets to 2-D using the two eigenimages from LDA. The unaveraged training data are represented as points, the averaged training data are shown with lines joining successive aspect angles.*

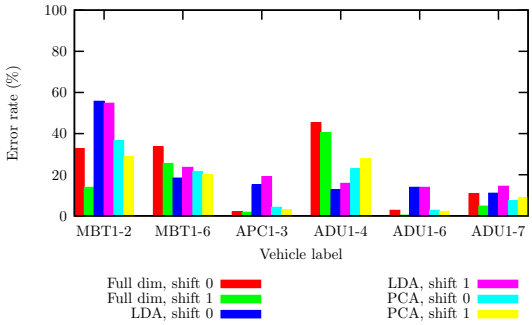
and in the LDA feature space of two dimensions. The results are shown in figure 5.14.

Figures 5.14a and 5.14b illustrate the error rate obtained on Groups 1 and 2, respectively. These two groups are important because they comprise the same vehicles used both to train the feature extraction and provide the class exemplars for the classifier. Classification performance in PCA feature space broadly matches that obtained using the full dimensionality images, whereas LDA sometimes yields results that differ noticeably. Compared to the error rates obtained using PCA and the full dimension images, the error rates after LDA are much higher for MBT1-5, MBT1-7, APC1-2, ADU1-5, MBT1-2, APC1-3, and ADU1-6 whilst falling to about 50% for ADU1-3. The error rate after LDA for ADU1-4 is also less than 50% of that for the full dimension images, although the differential relative to PCA is less. Furthermore, shifting benefits classification performed with the full dimension images to a degree not seen when classification is performed in either PCA or LDA feature space. All too often, shifting increases the error rate for classification performed in feature space. LDA suffers more than PCA in this regard.

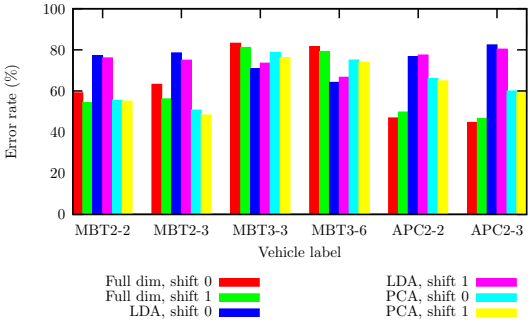
Figures 5.14c through 5.14h show the error rate obtained on the groups comprising vehicles not present during training. These groups are characterized by generally poor performance across all data sets. Notable exceptions are sets ADU2-1, ADU3-1, ADU3-2, ADU3-3, ADU3-4, and ADU3-5. These data sets exhibit noticeably lower error rates for the LDA feature space, relative to the full dimension and PCA results. This good performance on data sets ADU3-1, ADU3-2, ADU3-3, ADU3-4, and ADU3-5 is reassuring for LDA: ADU3 is a ZSU-23-4 (the same ADU vehicle type that was present in training)



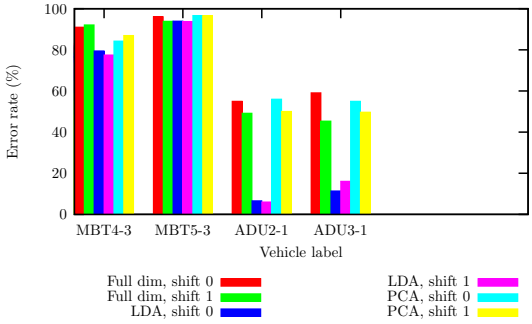
(a) *Group 1*



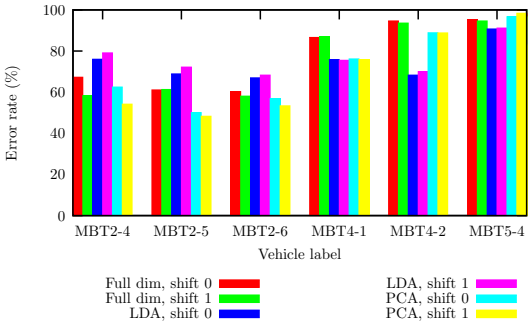
(b) *Group 2*



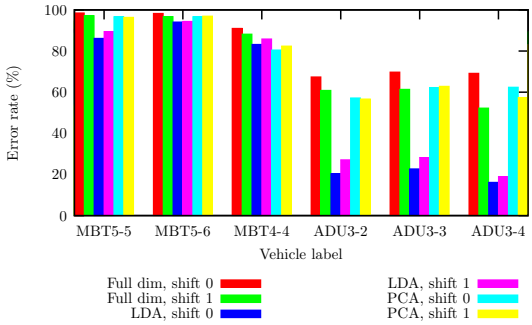
(c) *Group 3*



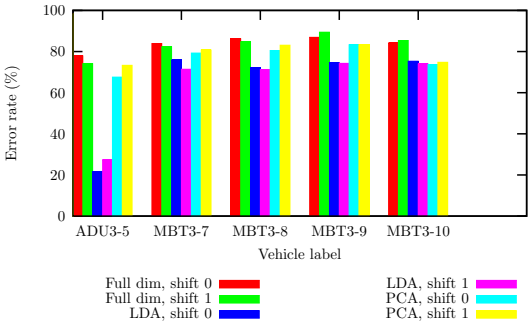
(d) *Group 3 (cont.)*



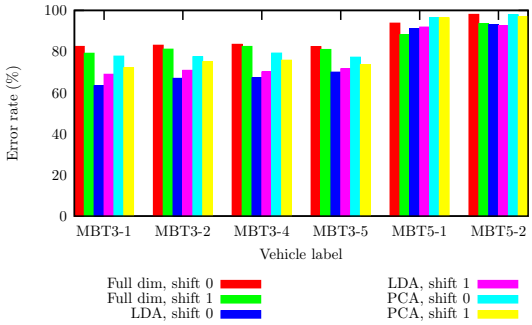
(e) *Group 4*



(f) *Group 4 (cont.)*



(g) *Group 4 (cont.)*



(h) *Group 5*

Figure 5.14: *Classification performance on data sets using full dimensionality, PCA features, and LDA features. For each data set, adjacent columns denote performance with no shifting of the images and with a shift of* ± 1 *pixels.*

albeit a different vehicle and measured some three years earlier. Despite the previous good performance of full dimensionality and PCA on this vehicle type (ADU1: figures 5.14a and 5.14b), these techniques now give high error rates (ADU3: figures 5.14d, 5.14f, and 5.14g) for this vehicle type. Another aspect of the classification results not immediately explicable is that of ADU2-1. Despite not being present during training, this vehicle evidently ends up closer to the ADU class in LDA feature space than to any other class.

Finally, Group 5 in figure 5.14h demonstrates consistently low classification accuracies, at the level of, or worse than, random chance.

5.5 Applying the radial basis function network to radar data

5.5.1 Model selection considerations

Unlike for PCA and LDA, there are model parameters to specify for the RBFN before training can proceed. These model parameters specify the form of the network that is to be trained. For the RBFN these model parameters are the number of centres and the widths of the basis functions. Strictly speaking, the positions of the basis-function centres must also be specified, but this can either be done as part of the network training if a more complicated nonlinear optimization problem is acceptable, or it can be performed by a separate unsupervised stage prior to network training; fundamentally it is the number of centres, rather than their positions, that must be specified in advance.

The architecture of the RBFN, and the form of RBF used, permit analytic predictions of the response of the network to changes in model parameters. These considerations will assist the task of model selection.

Number of centres

The output of the network is a linear weighted sum of nonlinear functions of the input data. The ith feature extracted from input data vector $\mathbf{x}$ is given by (3.1) for a network using l basis functions, and hence l centres. If an extra centre were added to the network, the new ith feature from vector $\mathbf{x}$ would be given by:

$$f'_i(\mathbf{x}) = \sum_{j=1}^{l} w_{ji}\phi_j(\mathbf{x}) + w_{l+1,i}\phi_{l+1}(\mathbf{x}) \tag{5.1}$$

The network comprising $(l+1)$ centres contains the l-centre network as a special case. Since the network is trained to minimize the stress (3.6), the addition of this extra centre

cannot result in an increase of stress. If the addition of this extra centre would cause the stress to increase, this could be prevented merely by setting the weight $w_{l+1,i}$ to zero. However, this applies when a new centre is added to a fixed constellation of l centres. In this study, the positions of the centres are optimized by a k-means algorithm and so the positions of the previous l centres are adjusted upon the inclusion of the $(l+1)$th centre. Nevertheless, provided nothing goes pathologically wrong with the centre-selection stage when the $(l+1)$th centre is added, it is reasonable to expect that the addition of extra centres will tend not to increase the stress.

Although a broadly-monotonic reduction in stress is thus expected with an increasing number of centres, such increase in the network complexity comes at the price of an increase in the computational burden. The methodology adopted here, of training the network on the five-degree averages of the input data, aids in this regard. Under this scheme, only a reduced-number of smoothed data points are used for choosing both the number and locations of RBF centres. As well as reducing the computational load, this is likely also to help guard against overtraining and the fitting of fine structure attributable only to noise. However, the final arbiter of the suitability of any given network should be its performance on validation data.

Kernel bandwidth

The form of the basis functions used here is the Gaussian (3.22). The bandwidth, h, of the function controls the domain of influence of each basis function about its centre. For sufficiently small h, a basis function will encompass a very small volume that does not contain any data points. As the value of h increases, the volume of influence of this basis function increases and starts to encompass more and more data points. For large enough h, the power-series expansion of the Gaussian can be truncated after the first term in $\mathbf{x}$:

$$\begin{aligned} f_i(\mathbf{x}) &= \sum_{j=1}^{l} w_{ji}\phi_j(\mathbf{x}) \\ &\approx \sum_{j=1}^{l} w_{ji}\left(1 - \frac{|\mathbf{x}-\mathbf{c}_j|^2}{h^2}\right) \\ &\approx \sum_{j=1}^{l} w_{ji} - \sum_{j=1}^{l} w_{ji}\frac{|\mathbf{x}-\mathbf{c}_j|^2}{h^2}, \end{aligned} \tag{5.2}$$

and the nonlinearity is approximated by a quadratic. The first term on the right-hand-side of (5.2) is merely an offset and is irrelevant in the calculation of distances between points. Therefore, the stress (3.6) can be minimized with respect to $w'_{ji} = w_{ji}/h^2$ without affecting the solution. This has the consequence, for sufficiently large h^2, that the weights

w_{ji} scale with h^2 and the stress is independent of h^2.

Therefore the following predictions can be made for the behaviour of the network as the value of h^2 changes. The stress (3.6) is expected to decrease initially as the value of h^2 increases from being very small and the basis functions start to encompass data points. As the value of h^2 increases, the nonlinearity will be approximated by a quadratic; the stress will become independent of h^2; and the points in feature space will be subject to an offset that scales with h^2.

Training on averaged images

Averaging the p-dimensional radar images reduces the variance and thus the spread of points in $\mathbb{R}^p$. Choosing centres based on the averaged images and training the RBFN using averaged images, although speeding up the training stage, will tend to select for model parameters that are optimal for the averaged images with their reduced spread. The effect will be to bias the RBF widths towards smaller values than are required to encompass the larger spread of the non-averaged images. Consequently, using RBF model parameters determined solely from averaged images will result in non-averaged images lying in the tails of their nearest basis functions. This would result in small feature-values calculated for non-averaged images. This situation can be avoided by performing model validation using non-averaged images.

5.5.2 Pilot studies

Unsupervised loss

The weights α_{ij}, in the loss function (3.6), control the contribution to the overall error of the mismatch between the input and output distances for points $\mathbf{x}_i$ and $\mathbf{x}_j$. As mentioned in section 3.3, a simple implementation is to have all point-pairs contribute equally to the loss. This renders the loss function independent of the class labels of the training data. In this regard it is unsupervised and the projection of the data will reflect the global data structure treating the data as a single ensemble.

To visualize this structure, projections to two, and sometimes three, dimensions are helpful. An initial grid search over the two RBF parameters of number of centres and width is required to identify suitable parameter values. In this manner, the values of the resultant stress define the height of a two-dimensional surface. A useful preliminary step is to plot a histogram of the distances of each training data point from its nearest basis-function centre for a given number of centres. This provides a guide to the values of width that should be considered.

It is convenient to plot distance squared, d^2_{Min}, rather than distance, as this then yields

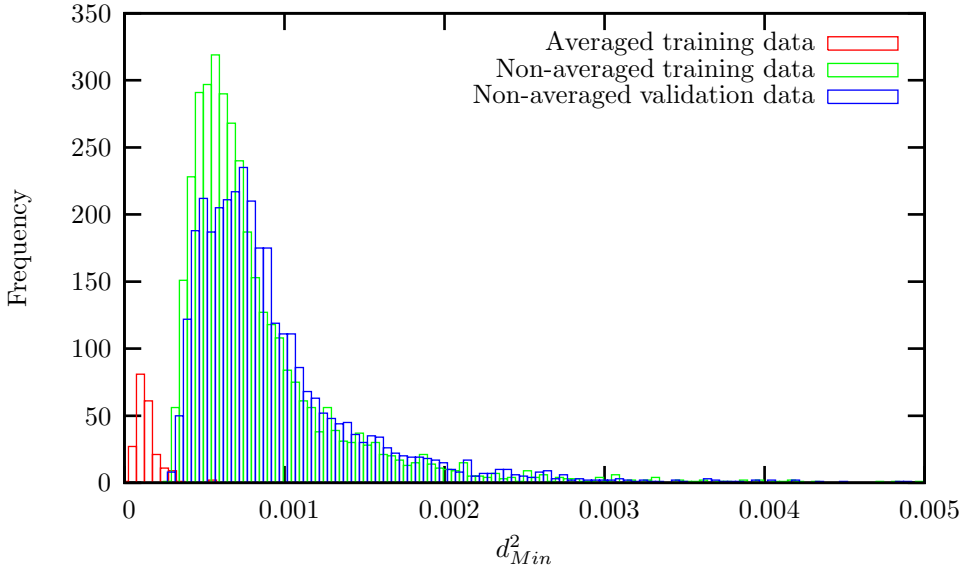
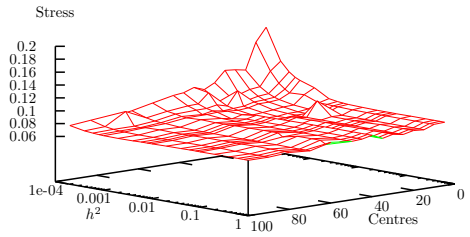


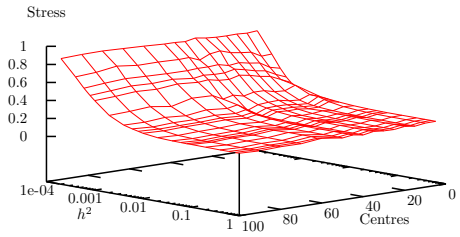
Figure 5.15: *Histogram of frequencies of minimum distances between data points and RBF centres when using 40 centres. The frequency is the number of samples falling within a given bin. The binsize is* 0.00005*. Histograms for averaged training data, non-averaged training data, and non-averaged validation data are shown.*

estimates of h^2 to insert into the basis function (3.22). Such a histogram is shown in figure 5.15 for the averaged training data points, the non-averaged training data points, and the non-averaged data points belonging to the validation data set. The lower frequencies and smaller values of d^2_{Min} in the histogram for the averaged training data reflect both the smaller sample size and smaller variance in the distances relative to the non-averaged data. The histograms for both the non-averaged training and validation data sets suggest that as the value of h^2 approaches 0.001, the width of the RBFs should be becoming large enough to encompass these data points. This value of h^2 can thus be expected to mark a region of rapidly changing stress for the non-averaged data.

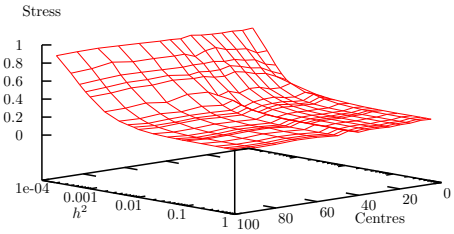
Surface plots arising from a grid search over h^2 and number of centres are presented in figure 5.16 for projecting to two dimensions. There is negligible change in stress for the non-averaged data as a function of the number of centres, with 40 appearing a reasonable choice. Figure 5.17 shows the stress obtained as a function of h^2 when using 40 centres. As expected from the histogram of distances, the most rapid reduction in stress calculated for non-averaged data occurs around $h^2 = 0.001$. The curves show little change in stress once h^2 reaches 0.01. This independence of stress on the value of h^2 for sufficiently large h^2 verifies a prediction of section 5.5.1. The similarity of the curves for non-averaged training and validation data is reassuring that once model parameters are selected that are suitable for the non-averaged training data, these parameters are also suitable for the validation data. The value of 0.01 for h^2 is a minimum value, below which the stress rises



(a) *Averaged training data*



(b) *Non-averaged training data*



(c) *Non-averaged validation data*

Figure 5.16: *Stress calculated when data are projected to two dimensions by RBFN trained using unsupervised loss function.*

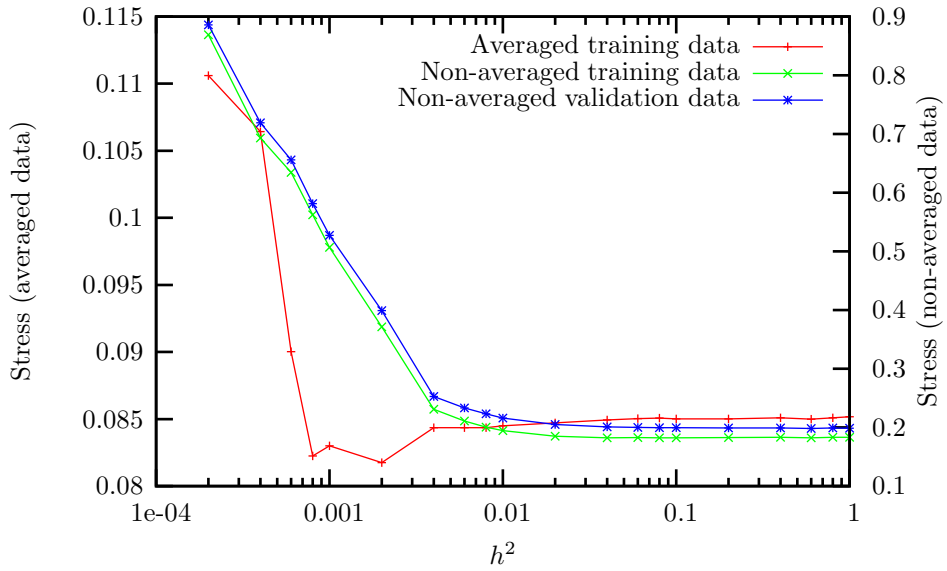


Figure 5.17: *Values of stress derived for projections to two dimensions using 40 centres. Plots are shown for the resultant stress after training on the averaged training data, and the stress subsequently calculated for the non-averaged training and validation data sets.*

rapidly. From the previous theoretical argument, further increasing h^2 should have no worse an effect than the addition of an offset. A justifiable choice of value for h^2 is 0.1 in order to provide added surety that the basis functions are wide enough to span the data. Thus, suitable parameters for projecting the data to two dimensions are indicated to be 40 centres with a value of 0.1 for h^2.

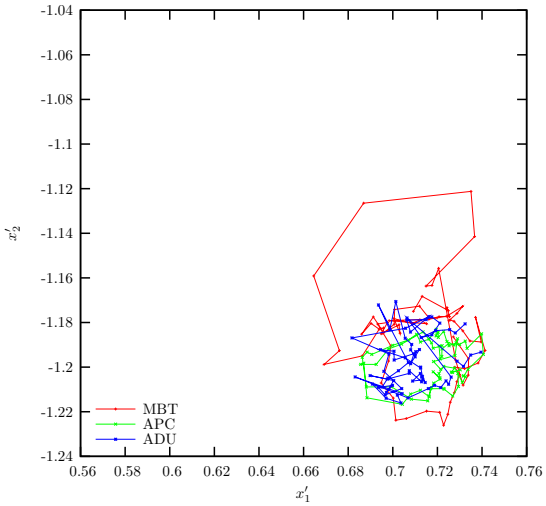
Using these parameters to specify an RBFN, followed by training on the averaged training data, produced the projection to two dimensions of the averaged and non-averaged training data points shown in figure 5.18. The plot of all three (averaged) data sets together, figure 5.18a, shows little separability between the three classes. The plots of averaged and non-averaged points from each class, figures 5.18b through 5.18d, show the majority of non-averaged points are projected around the averaged points, but with a small number biased off in the direction of the coordinate origin. This bias is consistent with the prediction that some data points could lie close to the tails of their nearest basis functions and give rise to small feature values.

Supervised loss

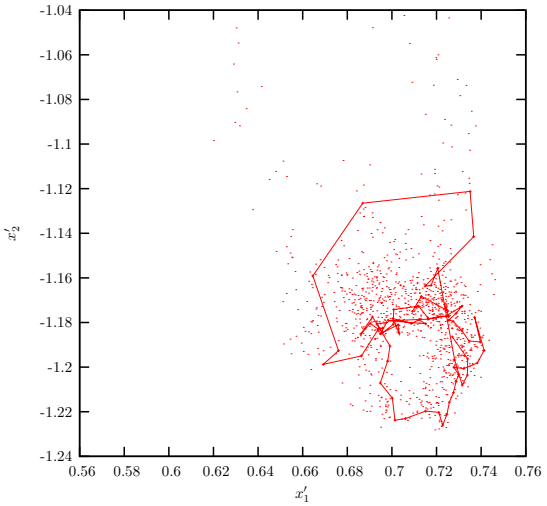
As described in section 3.3.1, the 'default' loss function of the previous section can be made sensitive to the class labels of the training data points. In this manner the loss function can be regarded as being supervised: it uses the class labels of the data used during training. The training data remain the same as before, the selected centres remain the same, and the structure being modelled (as given by Euclidean distance) is essentially the same. Thus it is reasonable to use the same network parameters that were used to specify the network of the previous section: 40 centres and $h^2 = 0.1$.

Another key parameter to be considered is ρ, as in (3.23), which is allowed to take values between zero and one. The value zero means that only distances between points belonging to different classes contribute to the stress; the value one means that only distances between points of the same class contribute. Values in-between result in a mixing of these two cases. The two extreme cases are thus of interest in comparing the two discrete components of the data structure: inter-class structure and intra-class structure. These two aspects of the data structure, for projections to two dimensions, are shown in figure 5.19.

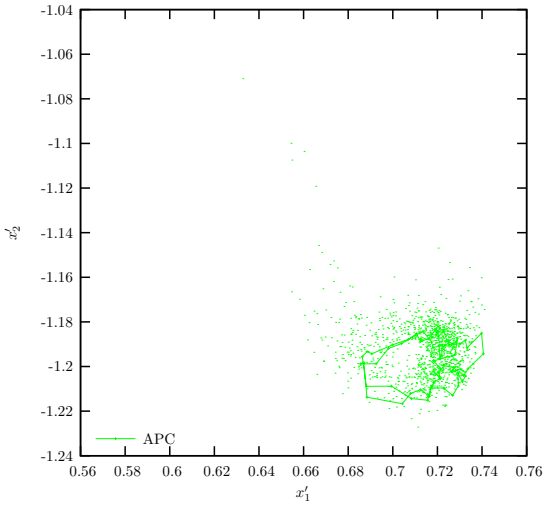
The projections of the averaged and non-averaged training data for $\rho = 0$ are shown in figures 5.19a and 5.19b, respectively. The corresponding projections for $\rho = 1$ are shown directly below those, in figures 5.19c and 5.19d. The projections of non-averaged data points exhibit essentially the same distribution as the projections of the corresponding averaged data points, albeit with larger spread. The projections learned using $\rho = 0$ (inter-class distances) show greater separability between the classes than those obtained



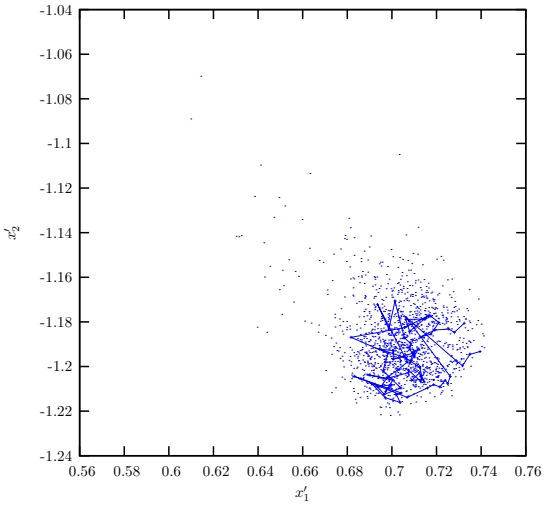
(a) *All three averaged training data sets projected to two dimensions.*



(b) *MBT training data, averaged and non-averaged, projected to two dimensions.*

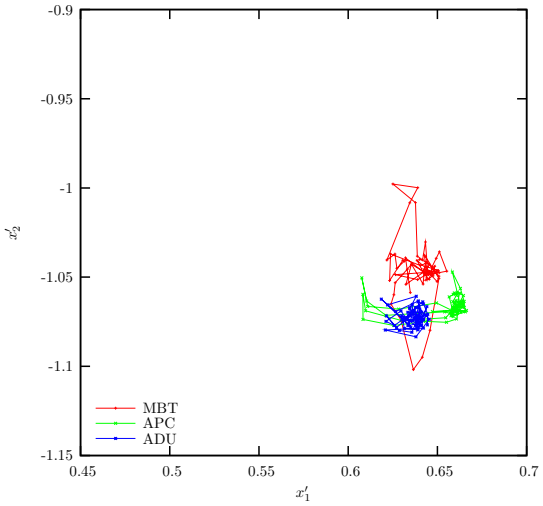


(c) *APC training data, averaged and non-averaged, projected to two dimensions.*

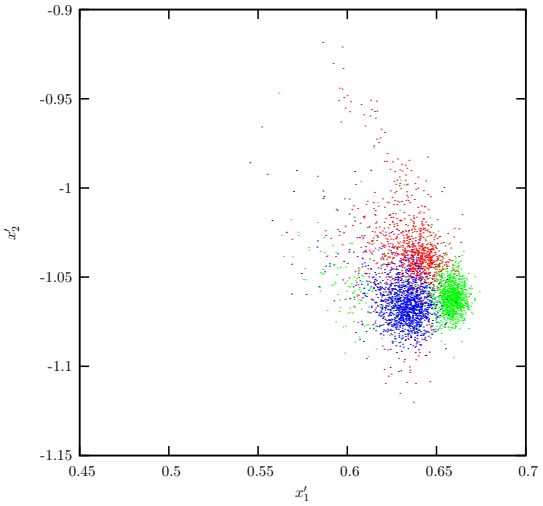


(d) *ADU training data, averaged and non-averaged, projected to two dimensions.*

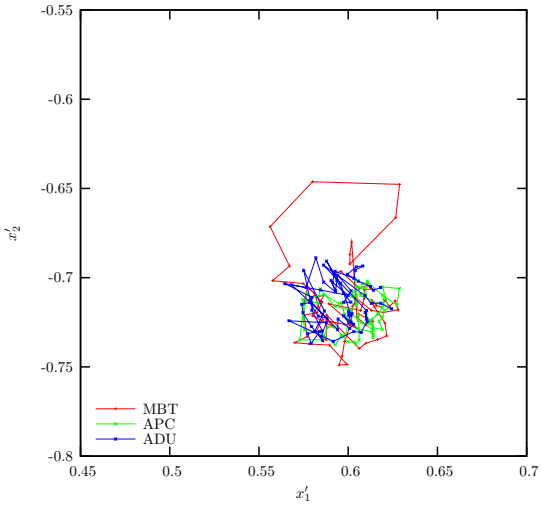
Figure 5.18: *Data sets used in training the RBFN with unsupervised loss function, projected to two dimensions. Averaged data are represented by points joined by solid lines, their non-averaged counterparts are represented as single dots.*



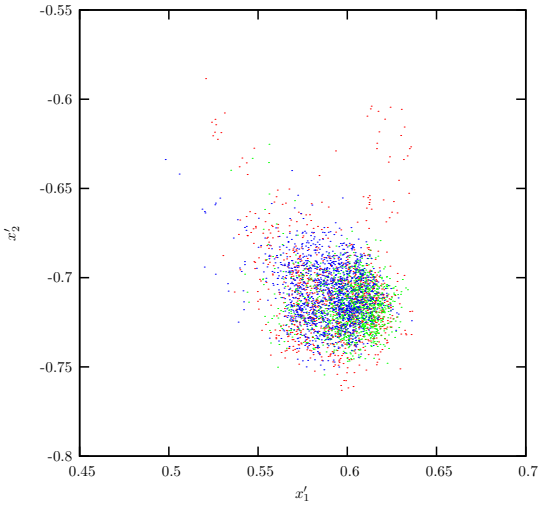
(a) *Averaged training data, inter-class distance.*



(b) *Non-averaged training data, inter-class distance.*



(c) *Averaged training data, intra-class distance.*



(d) *Non-averaged training data, intra-class distance.*

Figure 5.19: *Projections of averaged and non-averaged training data to two dimensions using supervised loss function for $\rho = 0$, responding to inter-class distances only, and $\rho = 1$, responding to intra-class distances only. Averaged data points are shown with solid lines joining successive data points, non-averaged data points are shown as single dots. The order of printing is red, green, then blue; with later points overplotting previous points.*

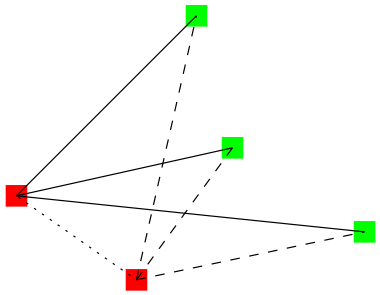


Figure 5.20: *Specifying distances between one of the red class points and the green class points (solid lines) and between another red class point and the green class points (dashed lines), places constraints on the distance between the two red class points (dotted line).*

after training with $\rho = 1$ (intra-class distances). The projections learned using $\rho = 1$ exhibit more structure reminiscent of that extracted using the unsupervised loss function as shown in the projections in figure 5.18.

It is interesting to note that, although the projections using $\rho = 0$ are not learned using any knowledge of intra-class distances; there is structure present within each projected class — each class does not tend towards a singularity. This is because the act of specifying distances between points of differing classes implicitly places constraints on the distances between points of the same class. This concept is illustrated graphically in figure 5.20.

Rescaled inter-class distances

In the previous section, the same network parameters determined for the unsupervised loss were assumed to be reasonable. This was because the data structure was the same, only differing aspects of it were selected as the parameter ρ was varied. In this section, the distances between points belonging to different classes are rescaled as described in section 3.3.2. This effectively alters the relationships being modelled between data points. As the data transformation being trained is now required to learn a very different projection than before, it is wise to perform another round of parameter selection.

Histograms of the minimum distances from data points to their nearest centres, when using 60 centres, are shown in figure 5.21. These distances are independent of the rescaling and naturally are very similar to the distances obtained using the same data but 40 centres as shown previously in figure 5.15. The histograms for non-averaged data again suggest that $h^2 = 0.001$ should coincide with a rapid drop in stress for non-averaged data.

The surface plots of stress as a function of number of centres and h^2 are presented in figure 5.22. For less than 40 centres, the stress decreases as the number of centres increases. By 60 centres the stress has largely levelled off. Figure 5.23 plots the stress obtained when projecting to two dimensions using 60 centres. The greatest reduction of stress for non-averaged data does indeed occur around $h^2 = 0.001$. The stress pertaining to

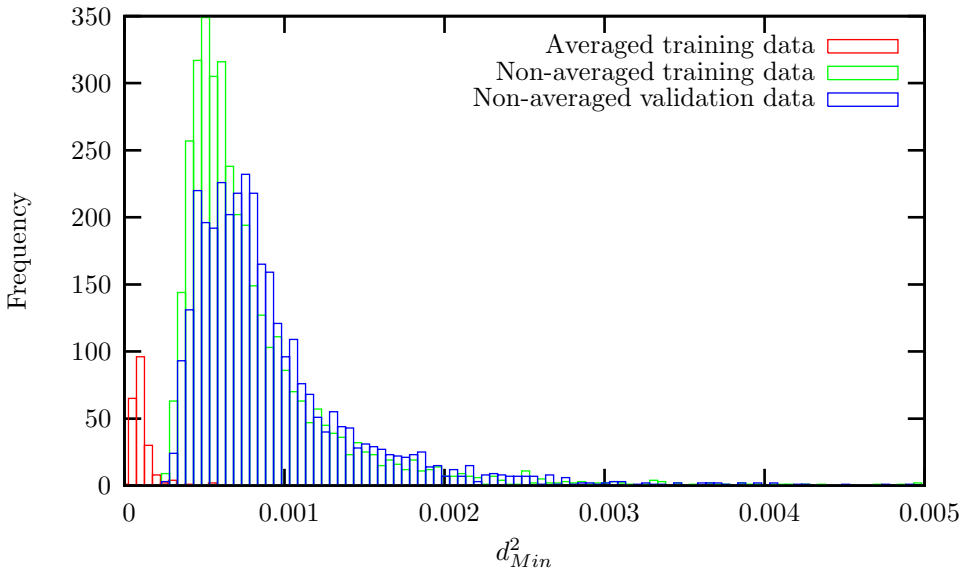
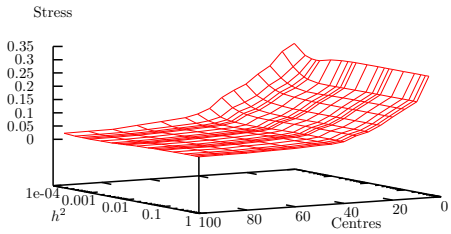
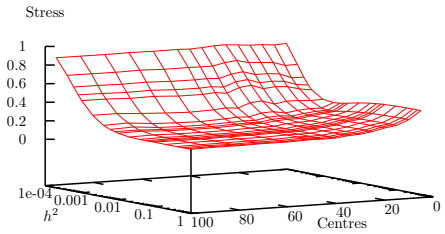


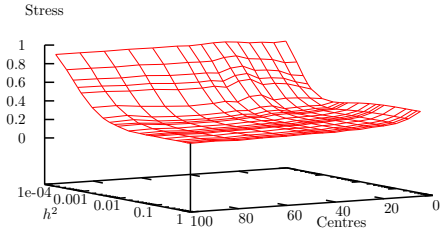
Figure 5.21: *Histogram of frequencies of minimum distances between data points and RBF centres when using 60 centres. The frequency is the number of samples falling within a given bin. The binsize is* 0.00005*. Histograms for averaged training data, non-averaged training data, and non-averaged validation data are shown.*



(a) *Averaged training data*



(b) *Non-averaged training data*



(c) *Non-averaged validation data*

Figure 5.22: *Stress calculated when data are projected to two dimensions by RBFN trained using rescaled inter-class distances.*

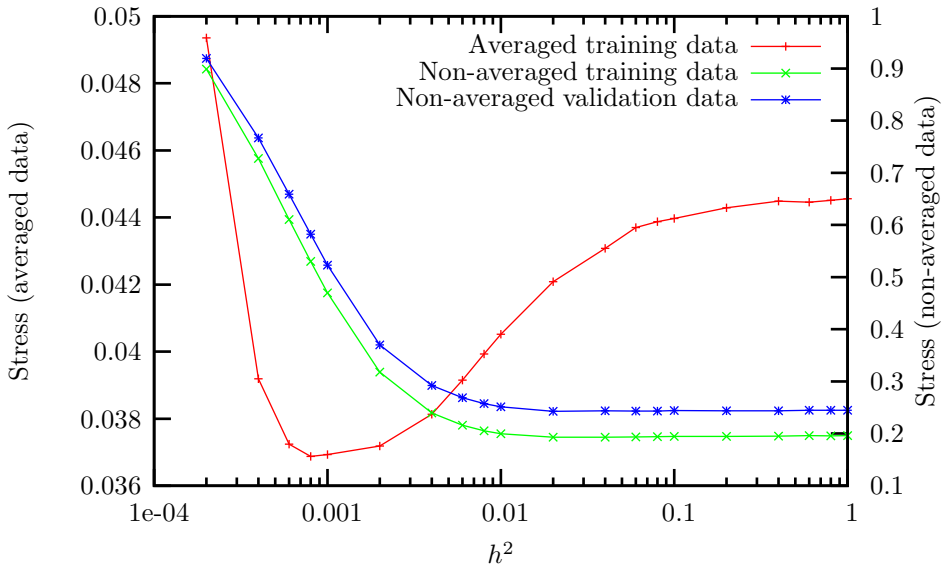


Figure 5.23: *Values of stress derived for projections to two dimensions using 60 centres. Plots are shown for the resultant stress after training on the averaged training data, and the stress subsequently calculated for the non-averaged training and validation data sets.*

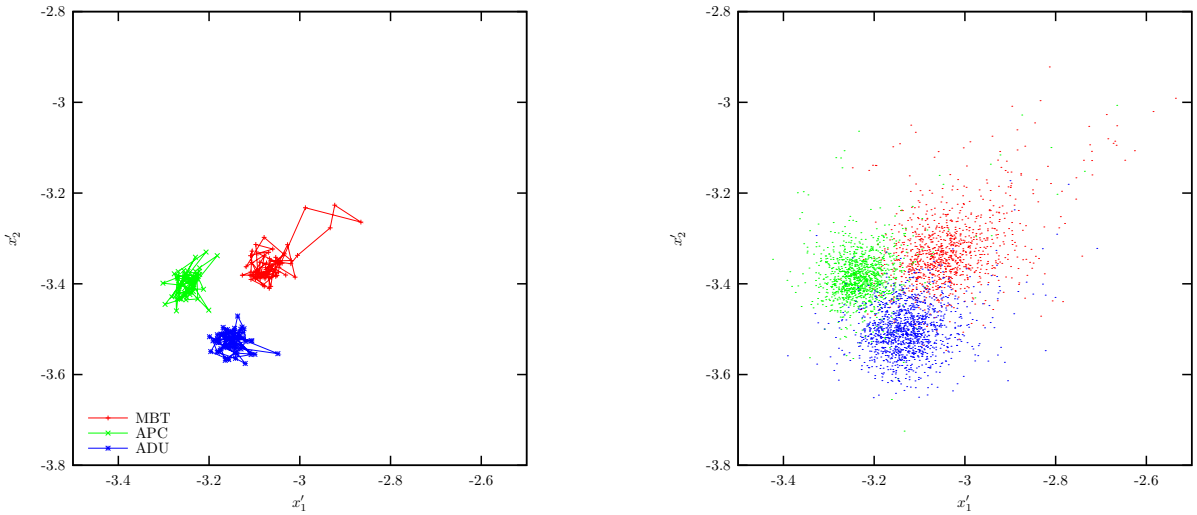
the non-averaged data has levelled off by $h^2 \approx 0.01$. By previous argument, using $h^2 = 0.1$ should only add an offset to the projections but will help ensure that non-averaged data points are encompassed by the basis functions. Thus 60 centres and $h^2 = 0.1$ are used for the choices of parameters.

Projections of the training data to two dimensions, by an RBFN duly trained using these parameters, are shown in figure 5.24. The averaged data points, used to train the network, shown in figure 5.24a, are well separated into their three distinct classes. Unlike the projections using only inter-class distance, shown in figure 5.19a, these projections demonstrate better internal structure (particularly the MBT class) as well as better class separability. The projected non-averaged data, shown in figure 5.24b, show greater spread but still form three largely separable classes.

The rescaling of the distances between data points effectively stretches, or warps, the subspace within which the data live, flattening it out into a two-dimensional surface. It is edifying that this is reminiscent of the early days of MDS, where reduced-dimension (usually two) representations of data were sought by scaling distances between points so as to deliberately flatten the embedding space and thus obtain the required low-dimensional representation [102].

Distances calculated using Box-Cox metric

This approach, described in section 3.3.3, takes the data 'as is'. It does not perform any pre-processing upon the data values but uses a different metric for calculating distances



(a) *Averaged training data*

(b) *Non-averaged training data*

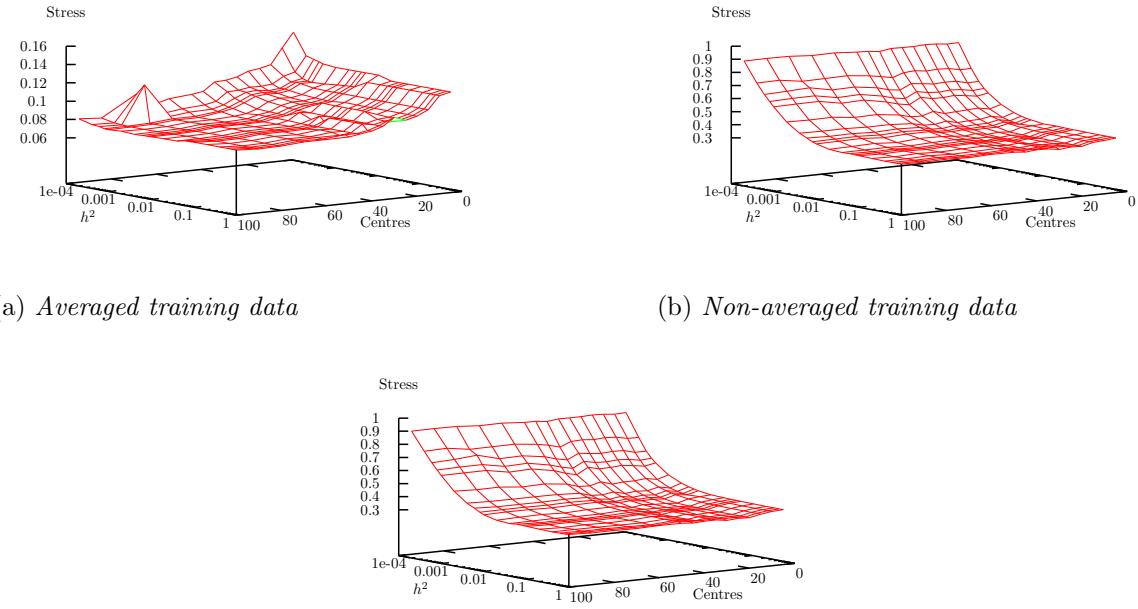
Figure 5.24: *Projection to two dimensions of the training data after network training using rescaled inter-class distances.*

between points. Although this was split into a 'transformation' component and a 'metric' component in (3.26) and (3.27), these two equations can be regarded as together defining the metric. The transformation (3.26) is only used in conjunction with the metric (3.27). As far as the RBFN is concerned, its input comprises non-transformed data. This can be interpreted as the network being asked to learn the Box-Cox transformation as a part of the nonlinear dimensionality-reduction transformation.

The input data are the same as used in the preceding sections and so similar network parameters are expected to be suitable here. Stress-surfaces resulting from grid search over number-of-centres and h^2 confirm this and are given in figure 5.25. Once again 40 centres with $h^2 = 0.1$ appear to be reasonable choices. Using these parameters and training on averaged data points, produces the projected-training-data plots shown in figure 5.26.

Box-Cox transformation applied to data

Instead of attempting to learn the Box-Cox transformation as part of the nonlinear projection, using training data that inevitably only span a subset of the input space; an alternative is to perform the Box-Cox transformation (3.26) globally on all of input space, and then learn projections that capture Euclidean structure in this transformed space. This pre-processing of the data acts on the data before they are input to the RBFN and so model selection must be performed anew to discover suitable parameters for projecting the Box-Cox-transformed input-data.

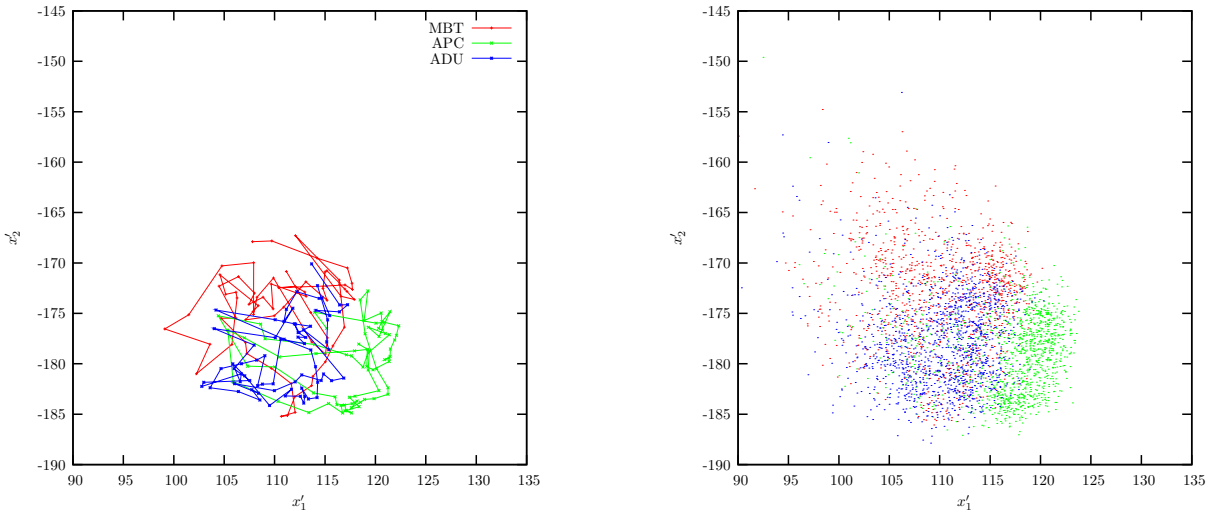


(a) *Averaged training data*

(b) *Non-averaged training data*

(c) *Non-averaged validation data*

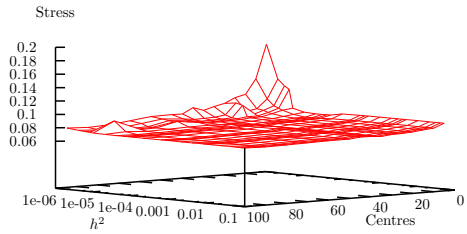
Figure 5.25: *Stress calculated when data are projected to two dimensions by RBFN trained using distances calculated with the Box-Cox metric* ($t = 0$).



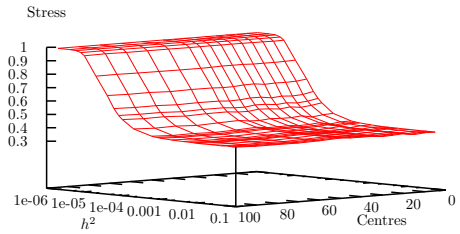
(a) *All three averaged training data sets.*

(b) *All three non-averaged training data sets.*

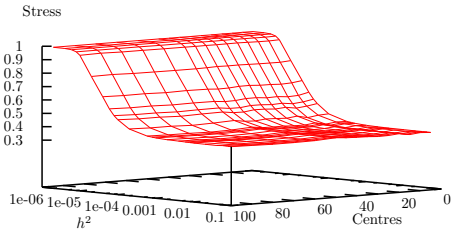
Figure 5.26: *Projections of training data to two dimensions, after unsupervised loss function trained using Box-Cox distances between points.*



(a) *Averaged training data*



(b) *Non-averaged training data*



(c) *Non-averaged validation data*

Figure 5.27: *Stress calculated when data are projected to two dimensions by RBFN trained after pre-processing the data with the Box-Cox transform* ($t = 0$).

Surfaces of stress as a function of both number of centres and h^2 are presented in figure 5.27. The response of the stress is rather flat with regards the number of centres. The dominant parameter is the width of the basis functions. Using 40 centres again seems reasonable and results in reasonably rapid training. A plot of the distributions of minimum distances from data points to RBF centres is given in figure 5.28. The distributions of distances for non-averaged data points suggest here that $h^2 \approx$ 4E-5 should coincide with a rapid decrease in stress against non-averaged data. This is borne out in figure 5.29, where stress is plotted as a function of h^2 when using 40 centres. In this plot, the stress for non-averaged data has levelled off by $h^2 \approx 0.001$.

In keeping with the previous argument in favour of using this value as a minimum, and that the independence of stress to h^2 suggests larger values are acceptable, we may choose to use a larger value such as 0.01. However, by the same argument the value of 0.1 is justifiable and this has the attraction of being the same value as used for the previous Box-Cox experiments using only the Box-Cox metric. The use of the same model parameters for the two sets of experiments does have some appeal for comparing the two. The final set of parameter values chosen was thus 40 centres and $h^2 = 0.1$.

Once again a network was trained on the averaged training data using these chosen network parameters and used to project the training data to two dimensions. The resulting

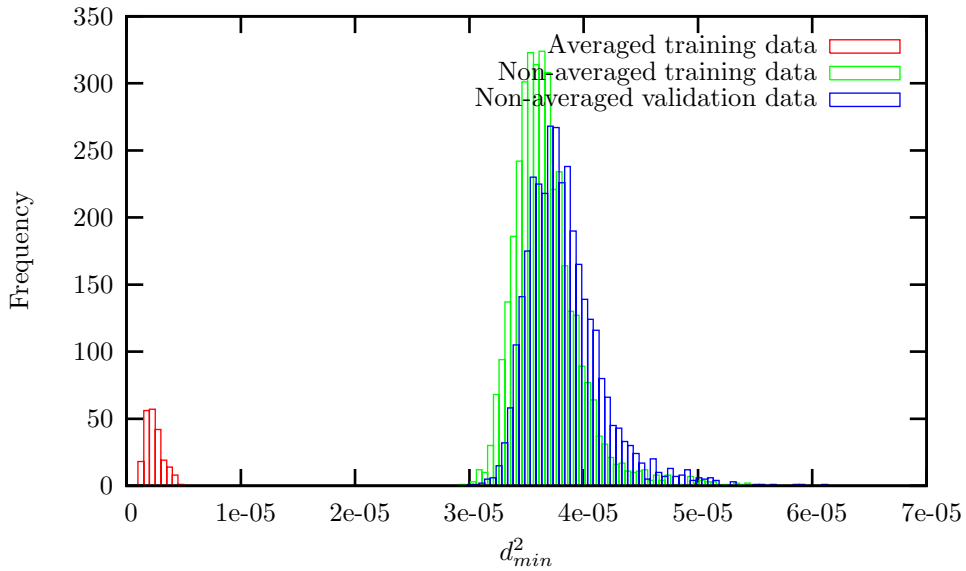


Figure 5.28: *Histogram of the frequencies of minimum distances between data points and RBF centres for Box-Cox transformed data when using 40 centres. The frequency is the number of samples falling within a given bin. The binsize is 5E-07. Histograms for averaged training data, non-averaged training data, and non-averaged validation data are shown.*

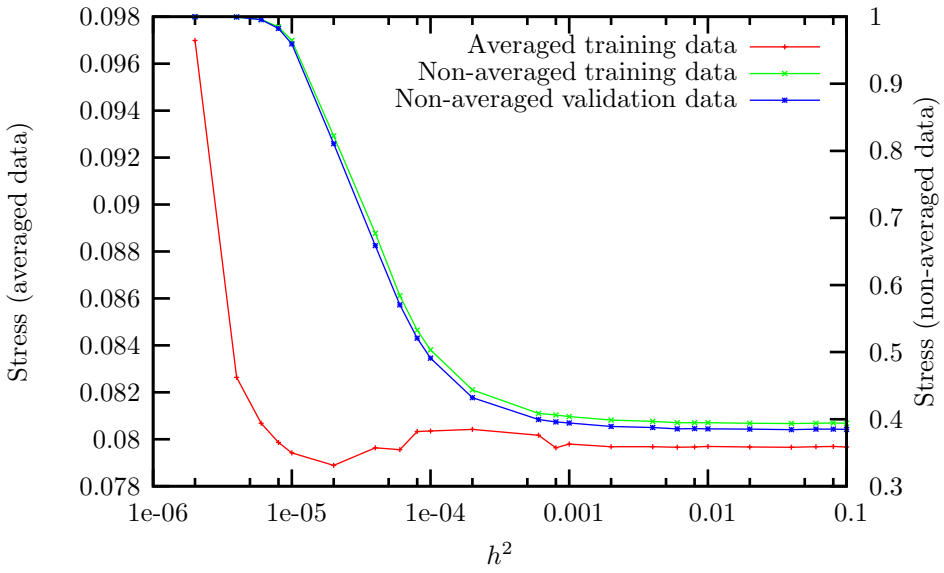
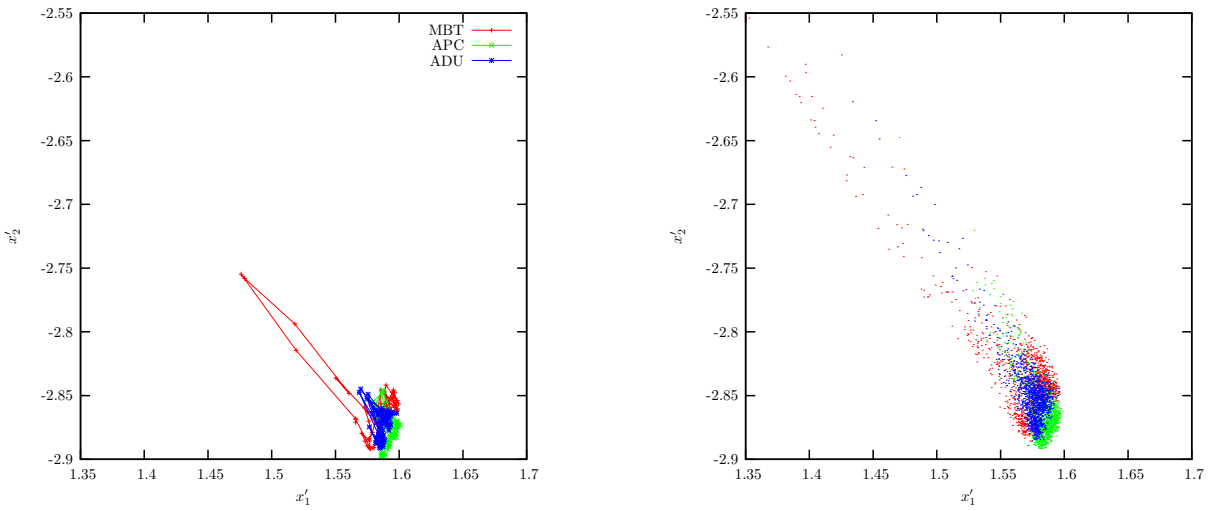


Figure 5.29: *Stress derived for projections of Box-Cox transformed data to two dimensions using 40 centres. Plots are shown for the resultant stress after training on averaged training data (left axis), and the stress subsequently calculated for the non-averaged training and validation data (right axis).*



(a) *All three averaged training data sets.* (b) *All three non-averaged training data sets.*

Figure 5.30: *Projections of training data to two dimensions, after application of the Box-Cox transformation, and then training the RBFN using the unsupervised loss function and distances calculated by the Euclidean metric. The order of plotting is red, green, blue; later plots overwrite previous plots.*

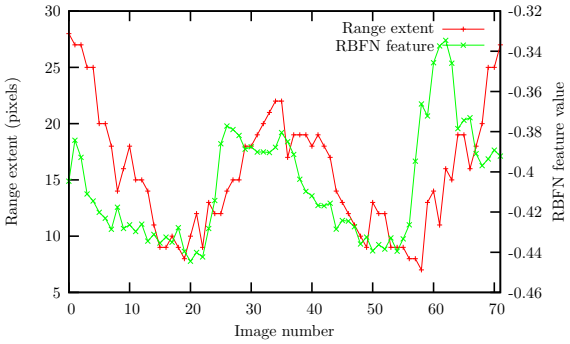
projections are given in figure 5.30. The structure in the data appears very different to that captured using the Box-Cox metric for calculating distances as shown previously in figure 5.26.

5.5.3 An interesting aside: data-driven feature extraction

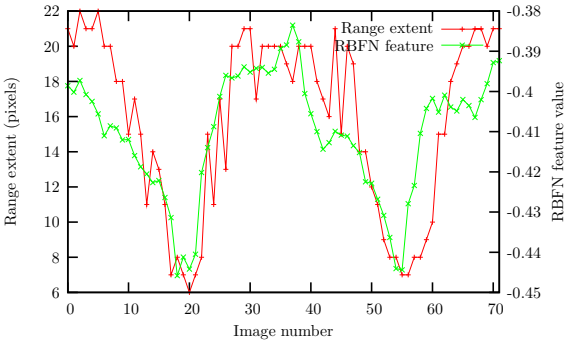
In this section we introduce supporting evidence for the RBFN extracting physically meaningful features.

The averaged training images, for each of the three classes of vehicle, were converted to binary images thresholded at the $\mu + 2s$ level, where μ is the mean pixel value and s the standard deviation of pixel values within each image. The range extent of each thresholded object was then calculated. These values were then compared with the corresponding RBFN-derived feature values as obtained by training an RBFN using 40 centres, a width of 0.1, and projecting to only one dimension. The comparisons are shown in figure 5.31a for the MBT vehicle, figure 5.31b for the APC vehicle, and figure 5.31c for the ADU vehicle. Visually, there are correspondences between the two curves for each vehicle, particularly for the APC in figure 5.31b.

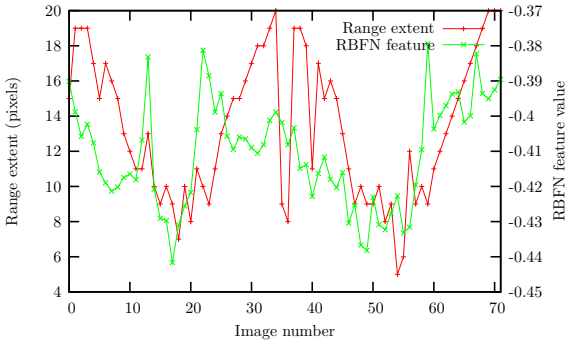
There is no reason to assume that the RBFN is implicitly calculating a radar length for each vehicle, nor that it is doing so in the same way as we have just done. Indeed, one could expend considerable effort in an attempt to determine an optimal 'length measuring'



(a) *MBT*



(b) *APC*



(c) *ADU*

Figure 5.31: *Comparison of a simple estimate of the radar range extent of vehicles with a single feature extracted using the RBFN. Averaged images were used and successive image numbers represent an increase in the aspect angle of the vehicles as they rotated.*

algorithm that best fits the feature calculated by the RBFN. Nevertheless, if we wished to derive a single number that best represented how each individual image contributed to the global structure of the data, the apparent contraction and expansion of range extent as the vehicles orient first their length and then their width to the radar as they rotate would be foremost in our minds as a reasonable starting point. These results encourage the hypothesis that the RBFN is picking up on just such a feature, or something very similar, and that the approach of using an RBFN to minimize the stress (3.6) is capable of extracting meaningful features from the data.

5.6 Summary

The two techniques for performing linear feature extraction, PCA and LDA, have very different optimization criteria. Whilst PCA is tasked with simply capturing data structure, in the sense of maximizing variance or minimizing the square errors, with no concept of distinct data classes; LDA is aware of the class labels of the training data and seeks to obtain a transformation that maximizes the variance between data belonging to different classes, with a constraint on the variance within classes.

These different goals were reflected in the observed projections of the training data onto feature subspaces. The PCA projections to two and three dimensions, shown in figures 5.4, 5.5, and 5.6, appeared to pick up on structure associated with the changing aspect angle of the targets, with little emphasis on the separability of classes. The LDA projection to two dimensions, shown in figure 5.13, did separate the different classes present in training, but this lost any discernable structure within a class.

The eigenvectors derived using PCA, shown in figure 5.3, placed emphasis on regions of the image containing the target. The eigenvectors derived using LDA, shown in figure 5.12 did practically the opposite with the only emphasis seemingly the radial extent of the target.

The classification results presented in section 5.4 showed that the error rates when training and testing on the same vehicles were, in general, much lower than those obtained when the training and testing vehicles differed. This is readily apparent when comparing the results shown in figures 5.14a and 5.14b for Groups 1 and 2, respectively, with those shown in figures 5.14c–5.14h for Groups 3–5, respectively.

We concentrate now on the results for Groups 1 and 2, shown in figures 5.14a and 5.14b. The inclusion of shifting of a test image in order to try and find a best match between train and test examples was of most benefit when classification was performed in the image (full dimension) space. It was often detrimental when applied prior to feature extraction. In terms of extracting features for classification, PCA defined a subspace

that generally matched the structure present in image space. LDA produced a subspace that produced many noticeably poorer classification results albeit with a small number of noticeably better results.

It should be stressed that these results are not presented with the claim of them being representative of 'state of the art'. They are given to give some context both to how easy or difficult the particular task at hand is, and how two standard linear techniques for feature extraction perform relative to each other and to full dimension data. What is clear is that obtaining a degree of classifier-robustness to changes in the configurations of vehicles seems a relatively easy problem compared to the hard one of generalizing an arbitrary class across different vehicle types. This is encouraging at least in that it suggests it may not be crucial to train an algorithm on large numbers of possible target articulations in order to recognize future observations at different articulations.

This chapter made theoretical predictions about the effect of the RBFN parameters when applied to the radar data. The predictions were borne out by pilot studies applying the RBFN to the radar data. The distributions of the minimum distances between RBF centres and data points provided guidance in the choice of model parameter values. Final choice of parameter values was based upon the values of stress calculated on non-averaged data points. The different modifications to how the network was trained resulted in projections often looking quite different, but with some common aspects of structure. The most promising projection to two dimensions, in terms of offering separability of the classes, was that using rescaled inter-class distances during training, which forced the classes apart in feature space. Otherwise there tended to be little separability of the classes in two dimensions — most likely because the data did not inherently live in a two-dimensional subspace of the original input space.

It is a non-trivial exercise to attempt to interpret the features being calculated by such a nonlinear projection of the data as realized using the RBFN. Nevertheless, a comparison of one-dimensional RBFN features with a simple calculation of vehicle radar-length showed some resemblance between the two. This offered encouragement that the RBFN was extracting physically meaningful features from the data, but was doing so in an entirely data-driven fashion — the features were not specified *a priori*.

This chapter has laid the groundwork for the next, where the RBFN will be used to project data prior to classification being performed in feature space. Such groundwork included the identification of suitable network parameters for each modification of the network. It is unlikely that relevant data structure can be captured by just two dimensions, a suspicion fed by the projections of the training data to two dimensions. As the dimensionality of the feature space is increased, separability of the classes will be expected to increase, up to a point, as the additional dimensions pull in more of the inherent data

structure. Thus a focus of the next chapter will be to examine how the classification error rate varies with feature-space dimension, using the RBFN parameters determined in this chapter.

Chapter 6

Radial basis function results

6.1 Introduction

This chapter introduces and discusses classification results obtained using RBFN-derived features as input to the 1-NNR classifier. In keeping with the presentation of previous results, error rate is given for each data set:

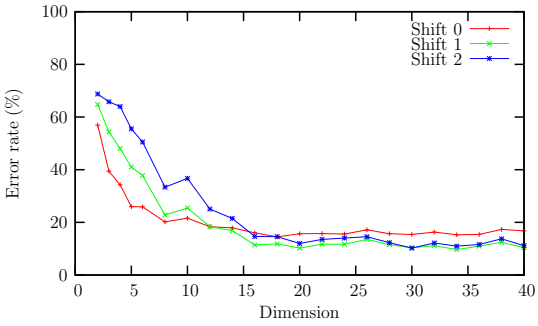
- as a function of the number of features used,
- for not allowing shifting of the original images and allowing shifting of the original images,
- and summarized using a 20 dimensional feature-space for comparison of results across data sets.

The small training set was used for generating the results. Results for features obtained using the unsupervised loss function are presented in section 6.2, the supervised loss function in section 6.3, rescaled inter-class distances in section 6.4, Box-Cox distances in section 6.5, and Box-Cox transformed data in section 6.6. Finally, a summary of the chapter is given in section 6.7

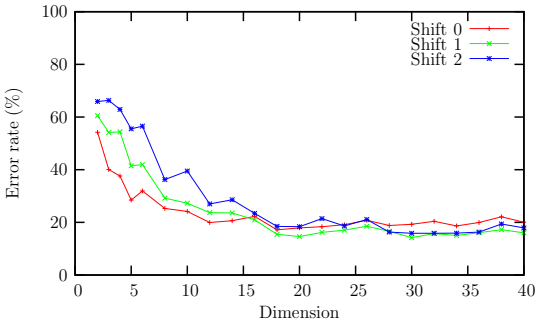
6.2 Unsupervised loss function

6.2.1 Error rate with number of features

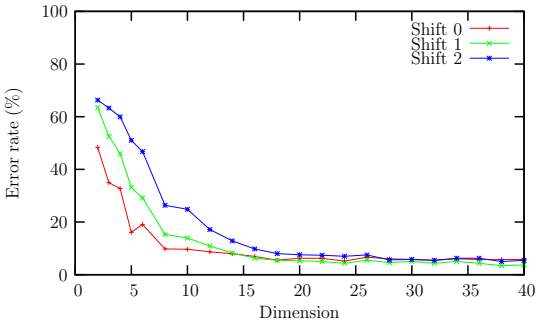
Plots of error rate against feature-space dimension for Group 1 are given in figure 6.1. The error rates start high at low feature-space dimensions and fall rapidly with increasing dimension. The decrease in error rate levels off by 20 dimensions. The amount of shifting performed on the input images has the greatest effect on error rate at the lower dimensions.



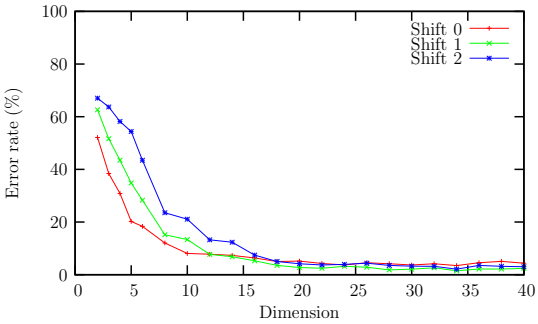
(a) *MBT1-3*



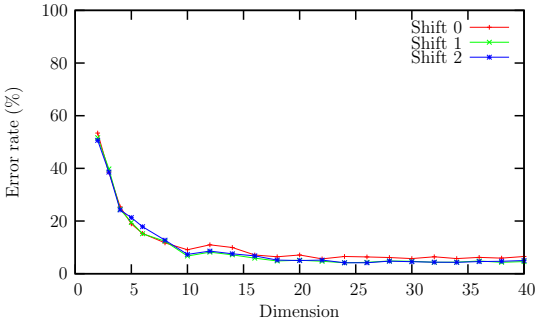
(b) *MBT1-4*



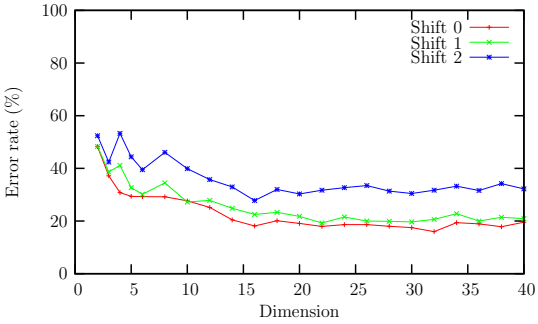
(c) *MBT1-5*



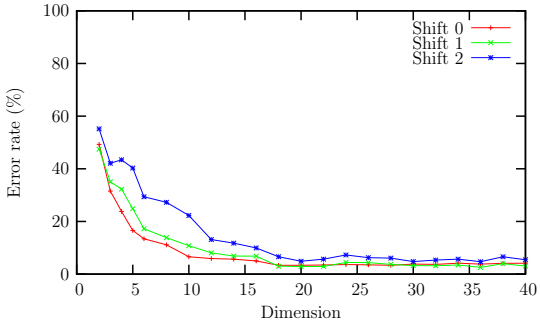
(d) *MBT1-7*



(e) *APC1-2*



(f) *ADU1-3*



(g) *ADU1-5*

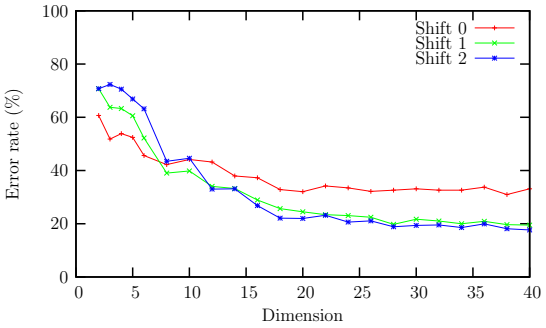
Figure 6.1: *Group 1 test results, with shifting, for RBF projection trained using unsupervised loss function.*

Broadly, the trend at low dimension is for no-shifting to give the lowest error rate, a shift of ± 1 pixel to give the next lowest, and a shift of ± 2 pixels to give the highest error rate. When the error rate has leveled off, by 20 dimensions, this pattern no longer holds; there is less margin between the different shifting regimes and the curves often cross over, changing the rank ordering of the error rates. Exceptions to this pattern are APC1-2 in figure 6.1e, for which the curves for different shifts remain very similar, and ADU1-3 in figure 6.1f, which shows little reduction in the error rate with increasing dimension and the greatest consistent separation of the curves for different shifts.

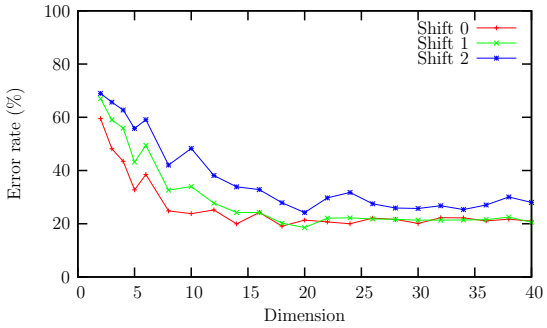
Plots of error rate against feature-space dimension for Group 2 are given in figure 6.2. The error rates start high at low dimensions and fall rapidly with increasing dimension. Again, it is a data set associated with ADU1, namely ADU1-4 shown in figure 6.2d, that least follows this pattern, with little change in error rate as the dimension increases. The error rate for all data sets levels off by 20 dimensions. For small feature-space dimensions, the lowest error rate is typically obtained when no shifting is applied to the input images, the next lowest when a shift of ± 1 pixels is applied, and the highest when a shift of ± 2 pixels is applied. Several data sets — MBT1-6 in figure 6.2b, ADU1-6 in figure 6.2e, and ADU1-7 in figure 6.2f — broadly retain this rank ordering of the curves with increasing dimension. APC1-3, shown in figure 6.2c, shows very similar curves for the different amounts of shift. MBT1-2, shown in figure 6.2a, is a clear instance of no-shift giving rise to the lowest error rate at low dimension but the highest error rate at higher dimension (above 10 dimensions).

Plots of error rate against feature-space dimension for Group 3 are given in figure 6.3. The only discernible decreasing trend is for the data sets of MBT2, namely MBT2-2 in figure 6.3a and MBT2-3 in figure 6.3b. The trend, however, is not as strong as for the data sets in the first two groups and the error rate remains high, although it does drop below the 70% random-chance level by 20 dimensions at which point it levels off. The effect of shifting on these two data sets is mixed, with no-shift giving rise to the lowest error rate across all dimensions for MBT2-2 but only for dimensions below 20 for MBT2-3, above which it gives the highest error rate. The differences in error rate between the different amounts of shifting are, however, often small. Other than for MBT2, the data sets are generally characterized by an error rate that is at least as high as that that would be obtained by random chance and remains largely independent of feature-space dimension. The only exception is ADU3-1, shown in figure 6.3j, which has an error rate that drops further below the level of random chance as the amount of shifting increases.

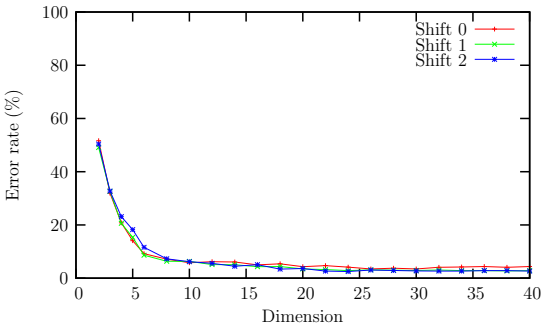
Plots of error rate against feature-space dimension for Group 4 are given in figure 6.4. The only discernible decrease in error rate with increasing dimension is for MBT2, namely MBT2-4 in figure 6.4a, MBT2-5 in figure 6.4b, and MBT2-6 in figure 6.4c. Again this



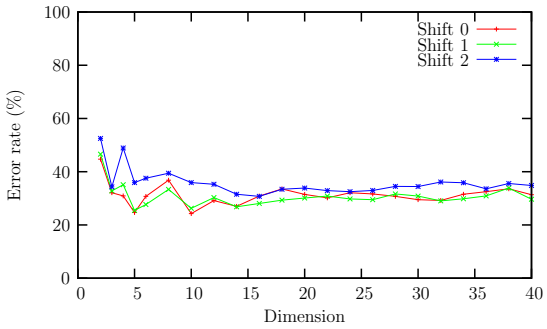
(a) *MBT1-2*



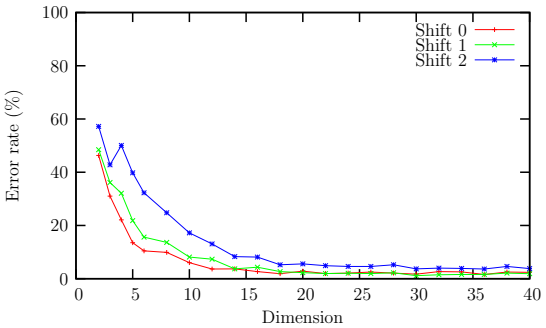
(b) *MBT1-6*



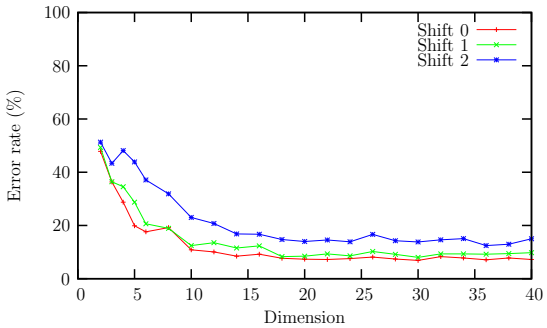
(c) *APC1-3*



(d) *ADU1-4*

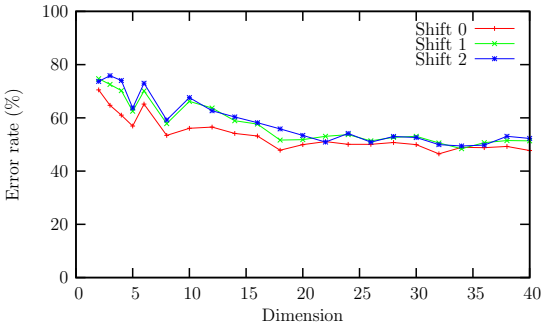


(e) *ADU1-6*

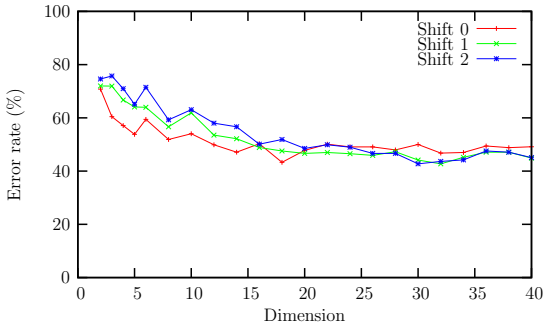


(f) *ADU1-7*

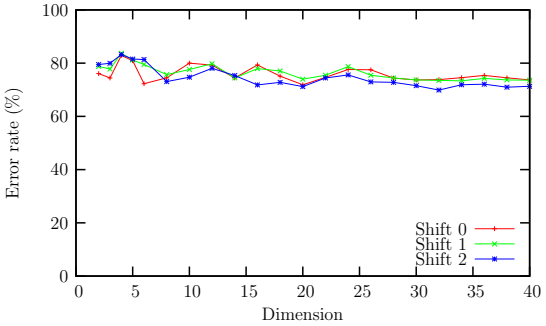
Figure 6.2: *Group 2 test results, with shifting, for RBF projection trained using unsupervised loss function.*



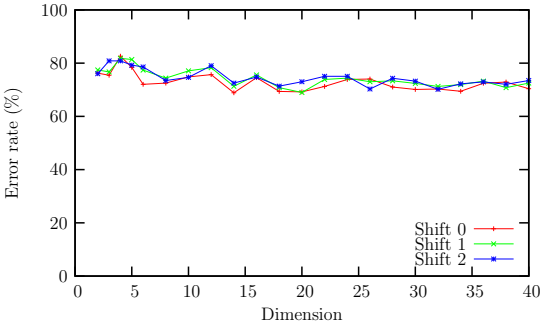
(a) *MBT2-2*



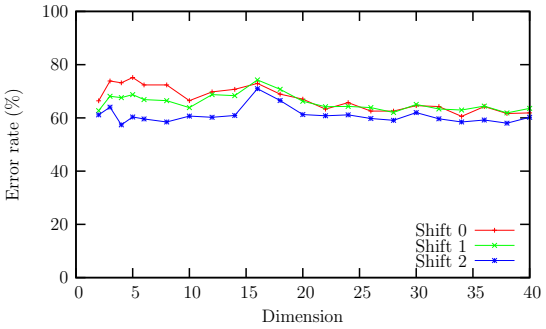
(b) *MBT2-3*



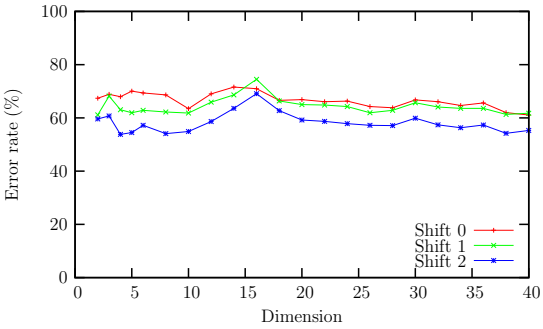
(c) *MBT3-3*



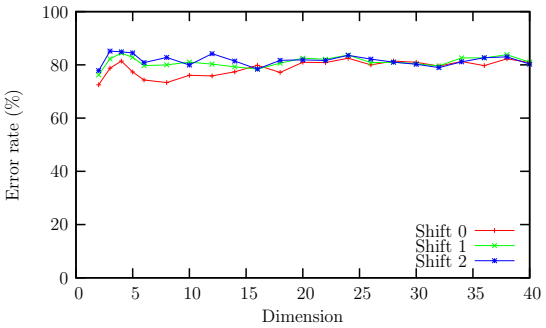
(d) *MBT3-6*



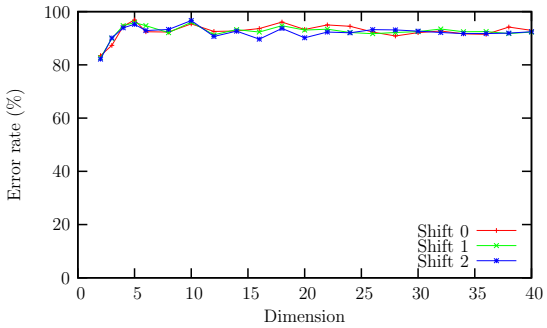
(e) *APC2-2*



(f) *APC2-3*

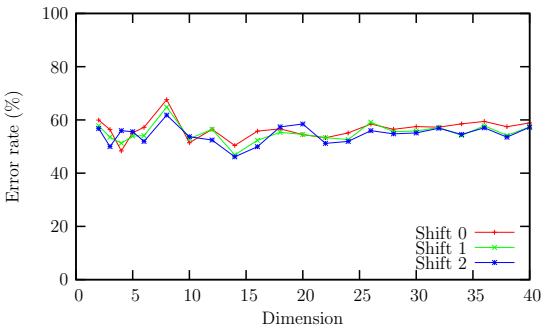


(g) *MBT4-3*

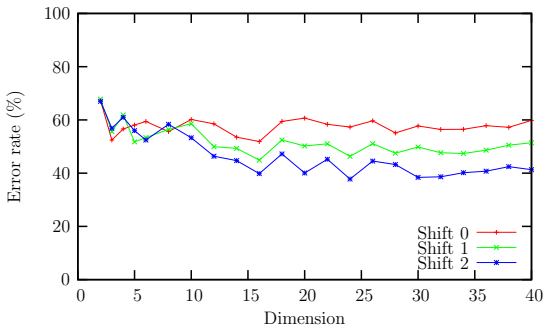


(h) *MBT5-3*

Figure 6.3: *Group 3 test results, with shifting, for RBF projection trained using unsupervised loss function.*



(i) *ADU2-1*



(j) *ADU3-1*

Figure 6.3: *Group 3 test results, with shifting, for RBF projection trained using unsupervised loss function (cont.)*

trend is not as strong as for the first two groups but does take the error rate below the level of random chance. The effect of shifting is also mixed for these data sets. No-shift produces the lowest error rate for MBT2-4 at low dimensions, but the highest error rate above approximately 20–25 dimensions. No-shift consistently produces the lowest error rate across all dimensions for MBT2-5. All other data sets have error rates approximately at, or above, the level expected for random chance.

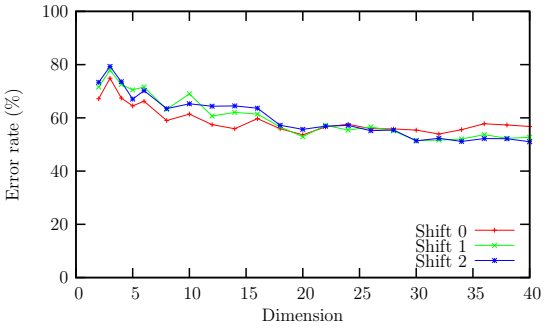
Plots of error rate against feature-space dimension for Group 5 are given in figure 6.5. There are no data sets that demonstrate a trend in the error rate with respect to feature-space dimension, nothing drops below the level of random chance, and the different degrees of shifting produce very similar results.

6.2.2 Summary of results

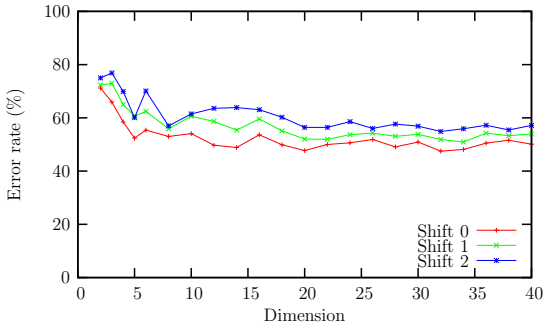
For comparison of results across data sets, the error rate is plotted for the 1-NNR using 20 features, for each of the data sets, in figure 6.6.

The results for Group 1 are shown in figure 6.6a. The data sets can be broadly partitioned into two groups based on error rate. The error rate is particularly low for MBT1-5, MBT1-7, APC1-2, and ADU1-5. Much larger error rates are obtained for MBT1-3, MBT1-4, and ADU1-3. This split is not according to vehicle type: MBT1 and ADU1 are on both sides of the split. The inclusion of a shift of ± 1 pixel mostly leads to a reduction in error rate. This reduction is up to five percentage points for MBT1-3, but is typically approximately two or three. Increasing the amount of shift to ± 2 pixels never gives a further decrease in the error rate and usually causes an increase.

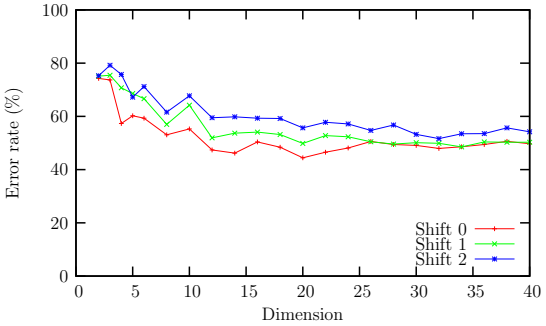
The results for Group 2 are shown in figure 6.6b. This group also possesses a large range of error rates. APC1-3 and ADU1-6 have particularly low error rates of less than 5%. ADU1-7 is a little higher, but MBT1-2, MBT1-6, and ADU1-4 are much higher.



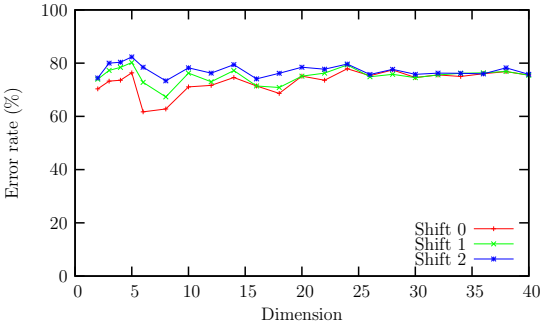
(a) *MBT2-4*



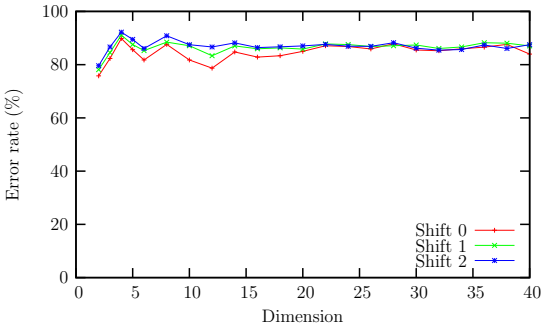
(b) *MBT2-5*



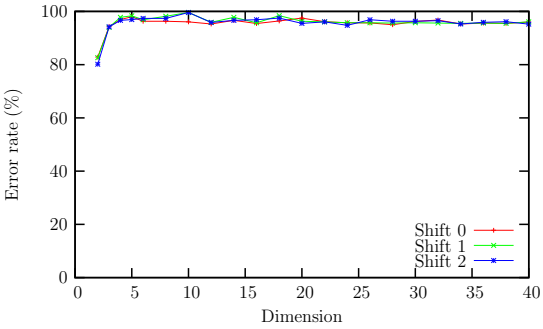
(c) *MBT2-6*



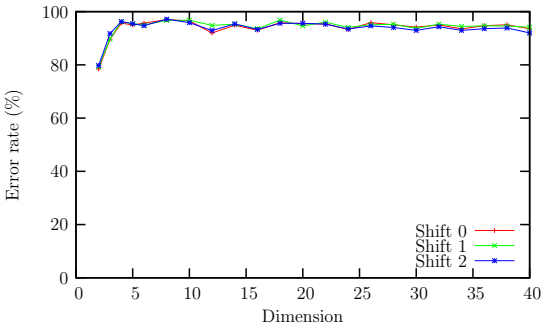
(d) *MBT4-1*



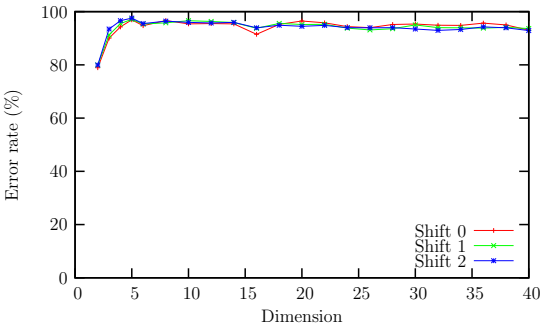
(e) *MBT4-2*



(f) *MBT5-4*

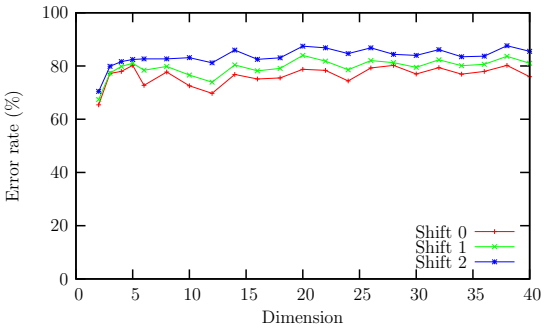


(g) *MBT5-5*

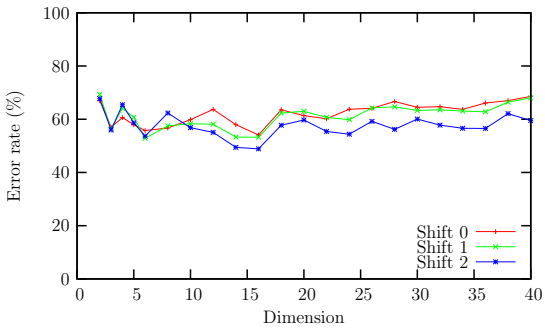


(h) *MBT5-6*

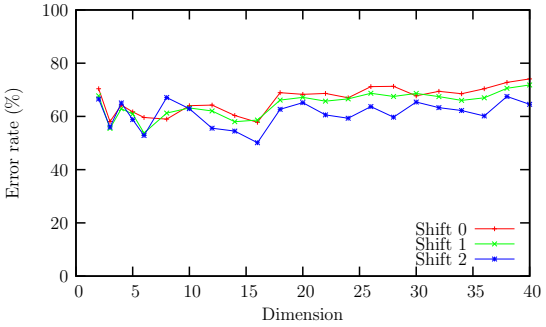
Figure 6.4: *Group 4 test results, with shifting, for RBF projection trained using unsupervised loss function.*



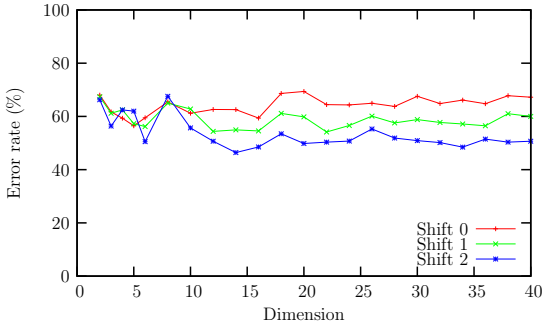
(i) *MBT4-4*



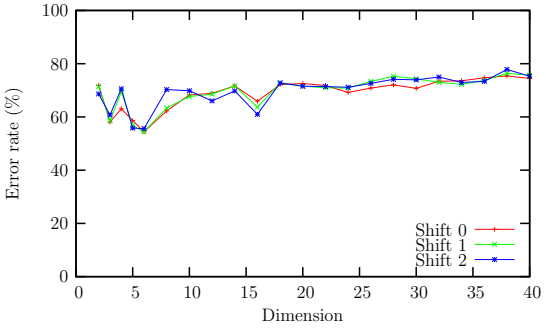
(j) *ADU3-2*



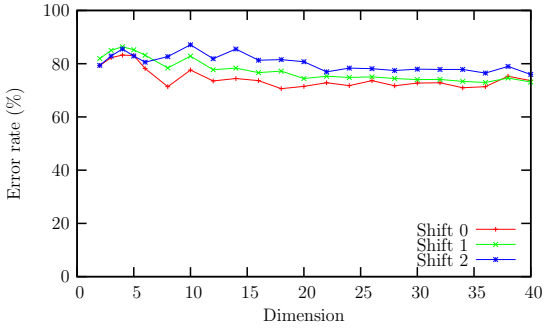
(k) *ADU3-3*



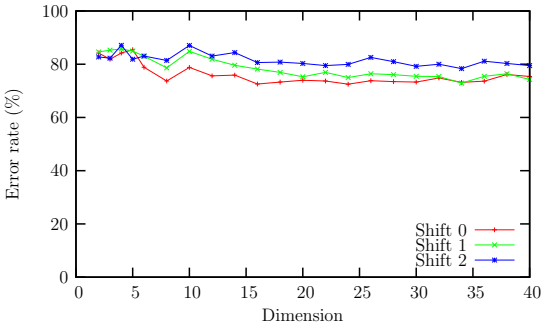
(l) *ADU3-4*



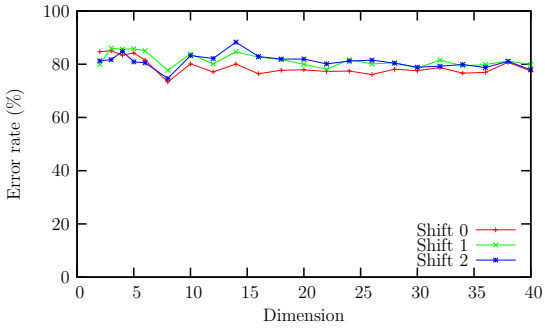
(m) *ADU3-5*



(n) *MBT3-7*

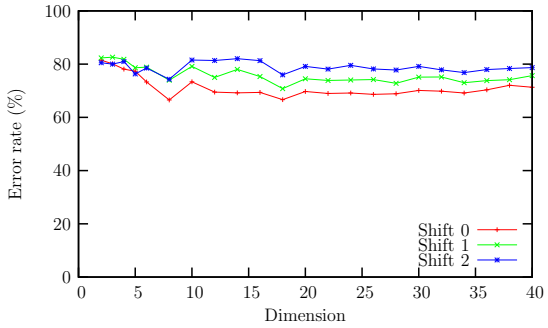


(o) *MBT3-8*



(p) *MBT3-9*

Figure 6.4: *Group 4 test results, with shifting, for RBF projection trained using unsupervised loss function (cont.)*



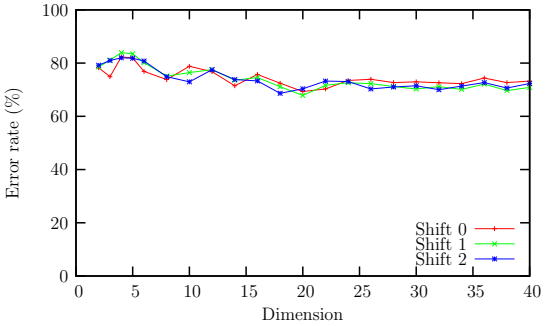
(q) *MBT3-10*

Figure 6.4: *Group 4 test results, with shifting, for RBF projection trained using unsupervised loss function (cont.)*

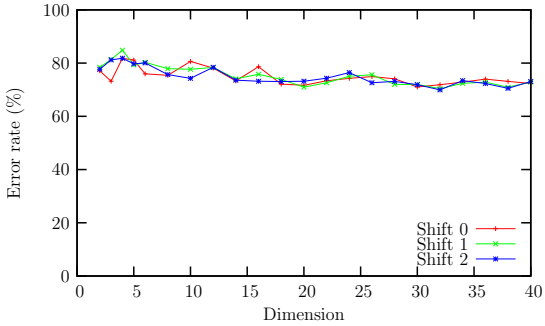
As with Group 1 this does not reflect different vehicles; ADU1 has both the lowest and highest error rates. A similar response to shifting as obtained for Group 1 is also evident. There is only one large drop with a shift of ±1 (MBT1-2); the changes for the other data sets are small. When a shift of ±2 pixels is performed, the most significant changes are increases in the error rates of up to five percentage points (e.g. MBT1-6 and ADU1-7). Only one data set, MBT1-2, has a further drop in error rate with shift ±2, most have a large increase, and APC1-3 is unchanged.

The results for Group 3 are shown in figures 6.6c and 6.6d. All error rates in this group are higher than in the first two groups. At approximately 50%, the error rates for MBT2-2 and MBT2-3 are amongst the lowest within the group. This is substantially better than the error rate that would be associated with random chance for the MBT class. The next lowest error rate is for ADU2-1 and ADU3-1, although when shifting is included the ADU3-1 error rate drops to approximately 40%. Performance on these two ADU vehicles is better than random chance, especially for ADU3-1 when shifting is included. Performance against the other data sets is approximately at the level of random chance, except MBT4-3 and MBT5-3, which exhibit error rates higher than for random chance. The inclusion of shifting does not provide a clear benefit, even for those data sets with a degree of discrimination that is better than random chance. ADU3-1 shows the greatest improvement in error rate with increasing amount of shifting, but results for the other data sets are mixed.

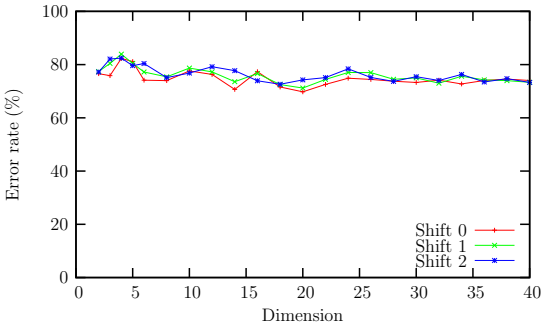
The results for Group 4 are shown in figures 6.6e through 6.6g. The only error rates notably better than random, with the exception of ADU3-4 when shifting is included, are for the MBT2 data sets: MBT2-4, MBT2-5, and MBT2-6 all shown in figure 6.6e. ADU3-4 starts off worse than random with no shifting, but each extra degree of shifting reduces the error rate by approximately 10 percentage points so that when shift ±2 is used the error rate is better than random. This pattern does not, however, extend to the other



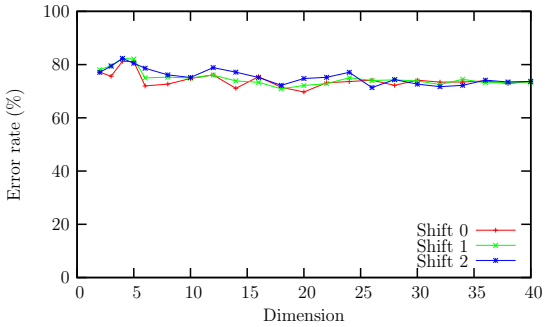
(a) *MBT3-1*



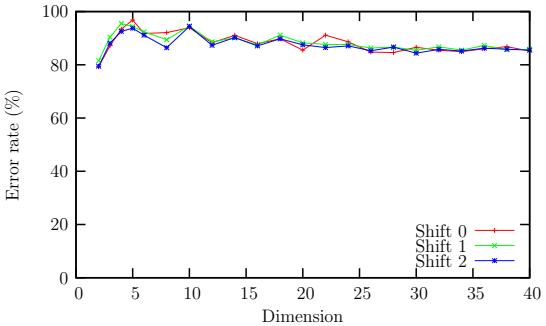
(b) *MBT3-2*



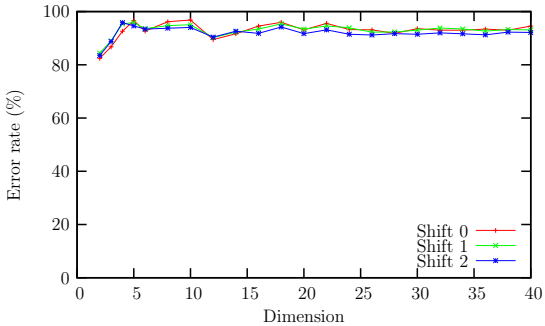
(c) *MBT3-4*



(d) *MBT3-5*

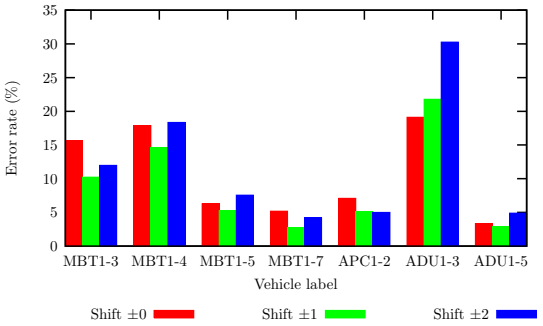


(e) *MBT5-1*

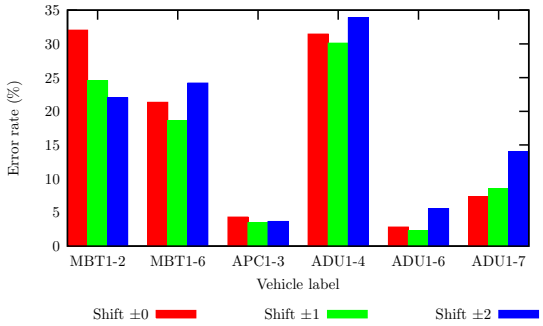


(f) *MBT5-2*

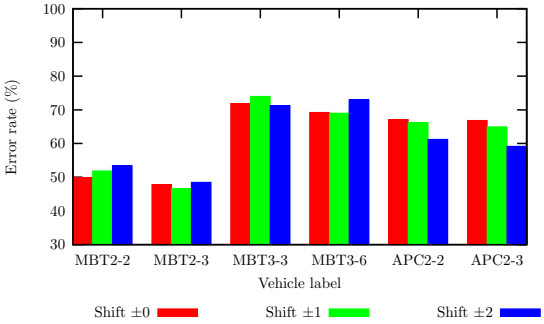
Figure 6.5: *Group 5 test results, with shifting, for RBF projection trained using unsupervised loss function (cont.)*



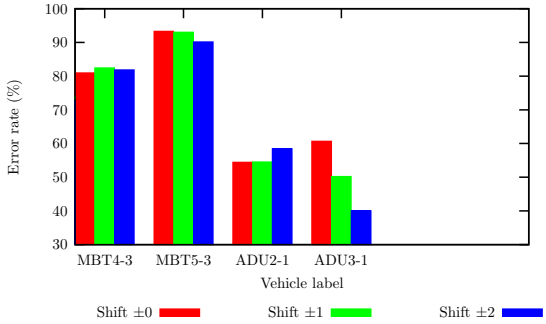
(a) *Group 1*



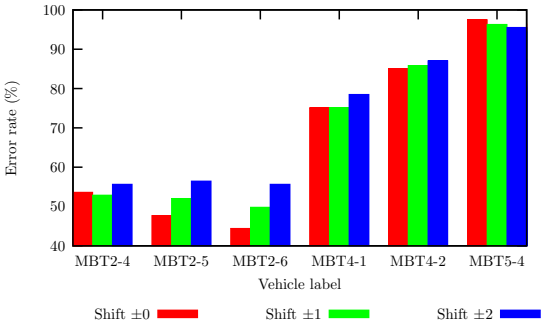
(b) *Group 2*



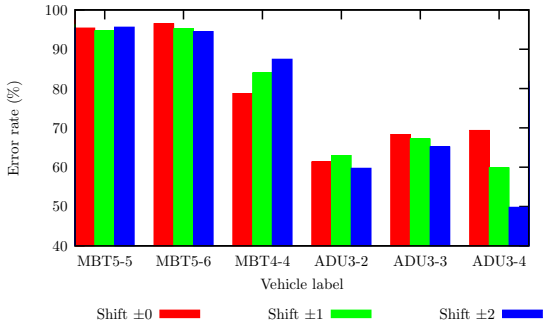
(c) *Group 3*



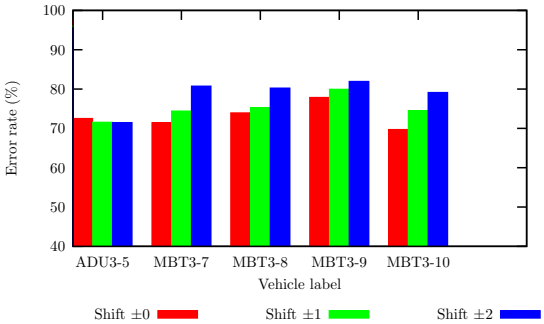
(d) *Group 3 (cont.)*



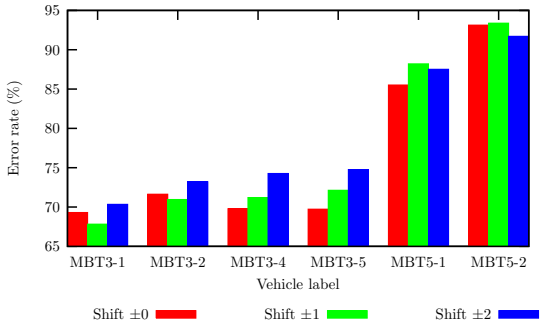
(e) *Group 4*



(f) *Group 4 (cont.)*



(g) *Group 4 (cont.)*



(h) *Group 5*

Figure 6.6: *Error rate from the 1-NNR using 20 features calculated by the RBFN trained using the unsupervised loss function. The larger groups (3 and 4) are split across subfigures, but the y-range is kept constant.*

ADU3 data sets in the group. ADU3-2, ADU3-3, and ADU3-5 all show performances of approximately random chance or worse. With the single exception of ADU3-4, the inclusion of shifting does not provide any useful improvement for the data sets in this group. On the contrary, the dominant effect of including shifting is an increase in error rate.

The results for Group 5 are shown in figure 6.6h. The data sets for MBT3 are approximately at the level of random chance and the data sets for MBT5 are much worse. The inclusion of shifting does not lead to a useful, consistent reduction in error rate.

6.2.3 Discussion

Error rates as a function of feature-space dimension for Groups 1 (figure 6.1) and 2 (figure 6.2) start high and fall rapidly as the dimension increases, levelling off by 20 dimensions. This is interpretable as there being initially too few features to discriminate between classes and as more features are added they add discriminatory information. Eventually all of the data structure relevant for discrimination is present in the extracted features and no further separation of the classes occurs.

In Group 3 the only discernible decrease in error rate with increasing dimension is for MBT2 (figures 6.3a and 6.3b), although the decrease is not as strong as that seen for the data sets in Groups 1 and 2. Interestingly, MBT2 achieves an error rate substantially lower than for random chance. These observations together suggest that some meaningful features are being extracted for this vehicle. The same comments apply to the MBT2 data sets in Group 4 (figures 6.4a through 6.4c).

Groups 1 and 2 comprised only the same physical vehicles that were used to create the training data. Groups 3, 4, and 5 comprised only vehicles that were not used to form the training data. Consistent with this distinction, the error rates seen in Groups 1 (figure 6.6a) and 2 (figure 6.6b) are lower than any of the error rates seen in Groups 3, 4, or 5 (figures 6.6c through 6.6h).

Whilst Groups 1 and 2 have the lowest error rates out of the five test groups, there is a spread of results within them (figures 6.6a and 6.6b). This spread is not explicable by vehicle-type alone.

Within Group 1, the lowest error rates are for MBT1-5, which had hatches open; MBT1-7, which had the engine running; APC1-2, which had hatches open; and ADU1-5, which had its gun barrels raised to 45°. The error rate more than doubles for MBT1-3 and MBT1-4, which had toolboxes open. The highest error rate in Group 1 belongs to ADU1-3, which had its gun turret rotated by 45°. The rank ordering of the results is consistent with the vehicle configuration changes. The hatches on the MBT are small and any shadow cast by them over the rest of the vehicle, or bright reflections from them, will

only occur over a small range of aspect angles. The hatches on the APC are large but located on the rear of the vehicle and so only visible from a small range of aspect angles. Furthermore, their effect should be local: they will not cast a shadow over the vehicle's surface in the way the top hatch on the MBT does. The toolboxes on the MBT have a greater effect. They are quite wide and located along the upper surface of one side of the vehicle, above the tracks. They can thus be expected to affect the signature of the upper surface of the vehicle over a wide range of aspect angles. The highest error rate is associated with the rotation of the ADU's gun turret. This turret is rectangular and large. Changing the orientation of this structure has a large impact on the vehicle's radar signature.

Within Group 2, the lowest error rates are for APC1-3 and ADU1-6. APC1-3 had its small, circular gun turret rotated by 90°. ADU1-6 had its engine running and tracking radar lowered. Opening the hatches on the ADU further increases the error rate, as seen by ADU1-7. The highest error rates within this group are for MBT1-2, MBT1-6, and ADU1-4; namely the MBT with hatches and two out of the three toolboxes open, the MBT with camouflage netting draped over, and the ADU with its turret rotated through 90°. The camouflage netting was only designed to provide visual disguise and was draped over the vehicle such that much of the vehicle's shape remained evident in the radar imagery. Nevertheless it distorts enough of the vehicle's radar signature to cause confusion to the classifier.

For Groups 3, 4, and 5 (figures 6.6c through 6.6h), the dominant parameter affecting the rank ordering of the results is vehicle type, not configuration. This can be seen in the tendency of same-vehicles to yield similar error rates despite being associated with different configurations.

The effects of including shifting are mixed. Some data sets benefit to a large degree with a shift of ± 1 pixel, e.g. MBT1-3 in figure 6.6a and MBT1-2 in figure 6.6b. Some data sets suffer with a shift of ± 1, e.g. ADU1-3 in figure 6.6a. Many data sets perform worse when a shift of ± 2 pixels is used, e.g. MBT1-4 and ADU1-3 in figure 6.6a. Furthermore, this behaviour is not always consistent as the number of features changes. For MBT1-2 (figure 6.2a) the drop in error rate when going from no-shift to shift ± 1 was both large and consistent above 10 dimensions. For MBT1-6 (figure 6.2b) the error rate at 20 dimensions is slightly lower for a shift of ± 1 than for no-shift, but this changes with dimension.

6.3 Supervised loss function

6.3.1 Error rate with number of features

Plots of error rate against feature-space dimension are shown for the data sets in Group 1 in figure 6.7. Previous results showed the effect of performing a shift of ± 2 pixels is highly variable. Because of this and the computational expense of implementing that degree of shifting, from now on only a shift of ± 1 pixel is considered. Computational effort is further an issue here because of the extra parameter, ρ, that varies.

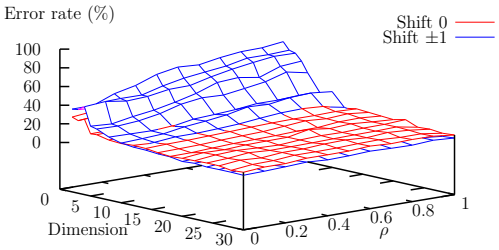
The error rate starts high at low dimension and decreases as the dimension increases. This is much more pronounced along $\rho = 1$ than along $\rho = 0$, with a more or less gradual transition between the two. APC1-2 in figure 6.7e is a particularly good example of this trend. For all values of ρ, the error rate levels off by 20 dimensions.

At low dimensions shift ± 1 generally gives the highest error rate. As dimension increases, the error rate for shift ± 1 usually drops below that for no-shift. The MBT error-rate surfaces best demonstrate this pattern. APC1-2 in figure 6.7e oscillates between shift 0 and shift ± 1 for the highest error rate at low dimensions, but consistently shows shift 0 giving the higher error rate at higher dimensions. For ADU1-3 in figure 6.7f shift ± 1 has the highest error rate for most combinations of ρ and dimension. In comparison, ADU1-5 in figure 6.7g has a greater tendency for no-shift to give the highest error rate at higher dimensions, although the results are mixed.

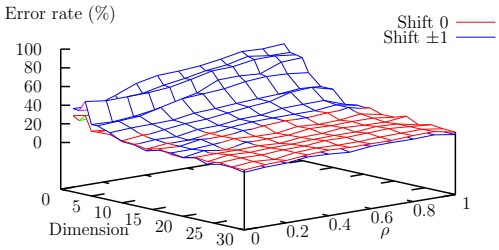
Plots for Group 2 are shown in figure 6.8. Again the pattern is for error rate to start high at low dimension and decrease as the dimension increases. This is more pronounced for $\rho = 1$ than for $\rho = 0$, with a gradual transition between the two. Again a data set of APC1, namely APC1-3 in figure 6.8c, provides a particularly clear example of this. For all values of ρ the error rate levels off by 20 dimensions.

With regards to the effect of shifting, the results for this group are more mixed than for Group 1. MBT1-2 in figure 6.8a shows a distinct trend for shift ± 1 to give the highest error rate at low dimensions but the lowest error rate at high dimensions. The next most similar trend is for APC1-3 in figure 6.8c. ADU1-4 in figure 6.8d is very variable, with the two surfaces for shifting often crossing each other. MBT1-6, ADU1-6, and ADU1-7, in figures 6.8b, 6.8e, and 6.8f respectively, all show a preponderance of shift ± 1 giving the highest error rate.

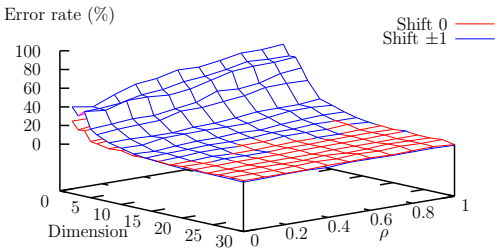
Plots for Group 3 are shown in figure 6.9. As observed previously, when using the unsupervised loss function, there is very little dependence of error rate on dimension. Only MBT2, in figures 6.9a and 6.9b, shows a weak dependence on dimension and this levels off by 20 dimensions. There is little effect of ρ on performance; even for MBT2 any effect is slight.



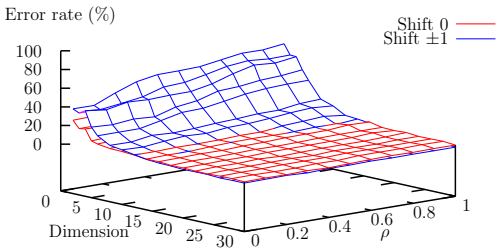
(a) *MBT1-3*



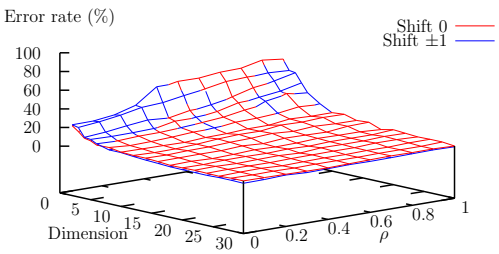
(b) *MBT1-4*



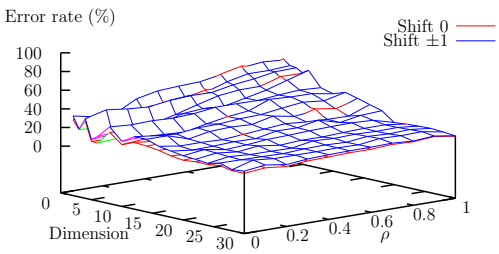
(c) *MBT1-5*



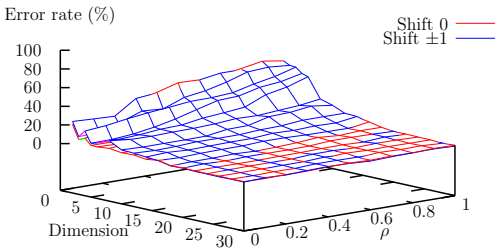
(d) *MBT1-7*



(e) *APC1-2*

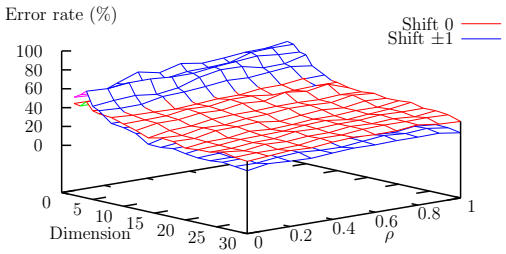


(f) *ADU1-3*

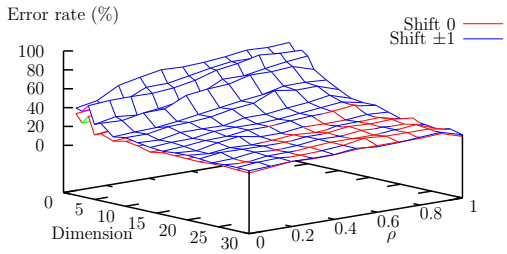


(g) *ADU1-5*

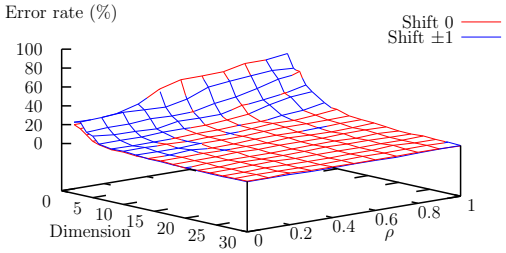
Figure 6.7: *Group 1 test results, with shifting, for RBF projection trained using supervised loss function.*



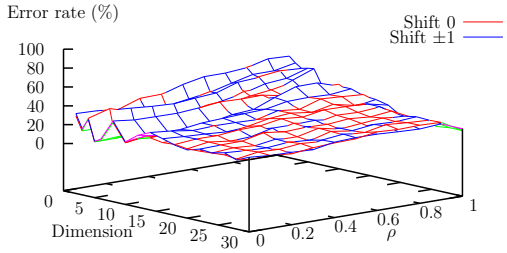
(a) *MBT1-2*



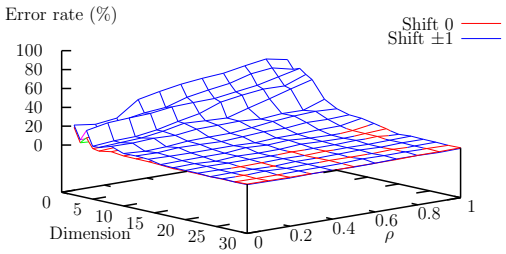
(b) *MBT1-6*



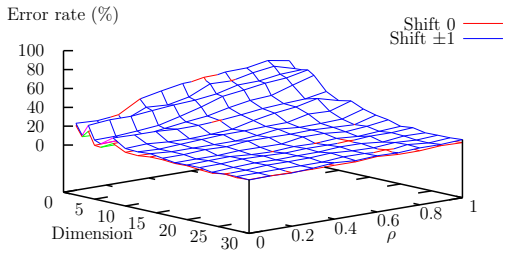
(c) *APC1-3*



(d) *ADU1-4*

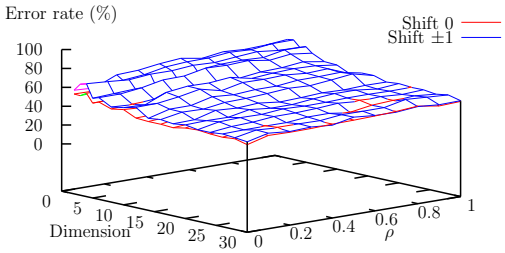


(e) *ADU1-6*

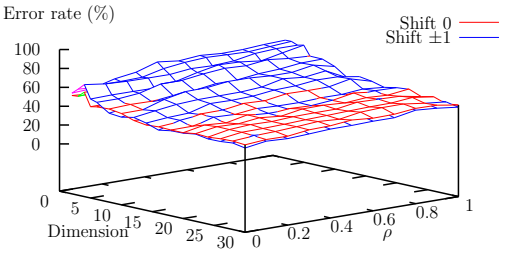


(f) *ADU1-7*

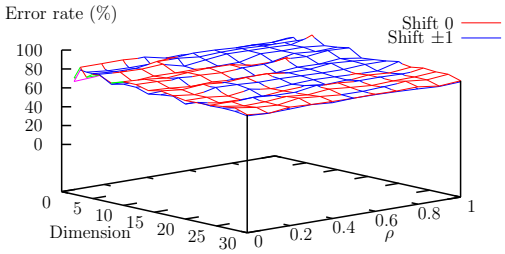
Figure 6.8: *Group 2 test results, with shifting, for RBF projection trained using supervised loss function.*



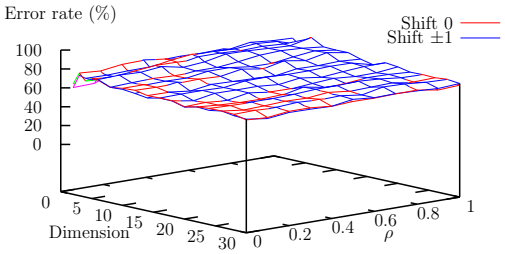
(a) *MBT2-2*



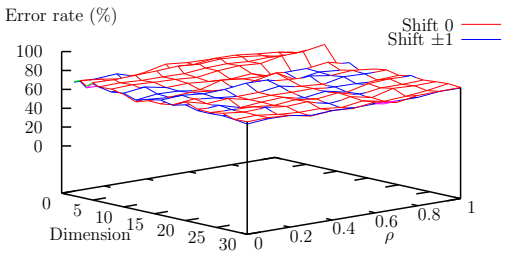
(b) *MBT2-3*



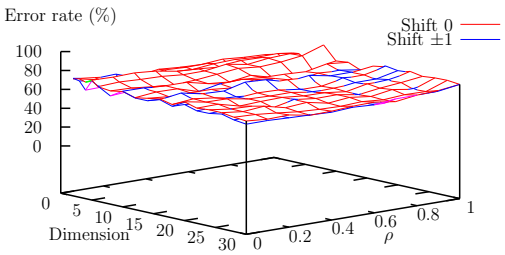
(c) *MBT3-3*



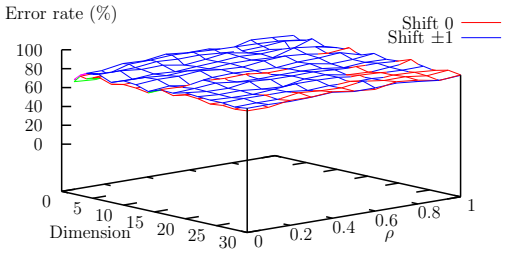
(d) *MBT3-6*



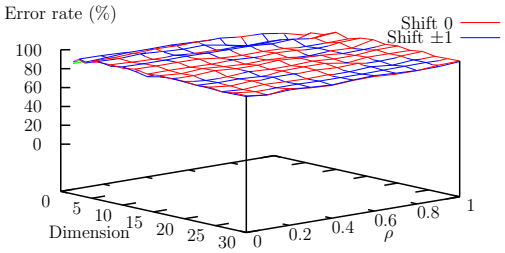
(e) *APC2-2*



(f) *APC2-3*

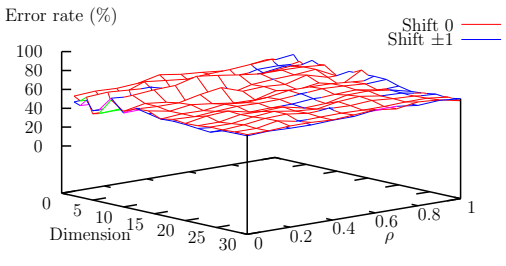


(g) *MBT4-3*

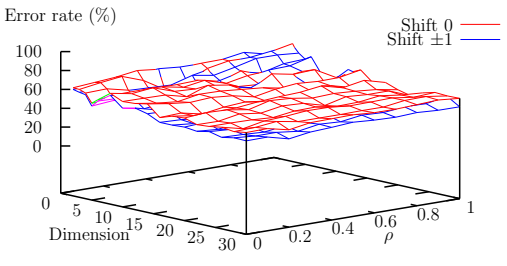


(h) *MBT5-3*

Figure 6.9: *Group 3 test results, with shifting, for RBF projection trained using supervised loss function.*



(i) *ADU2-1*



(j) *ADU3-1*

Figure 6.9: *Group 3 test results, with shifting, for RBF projection trained using supervised loss function (cont.)*

The effect of shifting on Group 3 is mixed. Concentrating just on the two data sets that achieve better than random error rate (MBT2-2 and MBT2-3 in figures 6.9a and 6.9b) the first has shift ±1 almost always giving the highest error rate whilst the second has an increased tendency for shift 0 to yield the highest error rate.

Plots for Group 4 are shown in figure 6.10. Only the results for MBT2, shown in figures 6.10a through 6.10c, show a decrease in error rate as dimension increases and this trend is a weak one. The use of a shift of ±1 pixel tends to yield a higher error rate than no-shift for MBT2, although the opposite is true for MBT2-4 (figure 6.10a) at high dimensions.

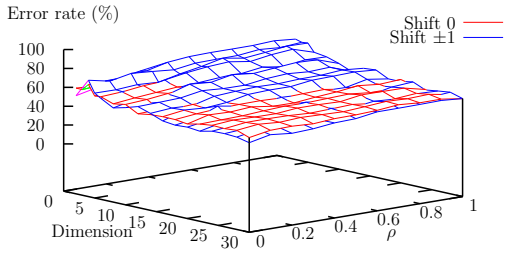
Plots of error rate as a function of ρ and feature-space dimension are shown for Group 5 in figure 6.11. No results are better than the level of random chance and neither is there any discernible pattern with either ρ, dimension, or shifting.

6.3.2 Summary of results

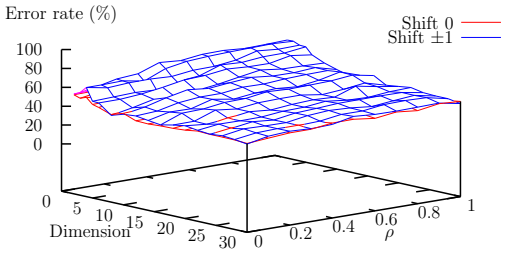
For comparison of results across data sets, the error rate is plotted for the 1-NNR using 20 features, for each of the data sets, in figure 6.12 (using $\rho = 0$).

The results for Group 1 are shown in figure 6.12a. The lowest error rates are obtained on MBT1-5, MBT1-7, APC1-2, and ADU1-5. Higher error rates were obtained for MBT1-3, MBT1-4, and ADU1-3. This split is not across vehicle type as MBT1 and ADU1 are present amongst both the lowest and highest error rates. The inclusion of shifting reduces some error rates, increases one, and has little effect on two others.

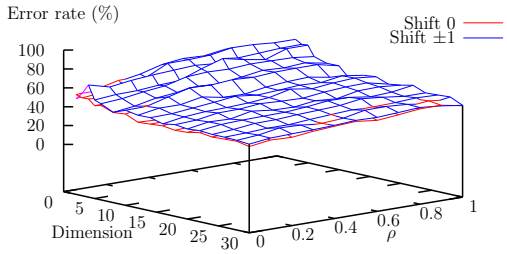
The results for Group 2 are shown in figure 6.12b. APC1-3 and ADU1-6 have particularly low error rates of less than 5%. ADU1-7 is a little higher. MBT1-2, MBT1-6, and ADU1-4 have the highest error rates. As with Group 1, this ordering does not reflect vehicle type: ADU1-6 is the lowest and ADU1-4 is one of the highest. The effect



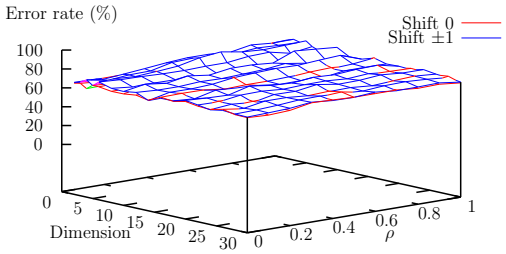
(a) *MBT2-4*



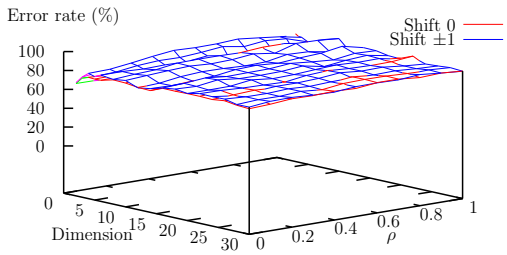
(b) *MBT2-5*



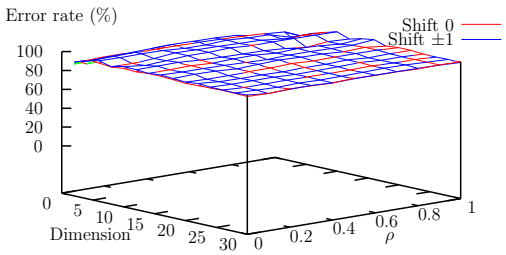
(c) *MBT2-6*



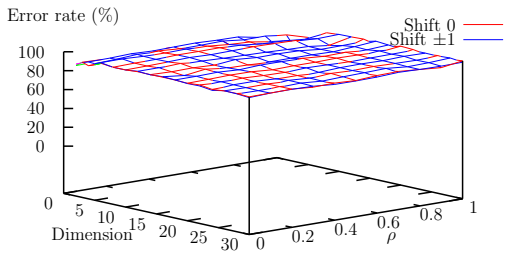
(d) *MBT4-1*



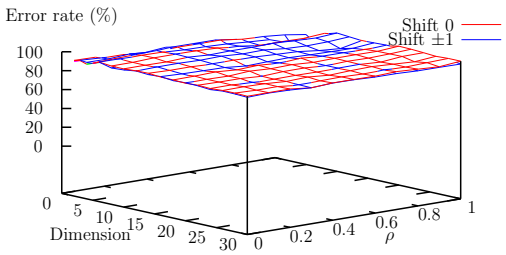
(e) *MBT4-2*



(f) *MBT5-4*

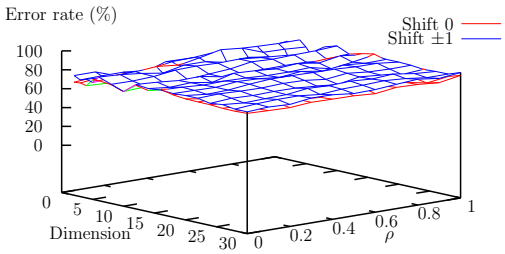


(g) *MBT5-5*

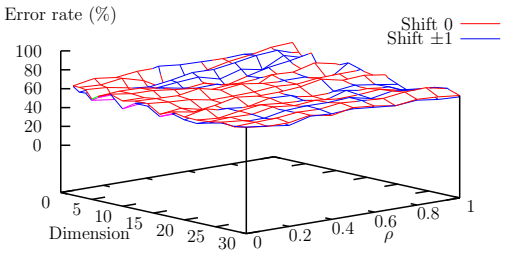


(h) *MBT5-6*

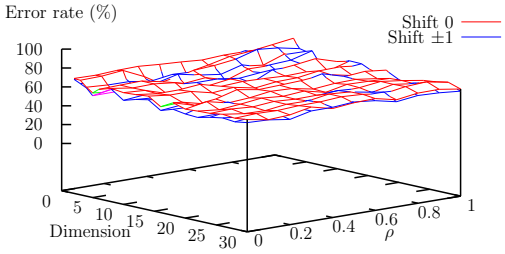
Figure 6.10: *Group 4 test results, with shifting, for RBF projection trained using supervised loss function.*



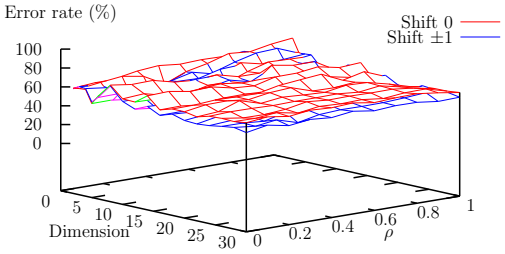
(i) *MBT4-4*



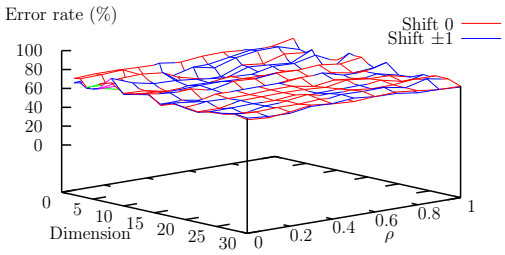
(j) *ADU3-2*



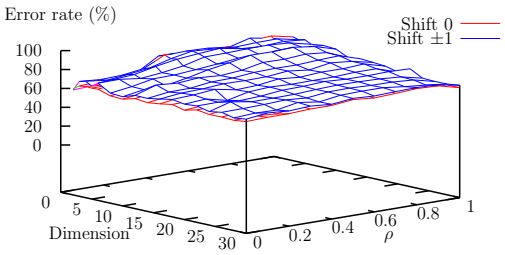
(k) *ADU3-3*



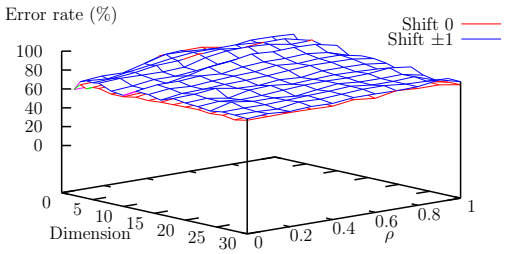
(l) *ADU3-4*



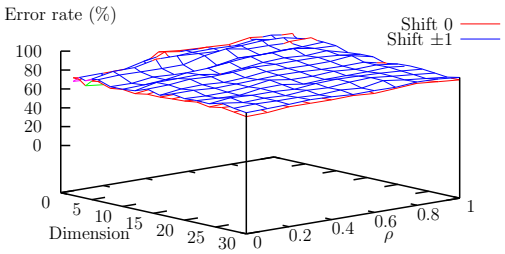
(m) *ADU3-5*



(n) *MBT3-7*

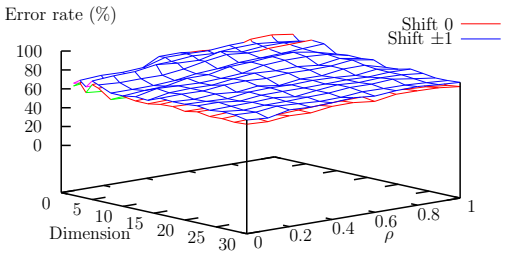


(o) *MBT3-8*



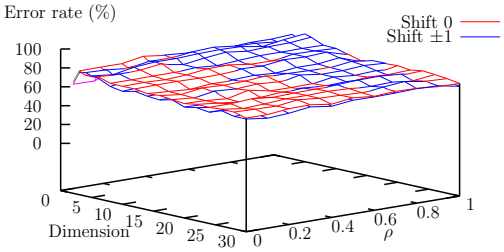
(p) *MBT3-9*

Figure 6.10: *Group 4 test results, with shifting, for RBF projection trained using supervised loss function (cont.)*

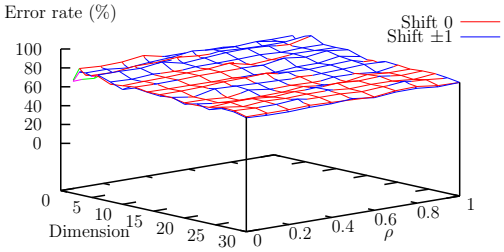


(q) *MBT3-10*

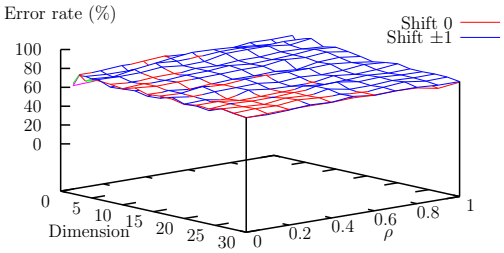
Figure 6.10: *Group 4 test results, with shifting, for RBF projection trained using supervised loss function (cont.)*



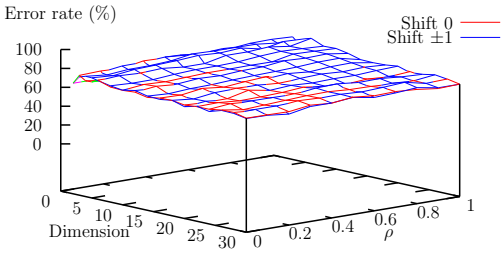
(a) *MBT3-1*



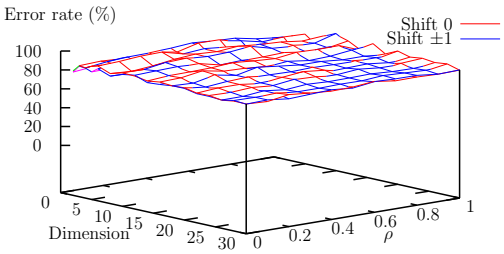
(b) *MBT3-2*



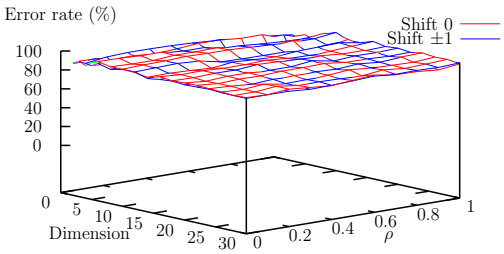
(c) *MBT3-4*



(d) *MBT3-5*

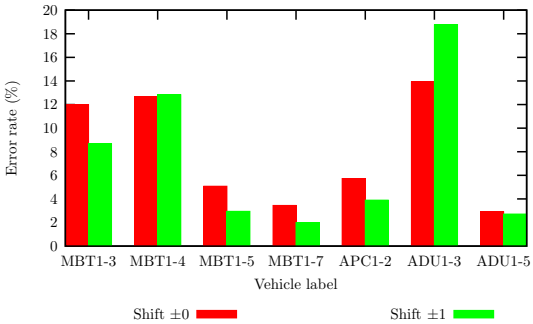


(e) *MBT5-1*

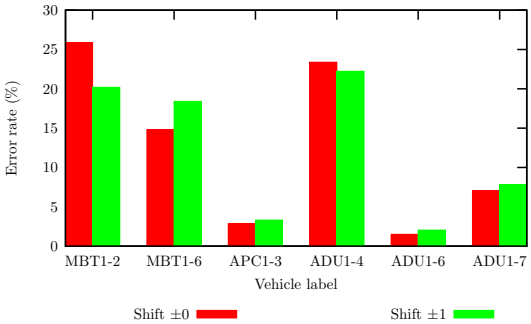


(f) *MBT5-2*

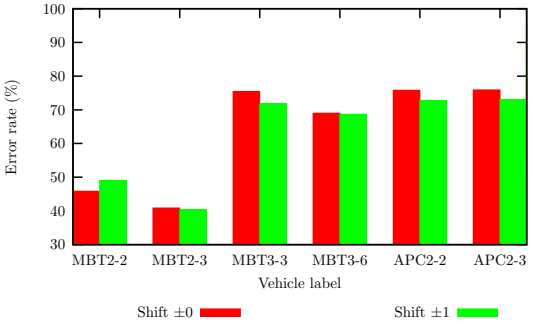
Figure 6.11: *Group 5 test results, with shifting, for RBF projection trained using supervised loss function.*



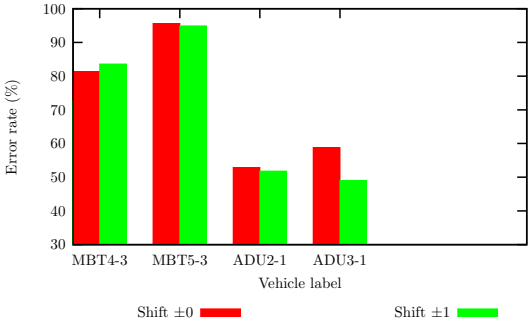
(a) *Group 1*



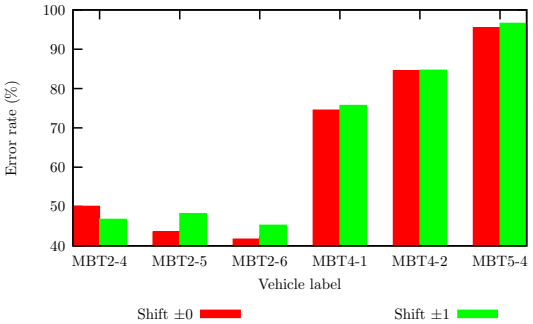
(b) *Group 2*



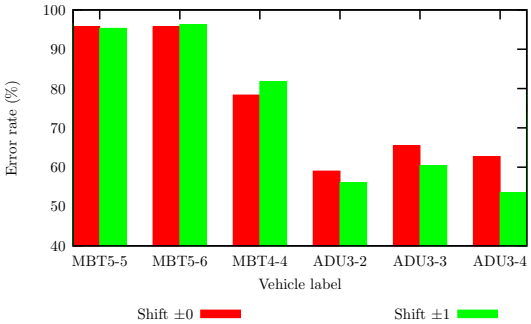
(c) *Group 3*



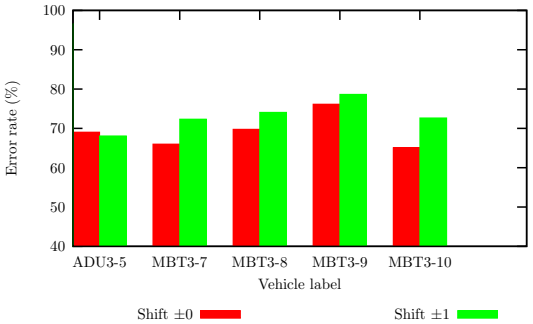
(d) *Group 3 (cont.)*



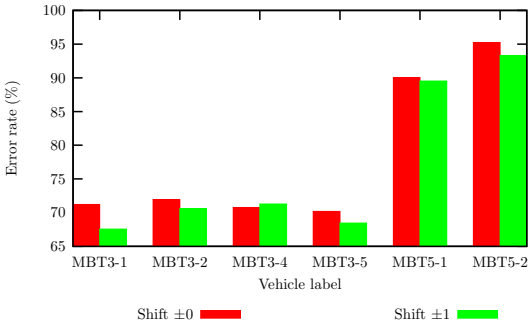
(e) *Group 4*



(f) *Group 4 (cont.)*



(g) *Group 4 (cont.)*



(h) *Group 5*

Figure 6.12: *Error rate from the 1-NNR using 20 features calculated by the RBFN trained using the supervised loss function with $\rho = 0$. The larger groups (3 and 4) are split across subfigures, but the y-range is kept constant.*

of including shifting is mixed, e.g. MBT1-2 has a decrease of approximately 5 percentage points with shifting, MBT1-6 has an increase of approximately 3 percentage points, and the others see very small changes.

The results for Group 3 are shown in figures 6.12c and 6.12d. Only MBT2-2, MBT2-3, ADU2-1, and ADU3-1 show error rates better than random. The best is MBT2-3, with an error rate approximately 30 percentage points better than random. Of these data sets, the inclusion of shifting increases the error rate for MBT2-2, decreases it for ADU3-1, and has little effect on the other two.

The results for Group 4 are shown in figures 6.12e through 6.12g. The lowest error rates are again associated with MBT2, namely MBT2-4, MBT2-5, and MBT2-6, with error rates at least 20 percentage points better than random chance. At approximately the same level of random chance or better are the data sets for ADU3, namely ADU3-2, ADU3-3, and ADU3-4. Again, the addition of shifting provides mixed results: the error rate falls for MBT2-4, but rises for MBT2-5 and MBT2-6, and falls for ADU3-2, ADU3-3, and ADU3-4.

The results for Group 5 are shown in figure 6.12h. The data sets for MBT3 have error rates at approximately the level of random chance. Those for MBT5 are considerably higher. The inclusion of shifting at best reduces the error rate for MBT3-1 by approximately three percentage points, but this still gives an error rate close to random and this reduction in error rate is not consistent across the other MBT3 data sets.

6.3.3 Discussion

The lowest error rates are obtained in Groups 1 and 2, consistent with the fact that these two groups comprised only vehicles present during training and the other groups comprised only vehicles not present during training.

The error rates for Groups 1 and 2 start high and fall rapidly as dimension increases. As with the unsupervised-loss case, this is interpretable as successive dimensions increasingly representing structure in the data that is useful for discrimination. By 20 dimensions the error rate levels off and this is interpreted to be because the extra features are not useful for discrimination, e.g. they just describe noise in the data. At low dimensions the lowest error rate is for $\rho = 0$, but this is no better than that obtainable using $\rho > 0$ with more features.

Once the error rate levels off with increasing dimension, it is independent of ρ. This is consistent with $\rho = 0$ selecting for structure that describes inter-class relationships. Such relationships are important for classification and so using $\rho = 0$ is beneficial at low dimensions because it preferentially selects features useful for discrimination. As the number of features increases sufficiently to capture all of the data structure, such

discriminatory information will be present regardless of the value of ρ.

The only data sets in Groups 3 and 4 to show a discernible dependence on ρ and dimension are those for MBT2, although the dependence is much weaker than for the data sets in Groups 1 and 2. It is also MBT2 that yields the lowest error rates (better than random) out of Groups 3 and 4. These two facts together suggest that useful, discriminatory features are being extracted for this vehicle.

The spread of results in Groups 1 and 2 is not attributable to vehicle type. The spread is the same as seen for the unsupervised loss and is explicable in the same way: small hatches on the MBT have the least effect on the radar signature, toolboxes have a greater effect, and the orientation of the large ADU gun turret has the largest effect. For Groups 3–5 the spread of results is associated with vehicle type. The same vehicles give similar error rates irrespective of configuration.

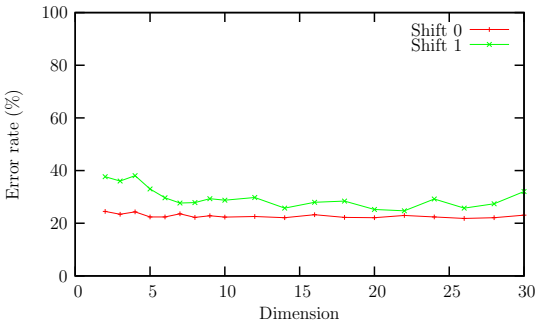
The effect of including shifting is mixed, even within a single group. For example, comparing across the data sets in Group 1, shown in figure 6.12a, some data sets benefit, e.g. MBT1-3, some are worse off, e.g. ADU1-3, and some show little change, e.g. MBT1-4. Furthermore, some data sets demonstrate consistency across ρ and dimension, e.g. APC1-2 in figure 6.7e, whilst others do not, e.g. MBT1-6 in figure 6.8b.

6.4 Rescaled inter-class distances

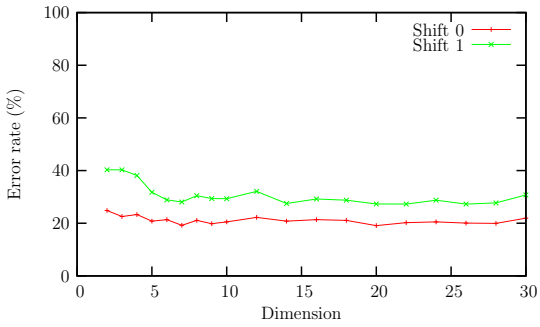
6.4.1 Error rate with number of features

Plots of error rate against feature-space dimension are shown for the data sets in Group 1 in figure 6.13. There is very little dependence of error rate on dimension. Any decrease that there is in the error rate with increasing dimension is most evident for the MBT1 data sets, namely MBT1-3, MBT1-4, MBT1-5, and MBT1-7 in figures 6.13a through 6.13d. Even then the trend is a weak one that extends only up to approximately 6 dimensions. Consistent across all data sets is that the error rate for shift 0 is lower than that for shift ± 1 over all dimensions, although the magnitude of the difference varies between data sets.

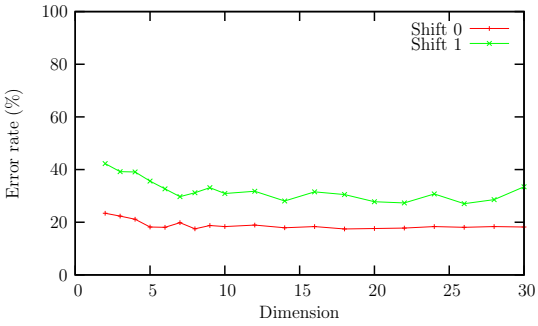
Plots of error rate against feature-space dimension are shown for the data sets in Group 2 in figure 6.14. There is very little dependence of error rate on dimension in this group. With the exception of MBT1-2 in figure 6.14a, all data sets have a lower error rate for shift 0 compared to shift ± 1. In the case of MBT1-2 there are some dimensions where shift ± 1 gives the lower error rate, although there is little difference between the two. The magnitude of the difference between the no-shift and shift cases varies between data sets.



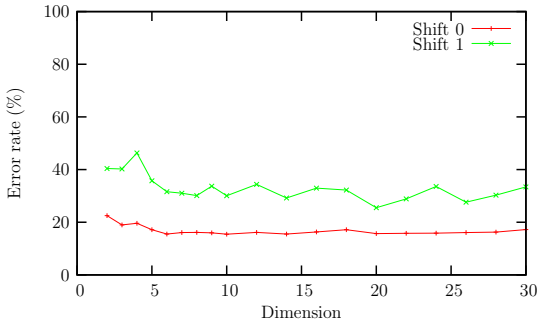
(a) *MBT1-3*



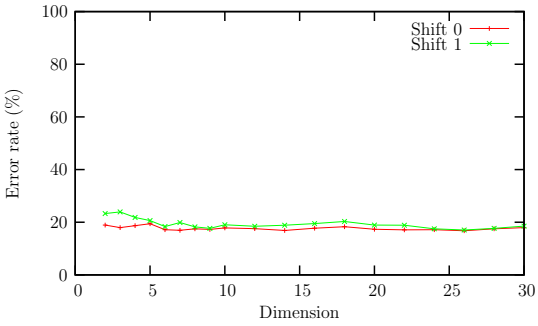
(b) *MBT1-4*



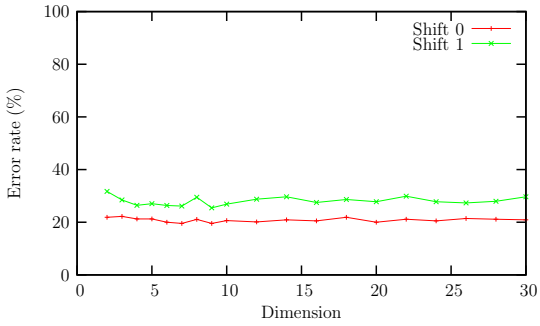
(c) *MBT1-5*



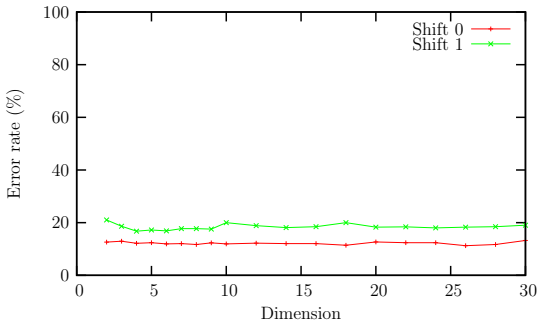
(d) *MBT1-7*



(e) *APC1-2*

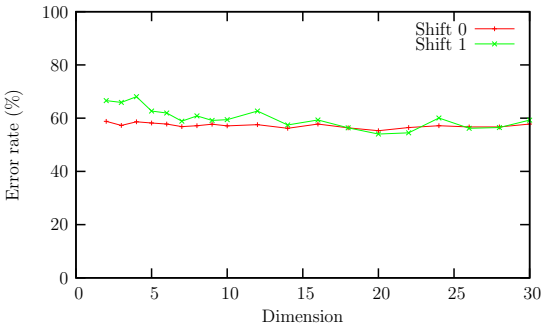


(f) *ADU1-3*

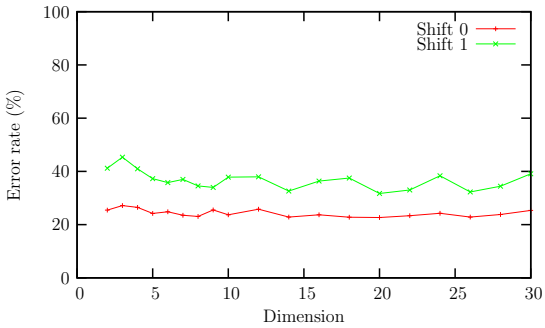


(g) *ADU1-5*

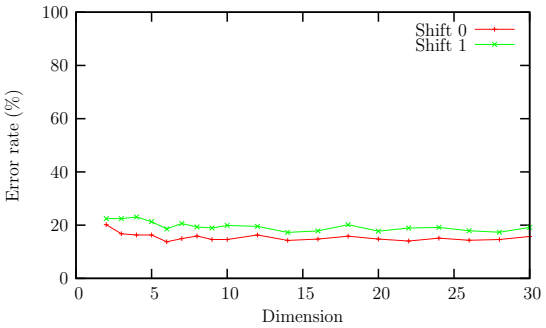
Figure 6.13: *Group 1 test results, with shifting, for RBF projection trained using rescaled inter-class distances.*



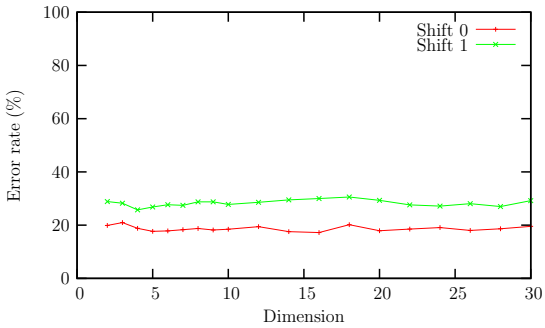
(a) *MBT1-2*



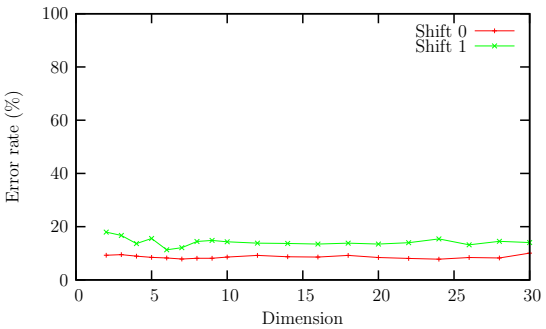
(b) *MBT1-6*



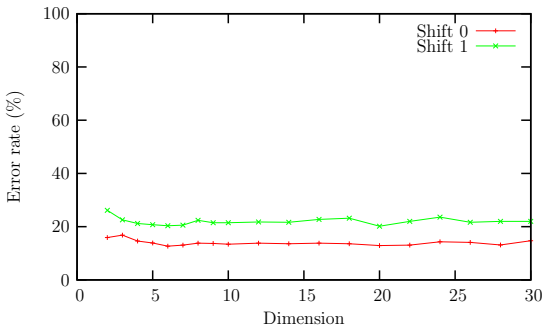
(c) *APC1-3*



(d) *ADU1-4*



(e) *ADU1-6*



(f) *ADU1-7*

Figure 6.14: *Group 2 test results, with shifting, for RBF projection trained using rescaled inter-class distances.*

Plots of error rate against feature-space dimension are shown for the data sets in Group 3 in figure 6.15. The only data sets for which the error rates are better than random are: MBT2-2 in figure 6.15a, MBT2-3 in figure 6.15b, and ADU2-1 in figure 6.15i. The inclusion of shifting has the tendency to increase the error rate for these data sets. This is generally consistent as dimension increases, but the difference is small, particularly for MBT2-3. The error rates for the remaining data sets are seen to be worse than random.

Plots of error rate against feature-space dimension are shown for the data sets in Group 4 in figure 6.16. All results appear to be largely independent of dimension. The MBT2 data sets, namely MBT2-4 in figure 6.16a, MBT2-5 in figure 6.16b, and MBT2-6 in figure 6.16c, achieve better than random-chance error rate. The inclusion of shifting makes little difference for these data sets. MBT4-1 in figure 6.16d, MBT4-4 in figure 6.16i, MBT3-7 in figure 6.16n, and MBT3-8 in figure 6.16o also achieve error rates better than expected for random chance alone. MBT3-10 in figure 6.16q gives error rates better than random when shifting of the input images is not performed. Including shifting raises the error rate to the level of random chance. The error rates for the remaining data sets are no better than random. Overall, there is little difference between the error rates with and without shifting of the images. The inclusion of shifting tends either to increase the error rate, such as for MBT4-4 in figure 6.16i and MBT3-7 in figure 6.16n, or else it makes no overall difference and the error rates for no-shift and shift ± 1 overlay one another, such as for MBT2-5 in figure 6.16b and MBT4-1 in figure 6.16d.

Plots of error rate against feature-space dimension are shown for the data sets in Group 5 in figure 6.17. All results are worse than random chance and there is no discernible trend of error rate with dimension.

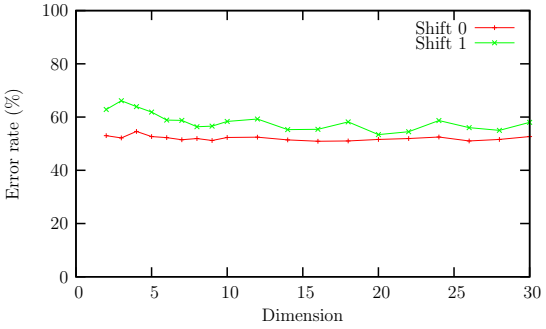
6.4.2 Summary of results

For comparison of results across data sets, the error rate is plotted for the 1-NNR using 20 features, for each of the data sets, in figure 6.18.

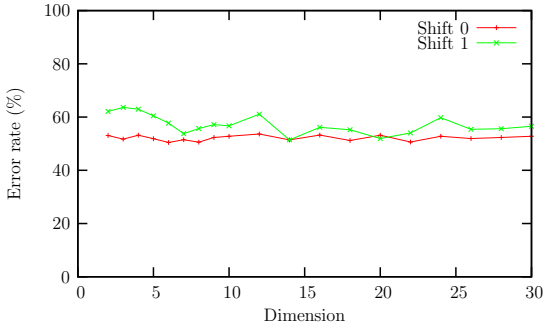
The summary for Group 1 is given in figure 6.18a. There is no easy partitioning of the results into 'low' and 'high' error rates. The inclusion of shifting increases all error rates.

The summary for Group 2 is given in figure 6.18b. There is a partitioning of the results within this group: the error rate for MBT1-2 is higher than all of the others in the group. The inclusion of shifting has little effect on the error rate for MBT1-2, but clearly increases the error rate for all other data sets.

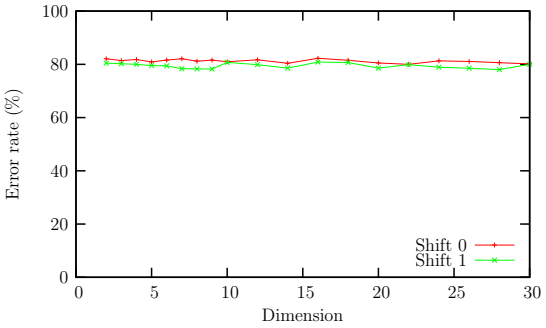
The summary for Group 3 is given in figures 6.18c and 6.18d. Data sets MBT2-2, MBT2-3, and ADU2-1 have error rates lower than that for random chance. For these data sets, the inclusion of shifting results in little change in the error rate for MBT2 and a noticeable increase for ADU2-1.



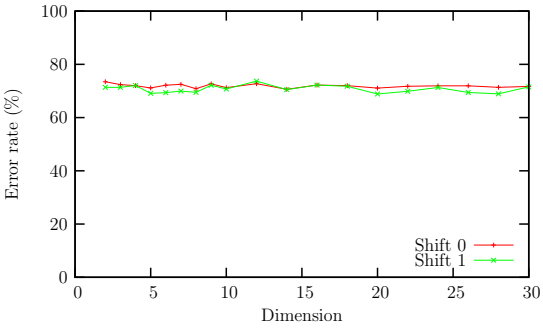
(a) *MBT2-2*



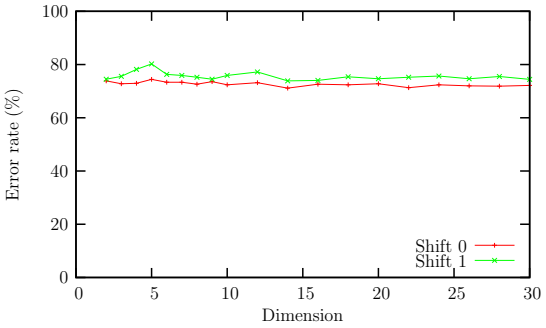
(b) *MBT2-3*



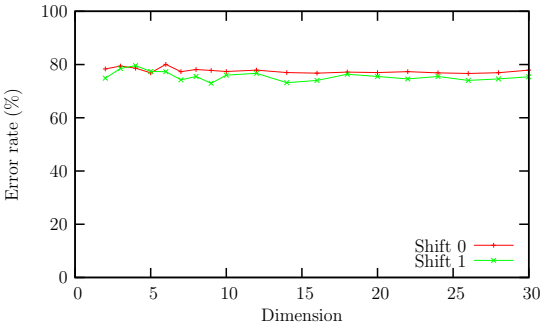
(c) *MBT3-3*



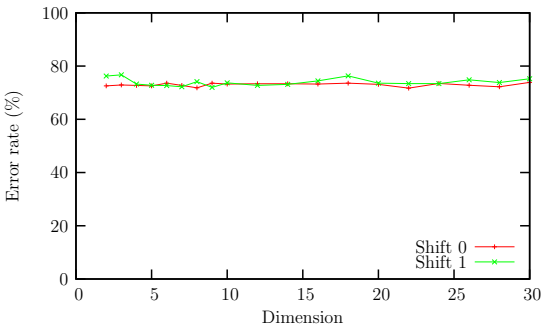
(d) *MBT3-6*



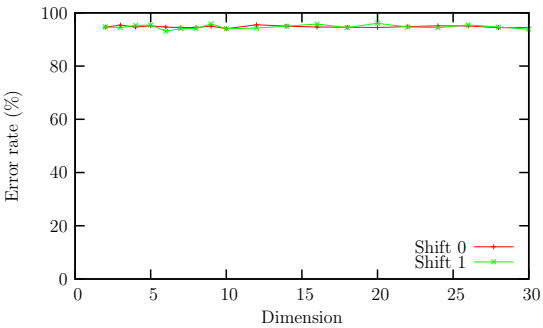
(e) *APC2-2*



(f) *APC2-3*

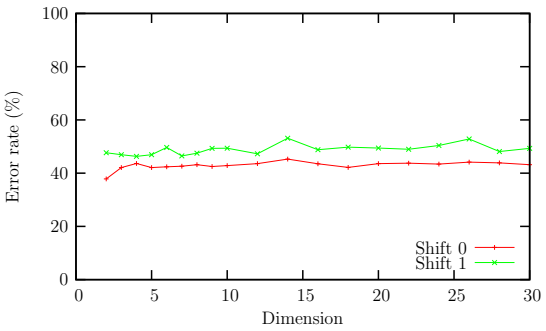


(g) *MBT4-3*

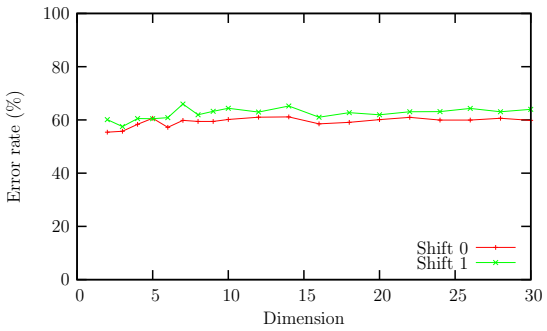


(h) *MBT5-3*

Figure 6.15: *Group 3 test results, with shifting, for RBF projection trained using rescaled inter-class distances.*



(i) *ADU2-1*



(j) *ADU3-1*

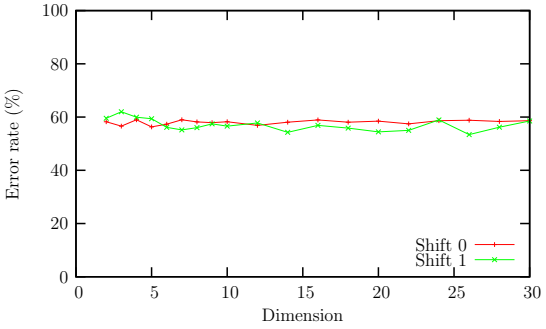
Figure 6.15: *Group 3 test results, with shifting, for RBF projection trained using rescaled inter-class distances (cont.)*

The summary for Group 4 is given in figures 6.18e through 6.18g. All of the MBT2 data sets, namely MBT2-4, MBT2-5, and MBT2-6, achieve better-than-random error rate, as does MBT4-1 and MBT4-4 but not MBT4-2. Although the error rates for ADU3-2, ADU3-3, and ADU3-4 are similar to that for MBT4-4, this is approximately at the level of random chance for the ADU class, which has a slightly lower prior probability than the MBT. Without shifting, ADU3-4 is marginally better than random. Also performing better than random are MBT3-7, MBT3-8, and MBT3-10 in figure 6.18g. The remaining MBT3 data set, MBT3-9, is at approximately the level of random chance. Amongst the meaningful, better than random chance, results the effect of shifting is mixed: some error rates decrease whilst some increase.

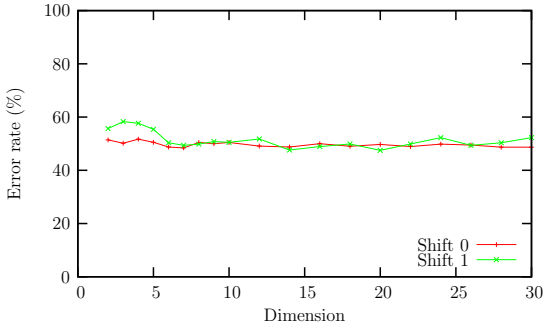
The summary for Group 5 is given in figure 6.18h. There is a grouping of results within this group. The MBT3 data sets have error rates approximately at the level of random chance, and the MBT5 data sets have much larger error rates.

6.4.3 Discussion

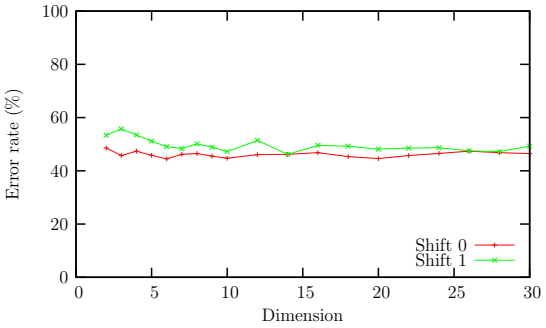
The pilot studies that were conducted using the rescaled inter-class distances show an encouraging separation of the training data when projected down to two dimensions, as shown in figure 5.24. This suggests that the first features calculated are efficient at extracting structure that separates the classes in the training data. This concept of concentrating the discriminatory data structure into a small number of features is consistent with the curves of error rate against dimension obtained for the test data, particularly Groups 1 and 2 in figures 6.13 and 6.14. Any decreasing trend in error rate as dimension increases is a relatively weak one and limited to a small number of features. Therefore, the first features to be calculated are rapidly extracting as much discriminatory information as the nonlinear transformation is capable of and subsequent



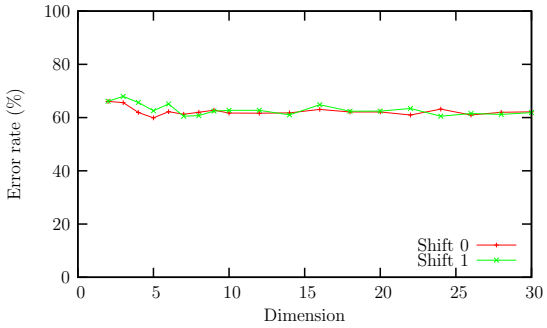
(a) *MBT2-4*



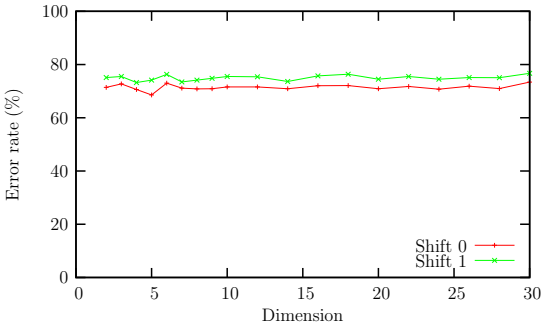
(b) *MBT2-5*



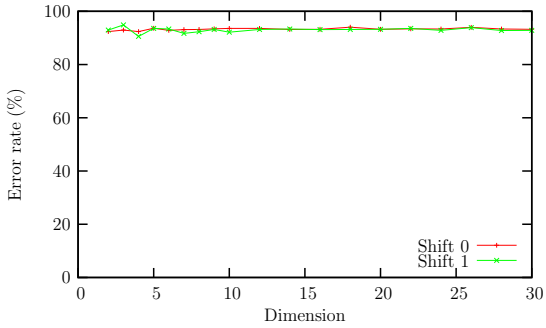
(c) *MBT2-6*



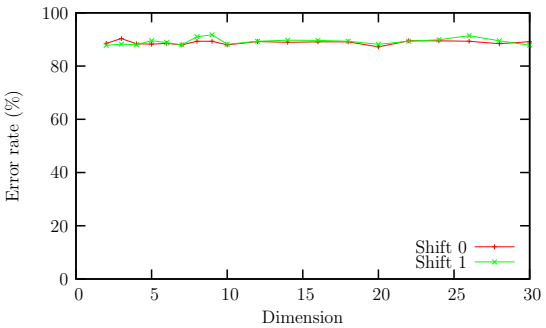
(d) *MBT4-1*



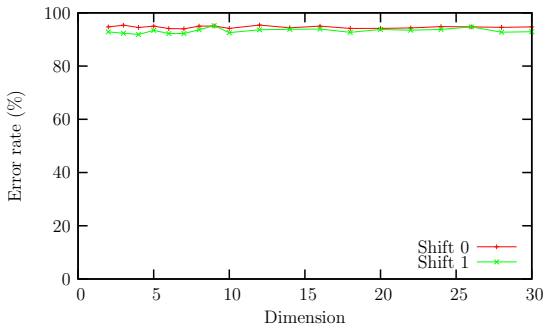
(e) *MBT4-2*



(f) *MBT5-4*

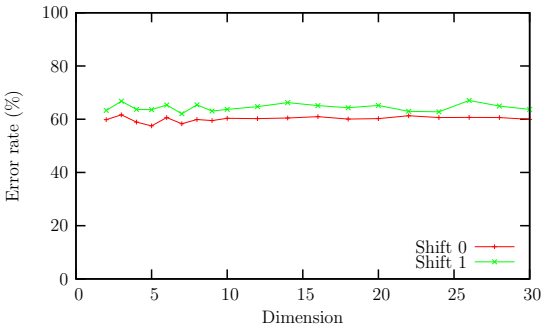


(g) *MBT5-5*

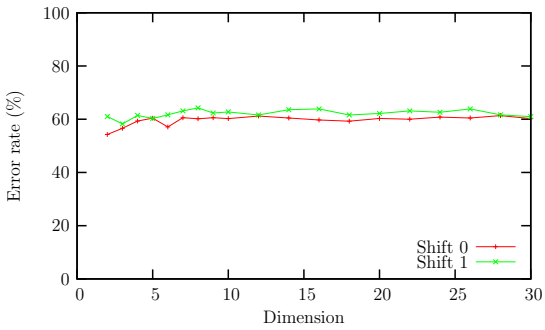


(h) *MBT5-6*

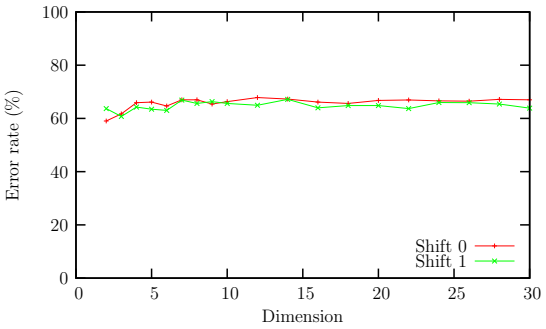
Figure 6.16: *Group 4 test results, with shifting, for RBF projection trained using rescaled inter-class distances.*



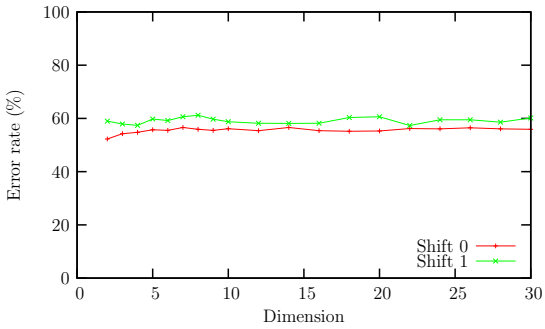
(i) *MBT4-4*



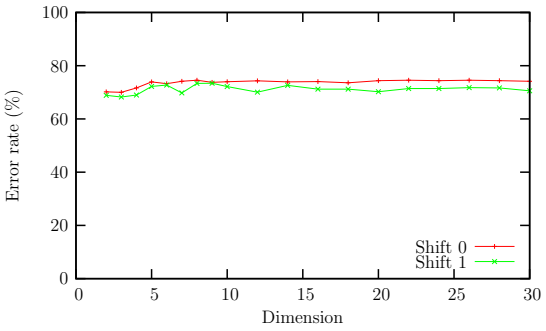
(j) *ADU3-2*



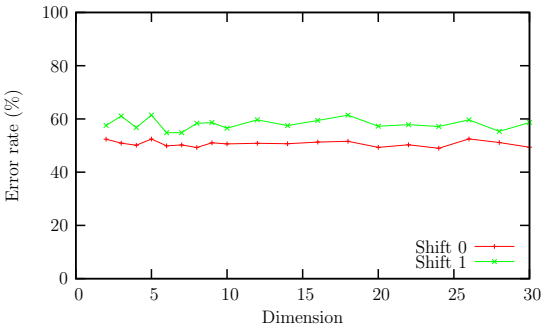
(k) *ADU3-3*



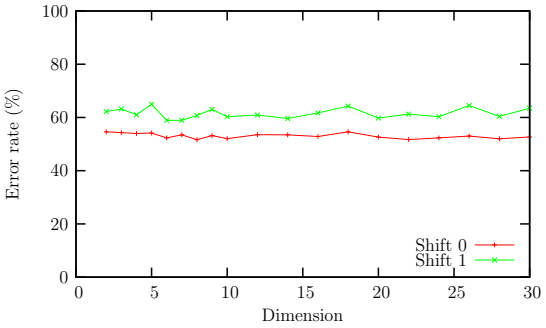
(l) *ADU3-4*



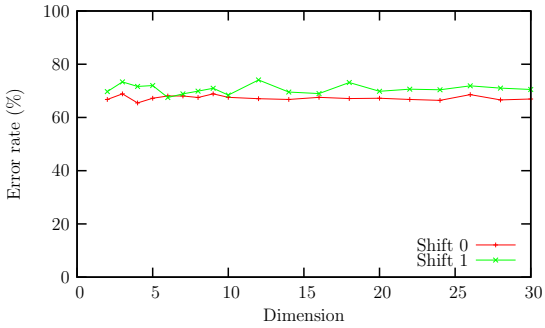
(m) *ADU3-5*



(n) *MBT3-7*

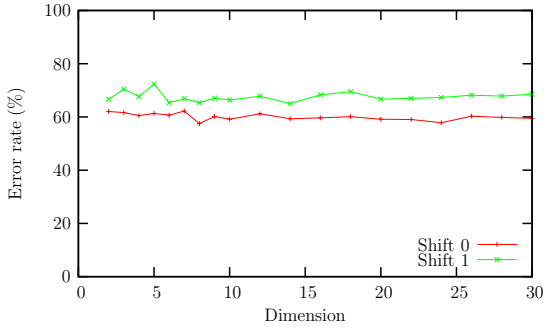


(o) *MBT3-8*



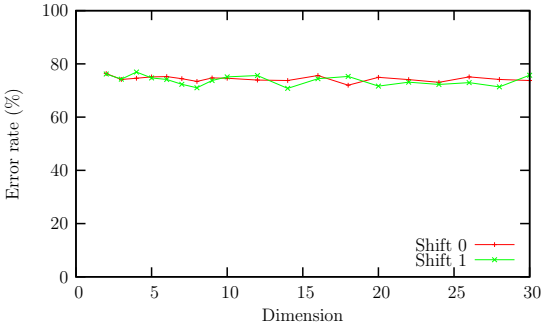
(p) *MBT3-9*

Figure 6.16: *Group 4 test results, with shifting, for RBF projection trained using rescaled inter-class distances (cont.)*

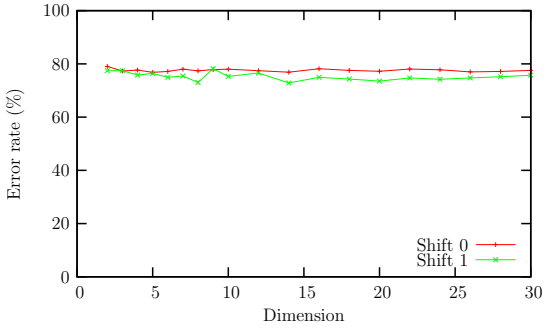


(q) *MBT3-10*

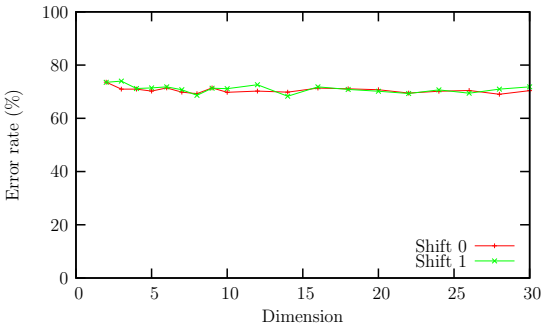
Figure 6.16: *Group 4 test results, with shifting, for RBF projection trained using rescaled inter-class distances (cont.)*



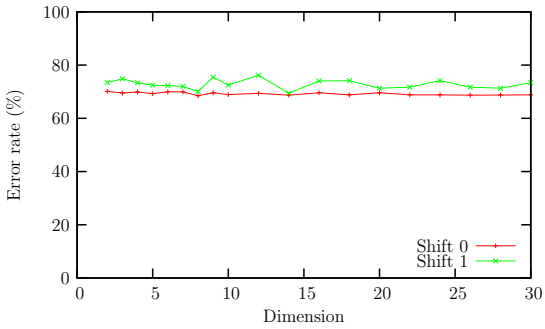
(a) *MBT3-1*



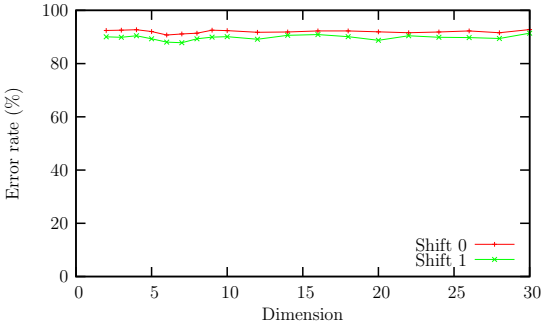
(b) *MBT3-2*



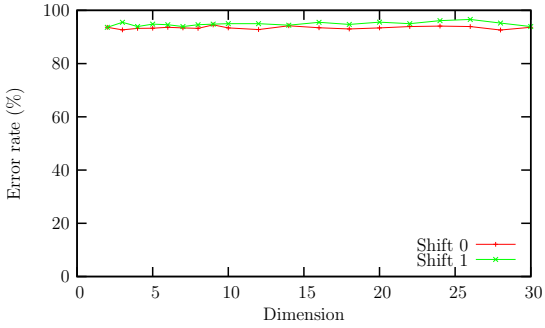
(c) *MBT3-4*



(d) *MBT3-5*

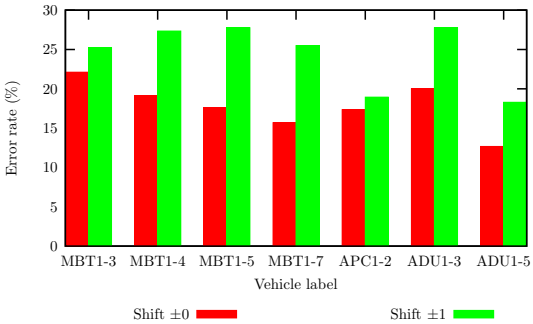


(e) *MBT5-1*

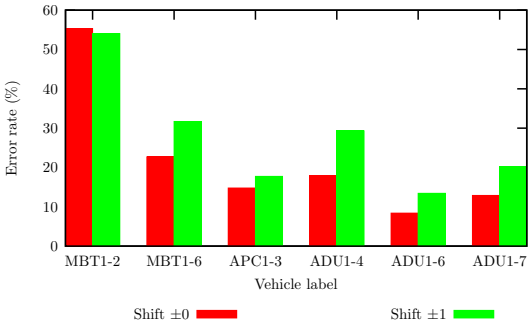


(f) *MBT5-2*

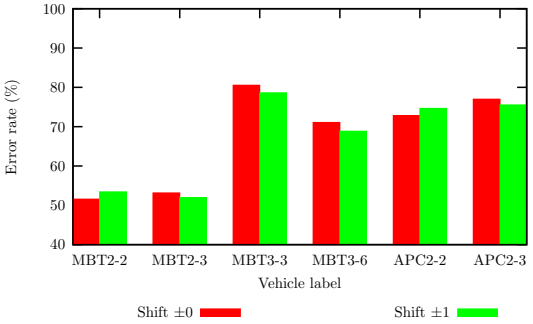
Figure 6.17: *Group 5 test results, with shifting, for RBF projection trained using rescaled inter-class distances.*



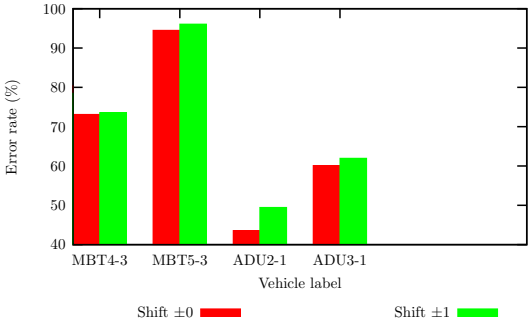
(a) *Group 1*



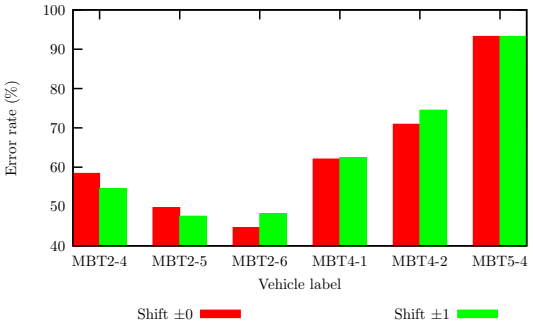
(b) *Group 2*



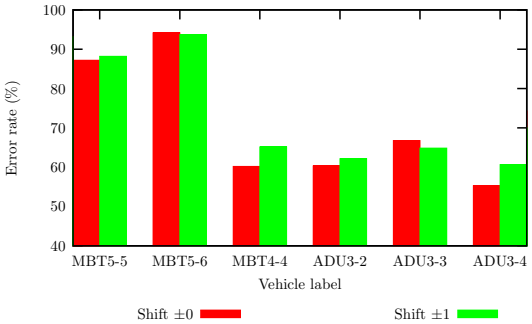
(c) *Group 3*



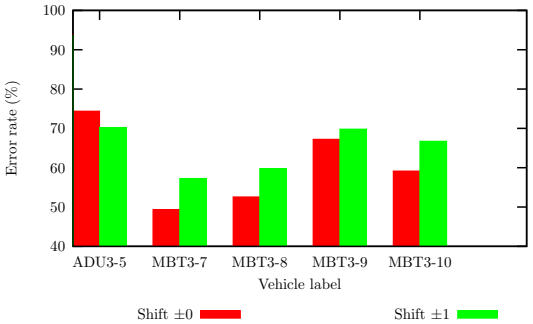
(d) *Group 3 (cont.)*



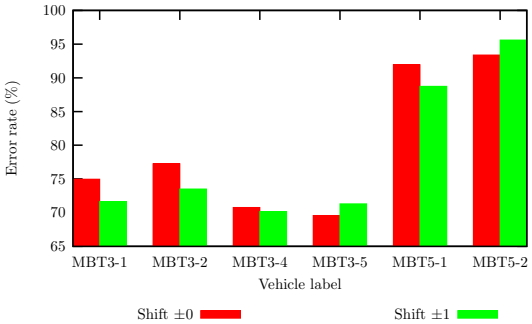
(e) *Group 4*



(f) *Group 4 (cont.)*



(g) *Group 4 (cont.)*



(h) *Group 5*

Figure 6.18: *Error rate from the 1-NNR using 20 features calculated by the RBFN trained using the rescaled inter-class distances. The larger groups (3 and 4) are split across subfigures, but the y-range is kept constant.*

features contribute very little discriminative power.

Within Groups 1 and 2 one thing that stands out in figure 6.18is the high error rate for MBT1-2 relative to the other data sets. There are data sets in Groups 3 and 4 for which the error rates were lower.

Relative to the previous two RBFN-based experiments, this approach of rescaling the inter-class distances has increased the consistency of the results for Groups 1 and 2, with the exception of MBT1-2, but at the expense of raising many error rates rather than lowering them. Provided the error rate remained acceptably low, however, this greater robustness to target configuration could be a desirable property in that it provides a greater confidence that classifier performance will be consistent across different target configurations.

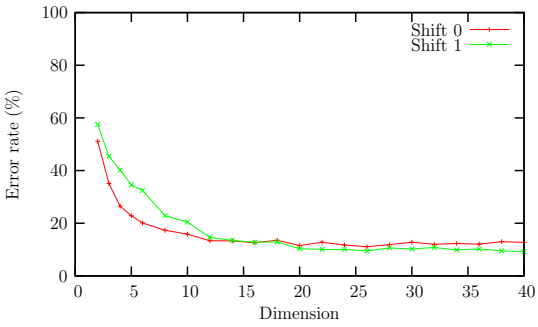
6.5 Box-Cox metric distances

6.5.1 Error rate with number of features

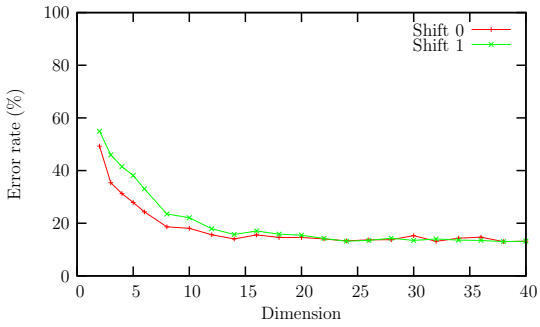
Plots of error rate as a function of the number of features extracted (feature-space dimension) for the data sets in Group 1 are given in figure 6.19. Generally there is an initial rapid decrease in error rate as the dimension increases. For all data sets there is little change in error rate beyond 15–20 dimensions. The error rates with shifting are very similar to those without shifting for all data sets. The greatest distinction between the two is for MBT1, shown in figures 6.19a through 6.19d, where the error rate at low dimension without shifting is consistently lower than that obtained with shifting. This distinction is lost between 15 and 20 dimensions.

Plots of error rate against feature-space dimension are shown for the data sets in Group 2 in figure 6.20. There is an initial large fall in error rate with increasing dimension. This levels off by 20 dimensions for all data sets, although some level off more rapidly, e.g. by 10 dimensions for APC1-3 in figure 6.20c. The effect of including shifting is mixed. Relative to results obtained without using shifting, the use of shifting for MBT1-2 in figure 6.20a gives lower error rates for dimensions fewer than 10 but for MBT1-6 in figure 6.20b it consistently yields higher error rates for all dimensions. The error rates for the other data sets have the two curves very close together, as was seen for the data sets of Group 1.

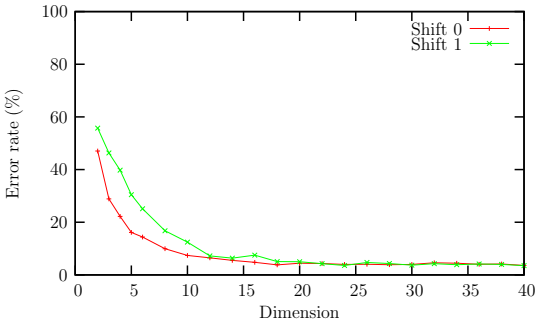
Plots of error rate against feature-space dimension are shown for the data sets in Group 3 in figure 6.21. All error rates are high in this group. Two data sets with error rate better than random are MBT2-2 in figure 6.21a and MBT2-3 in figure 6.21b. The MBT2-3 results show a weak trend for error rate to decrease with increasing dimension. If there is a similar trend for MBT2-2 it is even weaker. There is no discernible trend in



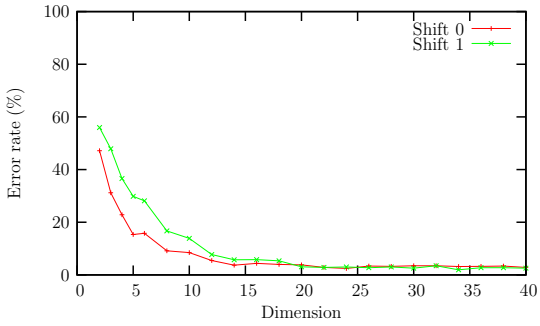
(a) *MBT1-3*



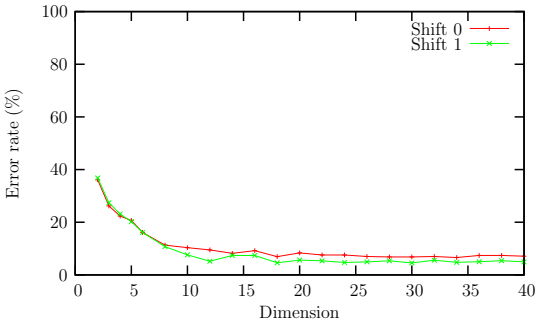
(b) *MBT1-4*



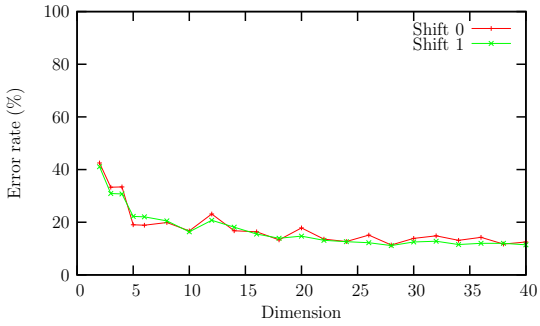
(c) *MBT1-5*



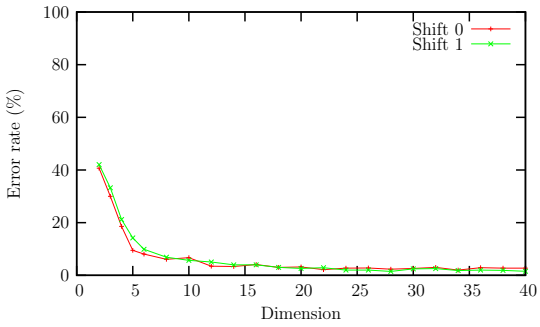
(d) *MBT1-7*



(e) *APC1-2*

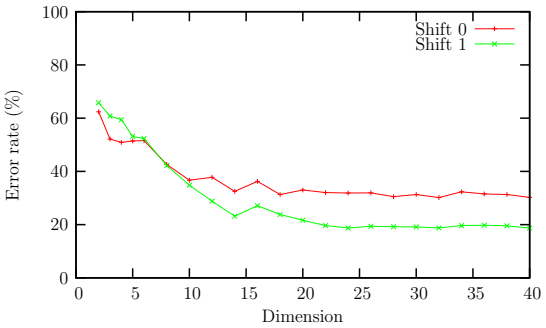


(f) *ADU1-3*

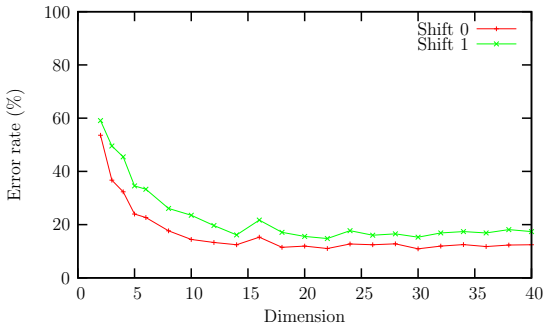


(g) *ADU1-5*

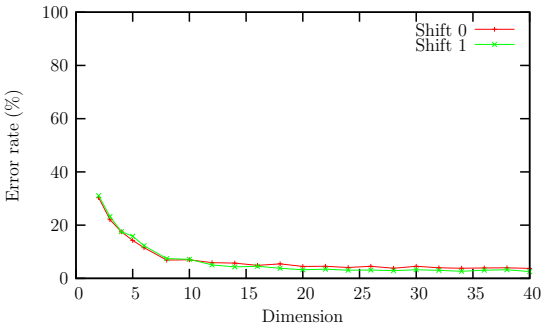
Figure 6.19: *Group 1 test results, with shifting, for RBF projection trained using Box-Cox metric distances.*



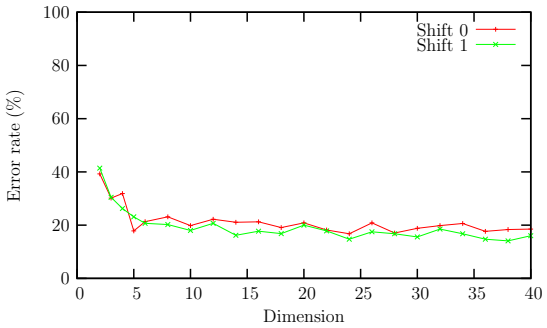
(a) *MBT1-2*



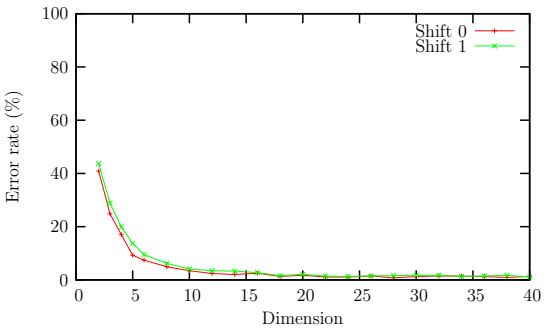
(b) *MBT1-6*



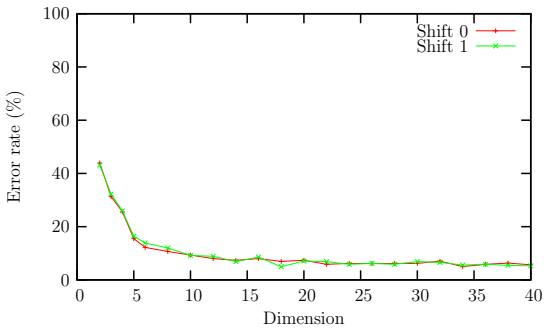
(c) *APC1-3*



(d) *ADU1-4*

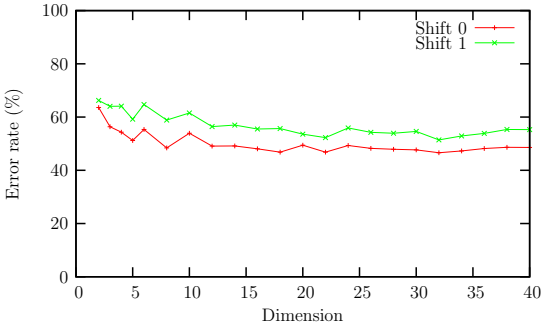


(e) *ADU1-6*

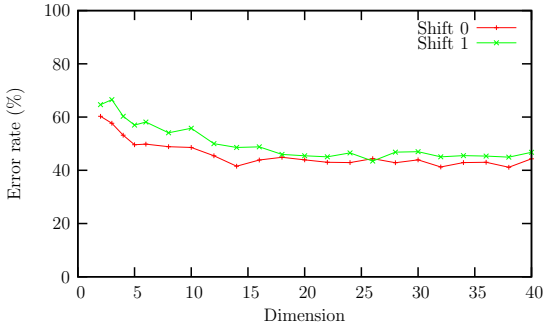


(f) *ADU1-7*

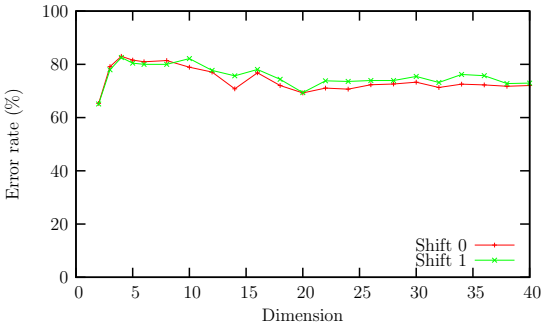
Figure 6.20: *Group 2 test results, with shifting, for RBF projection trained using Box-Cox metric distances.*



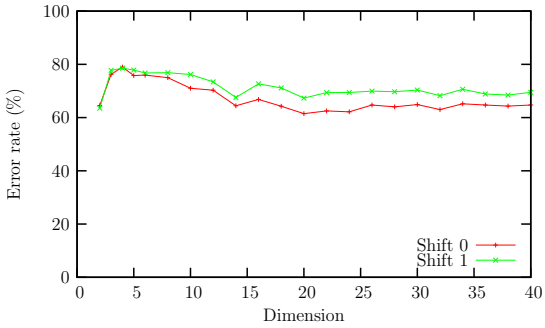
(a) *MBT2-2*



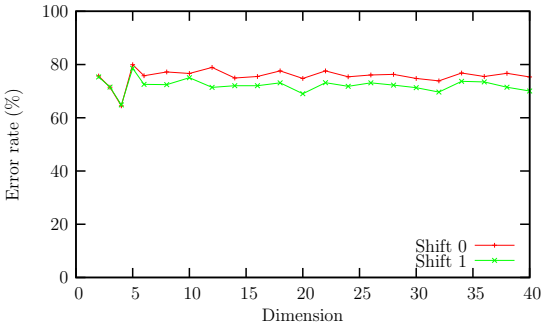
(b) *MBT2-3*



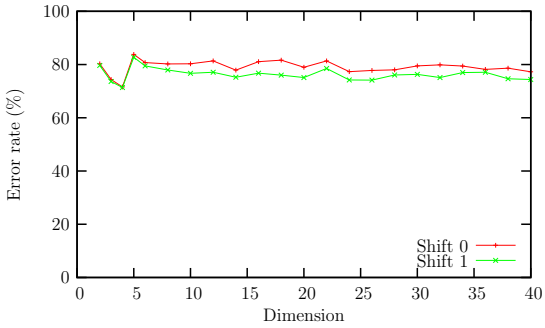
(c) *MBT3-3*



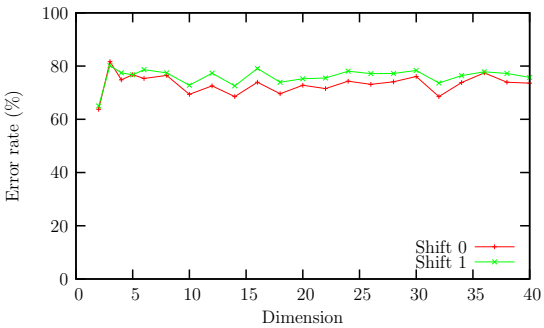
(d) *MBT3-6*



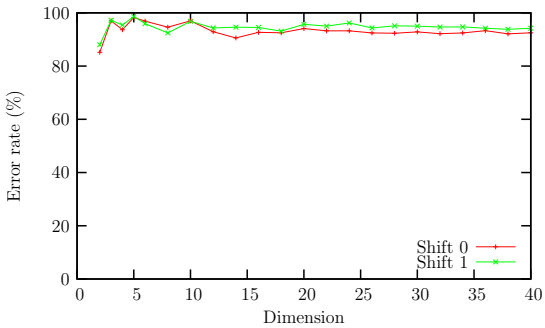
(e) *APC2-2*



(f) *APC2-3*

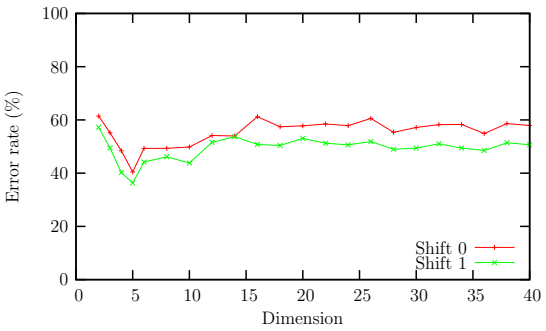


(g) *MBT4-3*

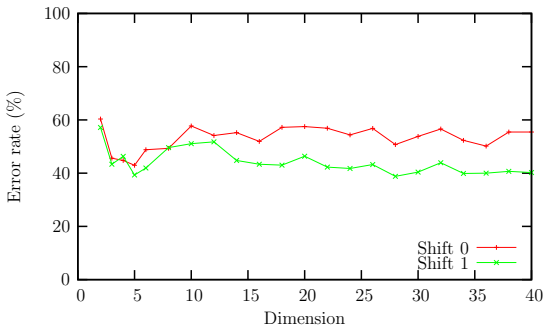


(h) *MBT5-3*

Figure 6.21: *Group 3 test results, with shifting, for RBF projection trained using Box-Cox metric distances.*



(i) *ADU2-1*



(j) *ADU3-1*

Figure 6.21: *Group 3 test results, with shifting, for RBF projection trained using Box-Cox metric distances (cont.)*

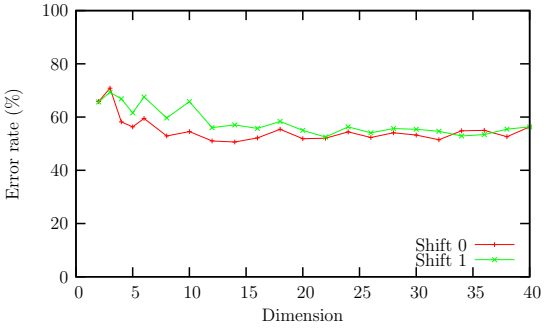
the error rate above 20 dimensions. For both of these data sets, shift 0 gives lower error rate than shift ±1. ADU2-1 and ADU3-1, in figures 6.21i and 6.21j respectively, also achieve better than random results, especially when shifting is included as this produces lower error rates. The error rate for both of these data sets drops rapidly to a minimum at 5 dimensions and thereafter increases. As for MBT2-2 and MBT2-3, there is no trend in the error rate with respect to dimension above 20 dimensions.

Plots of error rate against feature-space dimension are shown for the data sets in Group 4 in figure 6.22. Data sets for MBT2, in figures 6.22a through 6.22c, yield error rates better than random chance. The region of the curves at small dimension are marked by the trend for error rate to decrease with increasing dimension, but are also marked by variability. The decreasing trend and the variability largely level off above 15 dimensions. The lowest error rates are obtained by not using shifting. Results for ADU3, in figures 6.22j through 6.22m, also are better than random chance. The error rate drops rapidly to a global minimum at approximately 5 dimensions before rising again to approach the error rate for random chance. The effect of shifting is mixed, but the curves for shift ±1 are very close to those for shift 0, especially in the region of the minimum at approximately 5 dimensions.

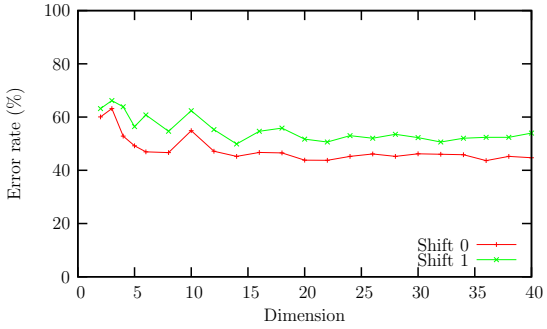
Plots of error rate against feature-space dimension are shown for the data sets in Group 5 in figure 6.23. The error rates for the MBT3 data sets, in figures 6.23a through 6.23d, are very much at the level of random chance. Those for MBT5, in figures 6.23e and 6.23f, are higher than for random chance.

6.5.2 Summary of results

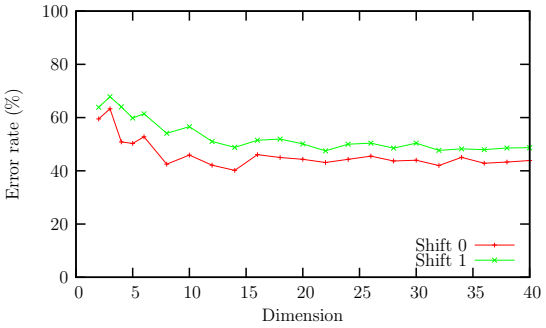
For comparison of results across data sets, the error rate is plotted for the 1-NNR using 20 features, for each of the data sets, in figure 6.24.



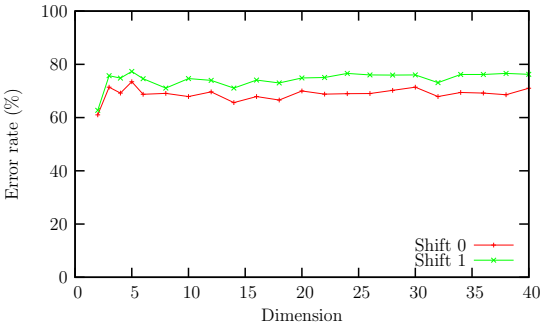
(a) *MBT2-4*



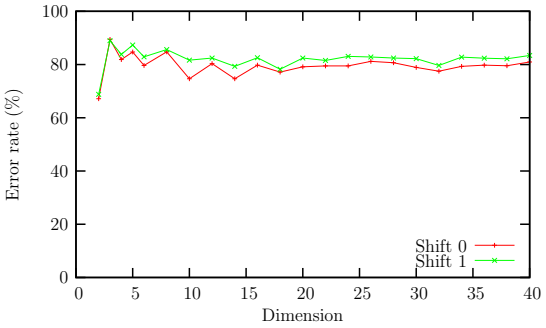
(b) *MBT2-5*



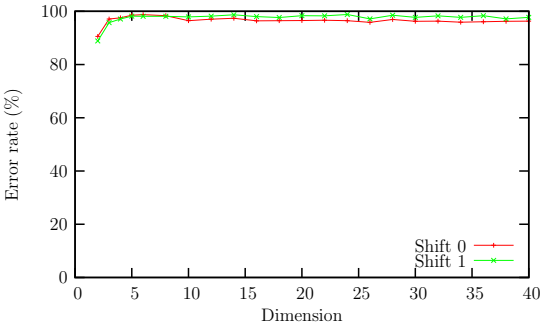
(c) *MBT2-6*



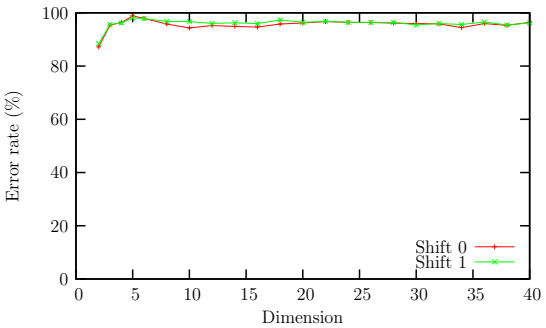
(d) *MBT4-1*



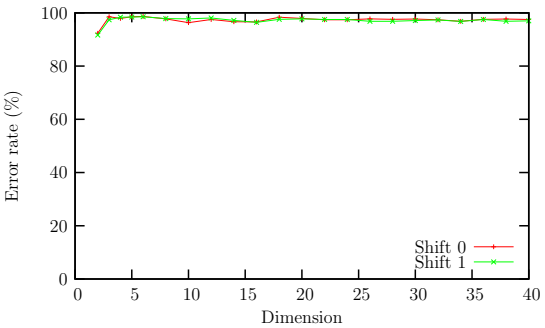
(e) *MBT4-2*



(f) *MBT5-4*

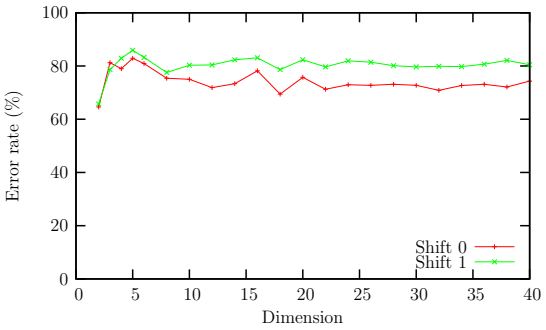


(g) *MBT5-5*

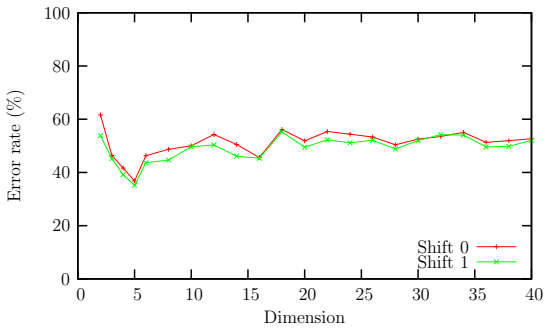


(h) *MBT5-6*

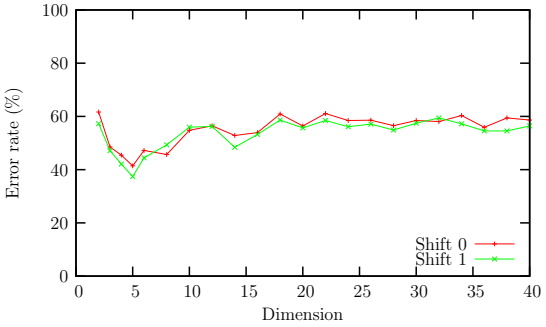
Figure 6.22: *Group 4 test results, with shifting, for RBF projection trained using Box-Cox metric distances.*



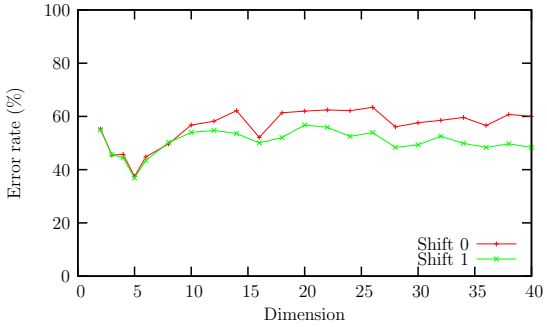
(i) *MBT4-4*



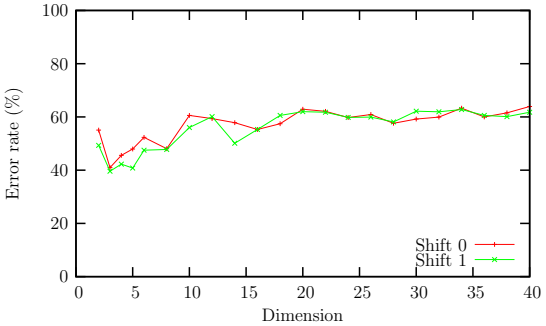
(j) *ADU3-2*



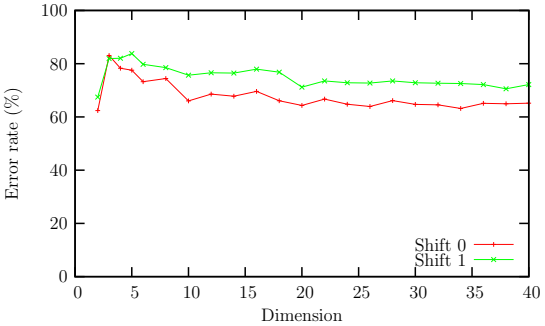
(k) *ADU3-3*



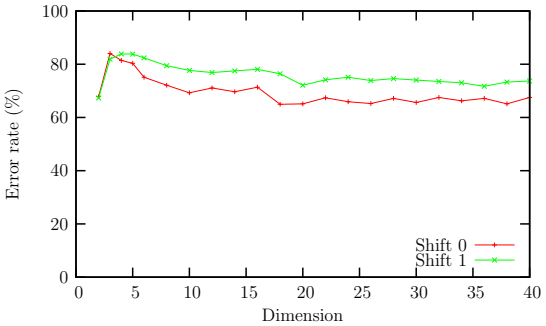
(l) *ADU3-4*



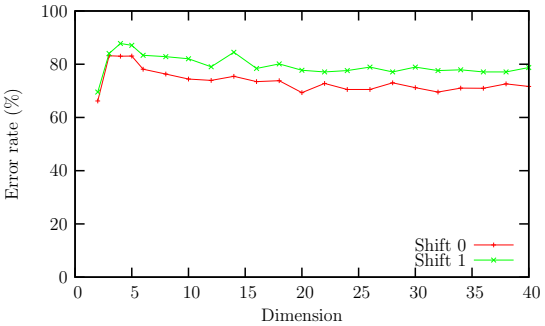
(m) *ADU3-5*



(n) *MBT3-7*

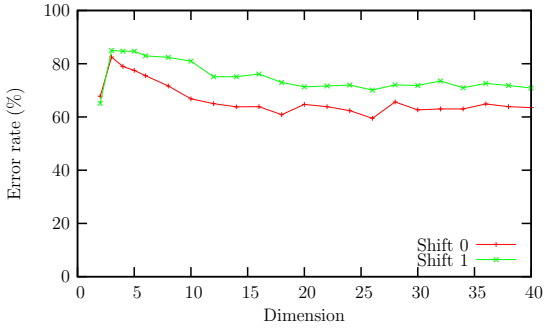


(o) *MBT3-8*



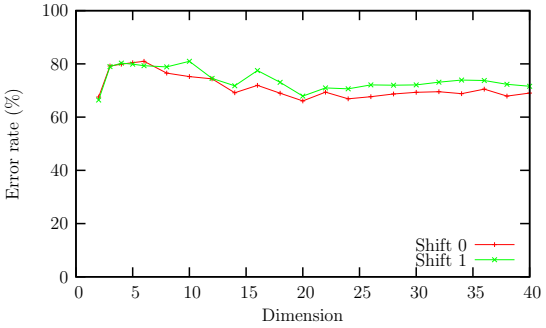
(p) *MBT3-9*

Figure 6.22: *Group 4 test results, with shifting, for RBF projection trained using Box-Cox metric distances (cont.)*

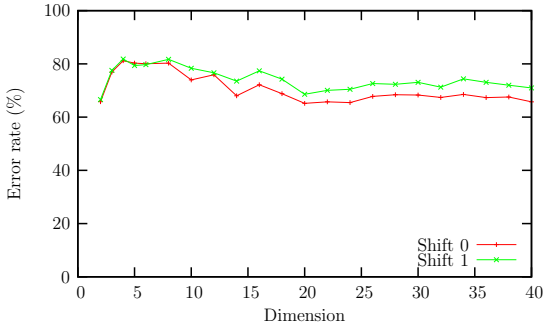


(q) *MBT3-10*

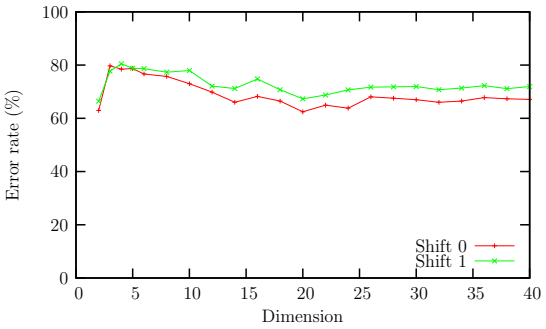
Figure 6.22: *Group 4 test results, with shifting, for RBF projection trained using Box-Cox metric distances (cont.)*



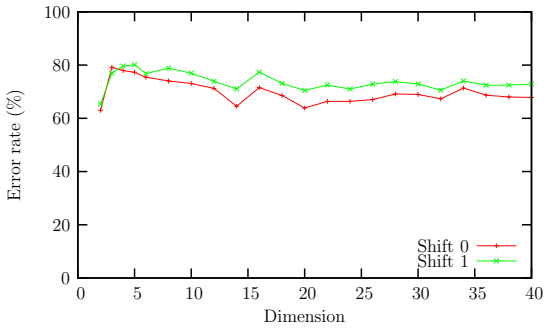
(a) *MBT3-1*



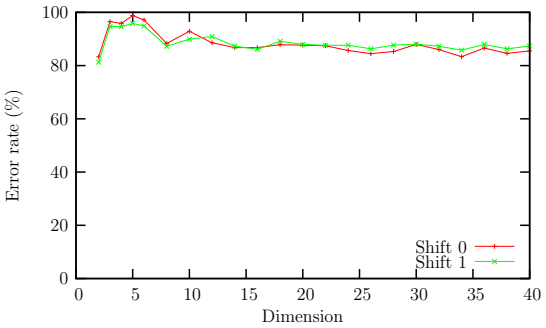
(b) *MBT3-2*



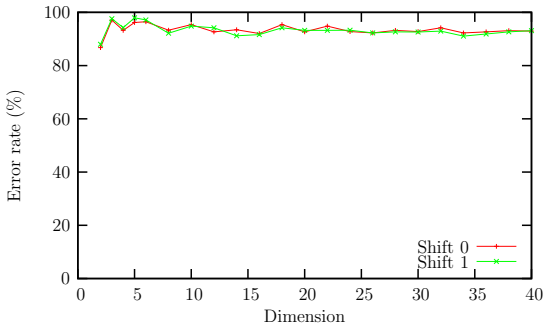
(c) *MBT3-4*



(d) *MBT3-5*

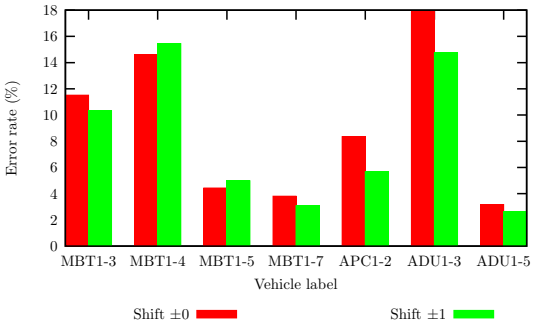


(e) *MBT5-1*

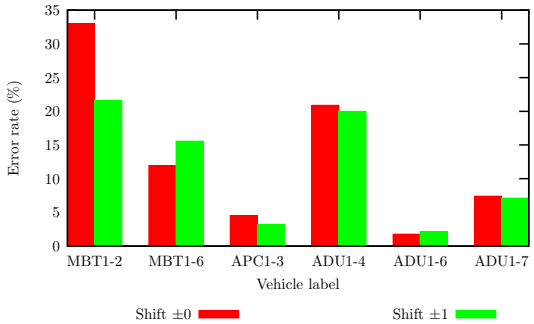


(f) *MBT5-2*

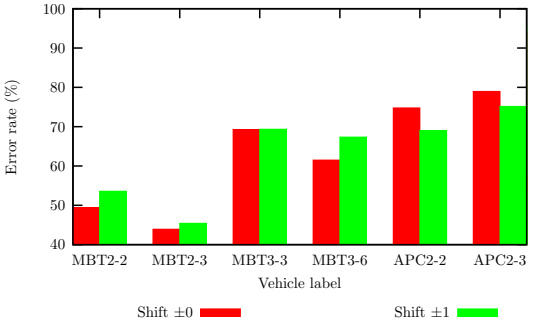
Figure 6.23: *Group 5 test results, with shifting, for RBF projection trained using Box-Cox metric distances.*



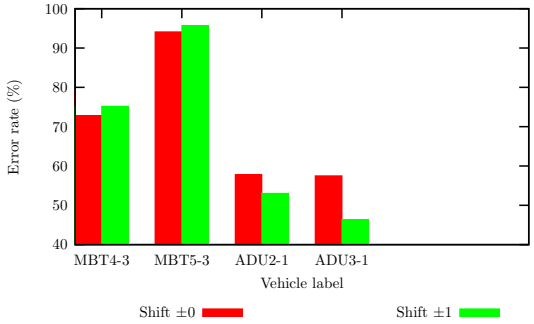
(a) *Group 1*



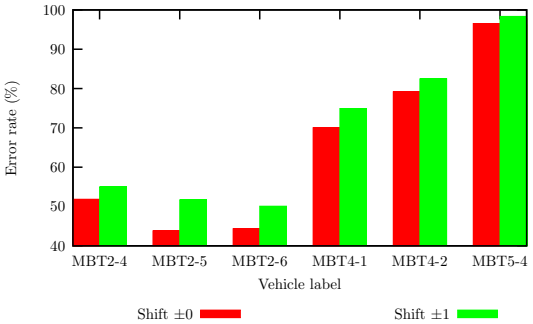
(b) *Group 2*



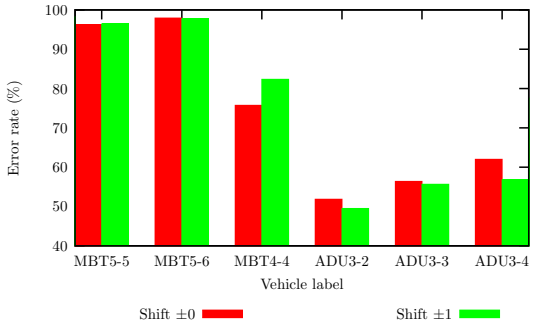
(c) *Group 3*



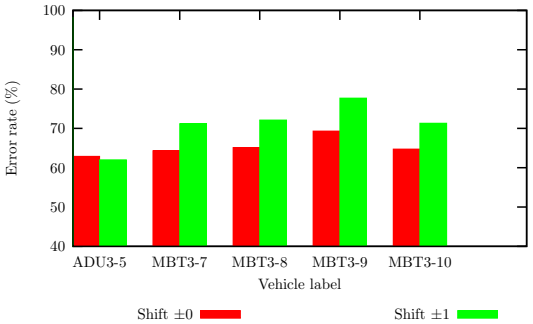
(d) *Group 3 (cont.)*



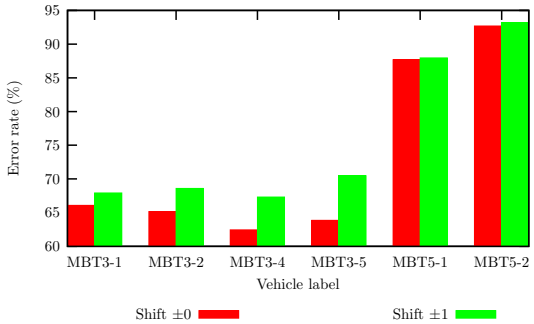
(e) *Group 4*



(f) *Group 4 (cont.)*



(g) *Group 4 (cont.)*



(h) *Group 5*

Figure 6.24: *Error rate from the 1-NNR using 20 features calculated by the RBFN trained using the Box-Cox metric distances. The larger groups (3 and 4) are split across subfigures, but the y-range is kept constant.*

The summary results for Group 1 are shown in figure 6.24a. The lowest error rates are for MBT1-5, MBT1-7, and ADU1-5. The error rates for APC1-2 are a little higher, but the highest error rates are for MBT1-3, MBT1-4, and ADU1-3. This ranking is not due to vehicle type as ADU1 has both the smallest error rate (ADU1-5) and the highest (ADU1-3). These summary results show some error rates increase with shifting, but the majority decrease.

The summary results for Group 2 are shown in figure 6.24b. The lowest error rates are for APC1-3 and ADU1-6. The next highest is for MBT1-6, then ADU1-4. Finally, the highest error rate is for MBT1-2. ADU1 has the lowest error rate as well as the second highest and so rank ordering of the results for this group also does not align with vehicle type. The effect of shifting is mixed. Most data sets show very little change. One, MBT1-2, shows a drop of more than 10 percentage points. Another, MBT1-6, shows an increase of approximately three percentage points. Thus there is not even consistency of the effect of shifting within vehicle type: the two MBT1 data sets respond differently.

The summary for Group 3 is shown in figures 6.24c and 6.24d. Data sets that have error rates lower than random chance are MBT2-2, MBT2-3, MBT3-6 (barely), ADU2-1, and ADU3-1. In the case of MBT3-6 the inclusion of shifting pushes the error rate up close to the level of random chance. For both ADU2-1 and ADU3-1 without shifting, the error rate is close to random and the inclusion of shifting reduces the error rate.

The summary for Group 4 is shown in figures 6.24e, 6.24f, and 6.24g. The data sets for MBT2 (MBT2-4, MBT2-5, and MBT2-6 in figure 6.24e) again have error rates lower than that for random chance. In figure 6.24f the ADU3-2 error rate is better than random, ADU3-3 is higher but still better than random, and ADU3-4 is very close to random but just under. In figure 6.24g the error rate for ADU3-5 is at the level of random chance. The data sets for MBT3 (MBT3-7, MBT3-8, MBT3-9, and MBT3-10 in figure 6.24g) have error rates just below the level of random chance without shifting, but the inclusion of shifting pushes them above.

The summary for Group 5 is shown in figure 6.24h. The MBT3 data sets (MBT3-1, MBT3-2, MBT3-4, and MBT3-5) achieve error rates better than random. This is especially true without shifting. Shifting pushes the error rate up towards the level of random chance, which MBT3-5 just reaches. Both of the MBT5 data sets (MBT5-1 and MBT5-2) perform worse than randomly.

6.5.3 Discussion

The error rates in Groups 1 and 2 are lower than those in Groups 3–5. This reflects the fact that Groups 1 and 2 comprise only vehicles present during training and Groups 3–5 comprise only vehicles not present during training. The rapid fall in error rate with increasing

dimension for Groups 1 and 2 suggests that as dimension increases, data structure useful for discrimination is being captured. The error rate then levels off when the addition of further dimensions adds only noise with no discriminatory information. This is consistent with the level of error rate attained being much less than random. In Groups 3 and 4 the data sets that achieve error rates better than random often show some dependence on dimension, although such dependence is either a weak trend or associated with a large amount of variability. These characteristics suggest that the feature extraction is finding some discriminatory information for those data sets but struggles to do so consistently.

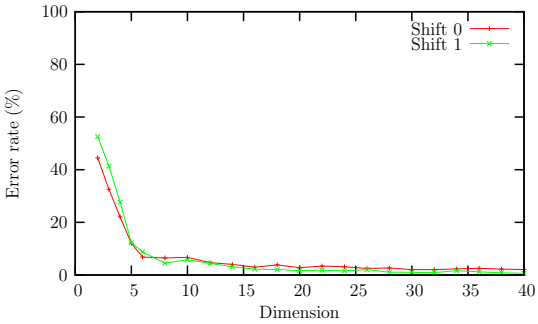
For Group 1 the rank ordering of the results is explicable when the configuration of the vehicles is considered. Hatches on the MBT have the least effect. Toolboxes on the MBT have a larger effect. The orientation of the turret on the ADU has the largest effect. The same is true for Group 2. The orientation of the small, smooth, and symmetrical APC turret has the smallest effect, along with the position of the tracking radar and engine running for the ADU. Opening the hatches on the ADU has a bigger effect. Draping camouflage netting over the whole surface of the MBT retains it basic shape but distorts a lot of the internal radar signature and leads to a higher error rate. An even higher error rate is associated with the rotation of the ADU turret. The highest error rate, however, belongs to MBT1-2, and this is possibly because of the open toolboxes.

6.6 Box-Cox transformed data space

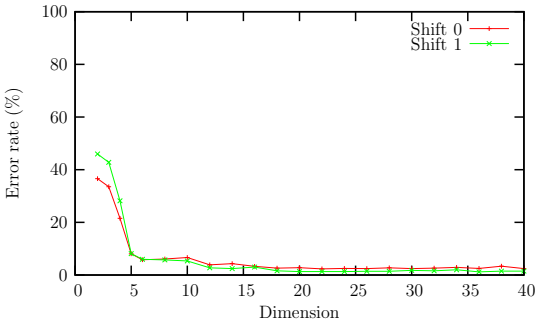
6.6.1 Error rate with number of features

The plots of error rate against feature-space dimension for the data sets in Group 1 are shown in figure 6.25. There is a very rapid initial fall in error rate with increasing dimension. The error rate levels off between 5 and 10 dimensions. This levelling off is quite abrupt. The curves of error rate for the two cases with and without shifting are largely indistinguishable.

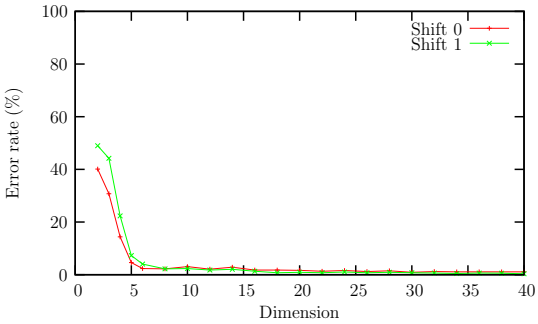
The plots of error rate against feature-space dimension for the data sets in Group 2 are shown in figure 6.26. With the exception of MBT1-2, the curves exhibit the same characteristics as in Group 1: a rapid drop in error rate that abuptly levels off between 5 and 10 dimensions (usually closer to 5) and nearly identical whether using shifting of the input images or not. The error rate for MBT1-2 in figure 6.26a behaves quite differently. There is a trend of decreasing error rate with increasing dimension that is most pronounced for shift ± 1, but even with this shifting the trend is quite weak and there is no easily determined shoulder that can be used to partition the curve into 'falling' and 'levelling off' regions. Using fewer than 10 dimensions, shift 0 gives the lowest error rates; above 10 dimensions, shifting consistently produces distinctly lower error rates.



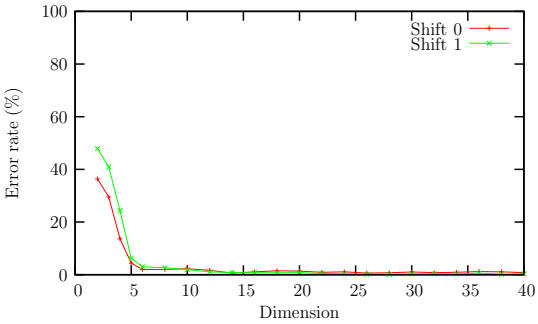
(a) *MBT1-3*



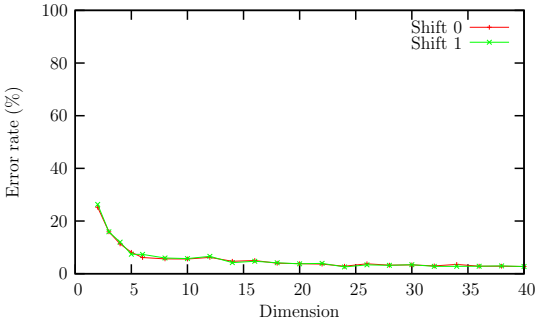
(b) *MBT1-4*



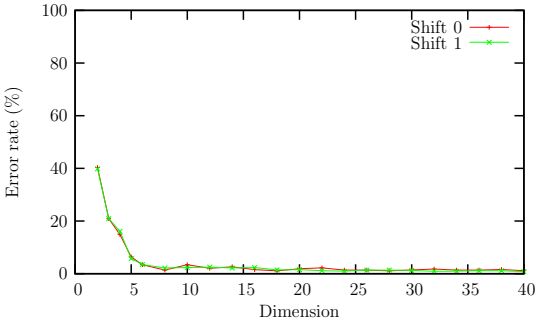
(c) *MBT1-5*



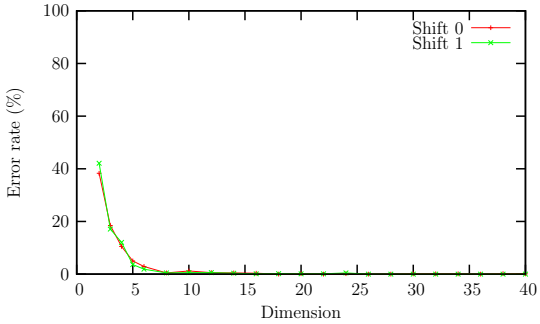
(d) *MBT1-7*



(e) *APC1-2*

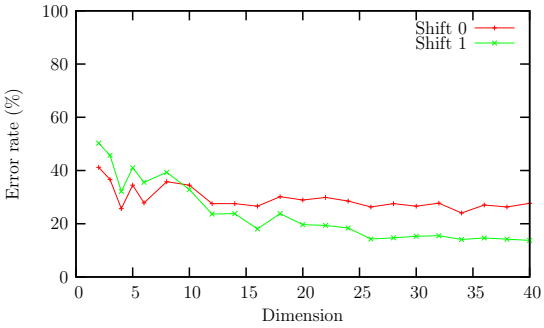


(f) *ADU1-3*

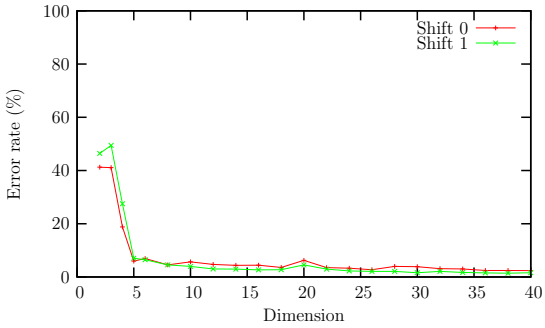


(g) *ADU1-5*

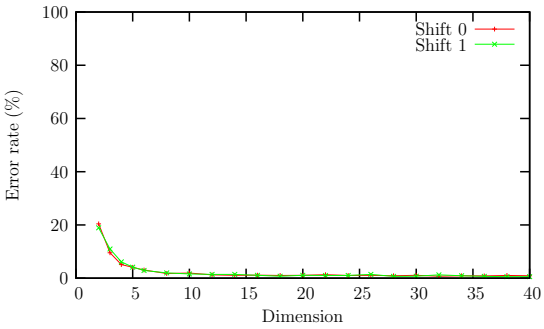
Figure 6.25: *Group 1 test results, with shifting, for RBF projection trained on Box-Cox transformed data and used to project Box-Cox transformed data.*



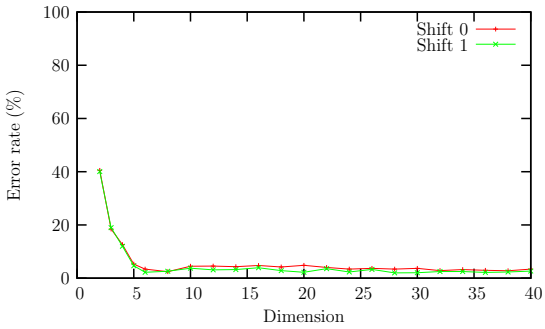
(a) *MBT1-2*



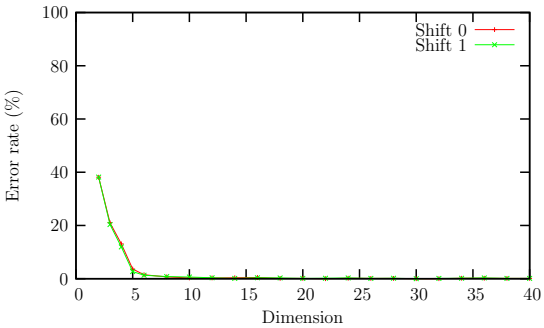
(b) *MBT1-6*



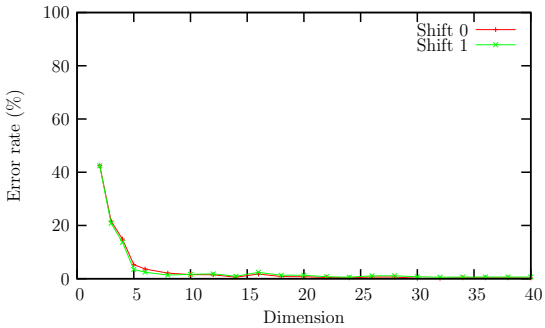
(c) *APC1-3*



(d) *ADU1-4*



(e) *ADU1-6*



(f) *ADU1-7*

Figure 6.26: *Group 2 test results, with shifting, for RBF projection trained on Box-Cox transformed data and used to project Box-Cox transformed data.*

The plots of error rate against feature-space dimension for the data sets in Group 3 are shown in figure 6.27. In keeping with the fact that this group comprises only vehicles not used in training, the error-rate curves for the Group 3 data sets are very different to those for the first two groups. There is a distinct local minimum at 6 dimensions for MBT2-2 in figure 6.27a and MBT2-3 in figure 6.27b that is approximately 40 percentage points below the level of random chance. A similar pattern is seen for ADU2-1 in figure 6.27i and ADU3-1 in figure 6.27j where a broad local minimum occurs between 5 and 10 dimensions. The two minima are approximately 50 and 40 percentage points better than random. Out of the remaining data sets, MBT3-3 in figure 6.27c, MBT3-6 in figure 6.27d, and MBT4-3 in figure 6.27g score better than random chance, with an initial variability with respect to dimension that settles down after approximately 10–15 dimensions. For all data sets there is little difference between shift 0 and shift ±1.

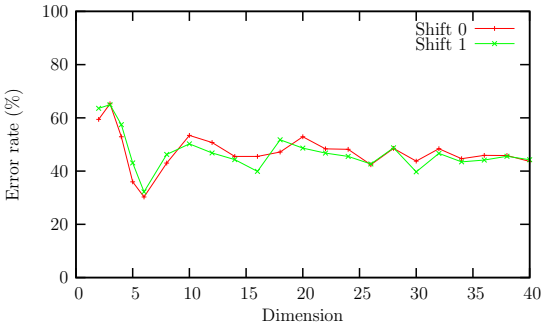
The plots of error rate against feature-space dimension for the data sets in Group 4 are shown in figure 6.28. As with Group 3, the MBT2 data sets, in figures 6.28a through 6.28c, all have a sharp local minimum at 6 dimensions. The minima are approximately 30 to 40 percentage points below the level of random chance. Very similar local minima are observed for the ADU3 data sets, shown in figures 6.28j through 6.28m. Only for ADU3-4 (figure 6.28l) is the sharpness of the minimum diminished. The depths of the minima approach 50 percentage points below the level of random chance. Several other data sets perform better than random chance after an initial rise to a local maximum between 5 and 10 dimensions. Furthermore, this local maximum is usually centred at 6 dimensions. For these data sets, error rate generally levels off by 15–20 dimensions. A good example of this is MBT4-4 in figure 6.28i with an error rate that rises to approximately 90% at 6 dimensions and falls to approximately 40% at 20 dimensions. The shift 0 and shift ±1 curves closely follow each other.

The plots of error rate against feature-space dimension for the data sets in Group 5 are shown in figure 6.29. Graphs for the same vehicle type show the same structure. For MBT3, in figures 6.29a through 6.29d, there is a local maximum in error rate, of approximately the level of random chance, between 5 and 10 dimensions. The error rate then levels off between 10 and 15 dimensions to a value below the level of random chance. Error rates for the two MBT5 data sets, in figures 6.29e and 6.29f, are similar and very high. The results obtained with and without shifting are also similar.

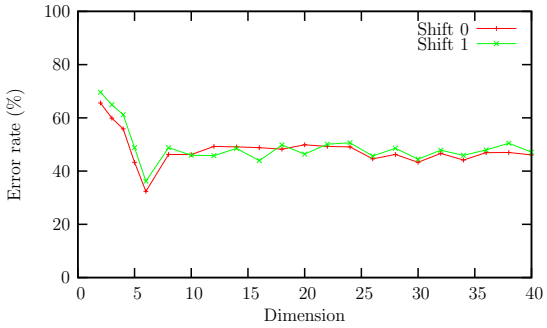
6.6.2 Summary of results

For comparison of results across data sets, the error rate is plotted for the 1-NNR using 20 features, for each of the data sets, in figure 6.30.

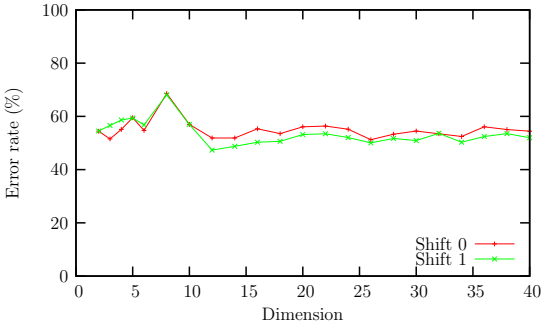
The summary results for Group 1 are plotted in figure 6.30a. The error rates for all



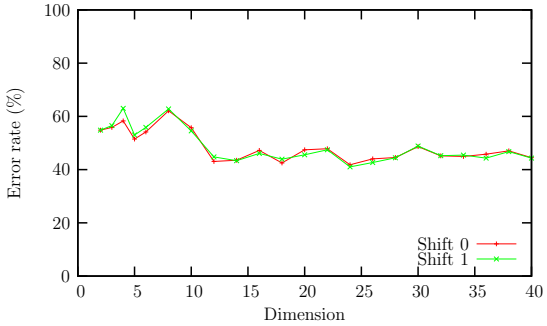
(a) *MBT2-2*



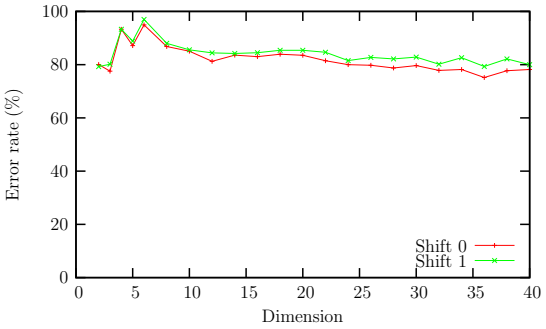
(b) *MBT2-3*



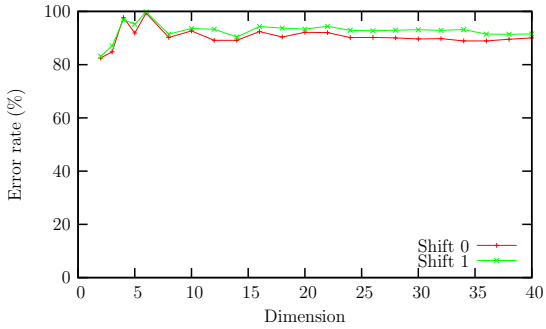
(c) *MBT3-3*



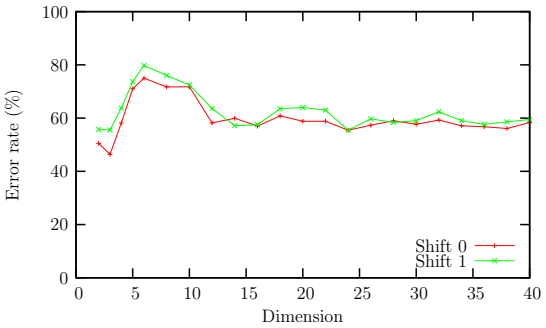
(d) *MBT3-6*



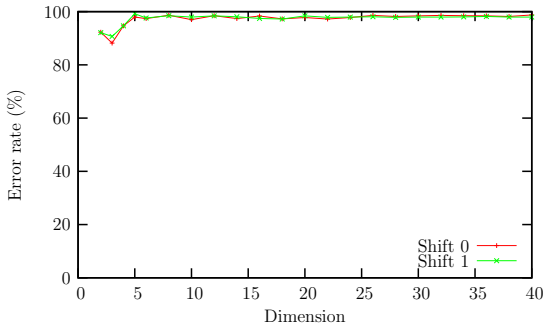
(e) *APC2-2*



(f) *APC2-3*

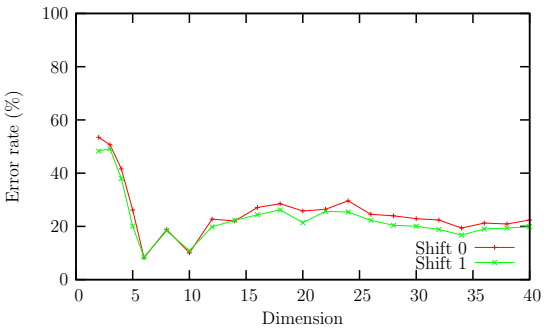


(g) *MBT4-3*

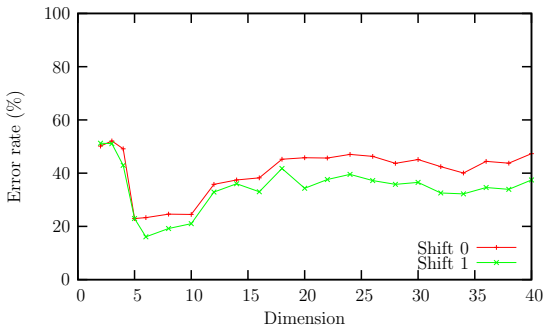


(h) *MBT5-3*

Figure 6.27: *Group 3 test results, with shifting, for RBF projection trained on Box-Cox transformed data and used to project Box-Cox transformed data.*



(i) *ADU2-1*



(j) *ADU3-1*

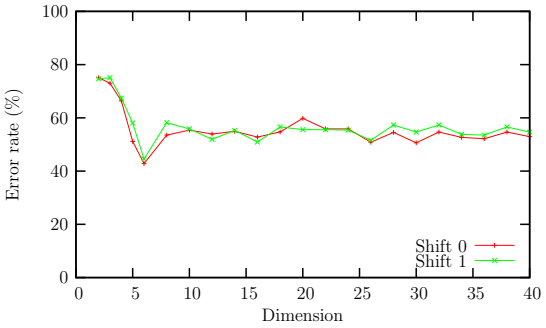
Figure 6.27: *Group 3 test results, with shifting, for RBF projection trained on Box-Cox transformed data and used to project Box-Cox transformed data (cont.)*

data sets are low; the highest is below 4%. The lowest error rate belongs to ADU1-5, corresponding to a configuration change of the gun raised by 45°. The highest error rate is for APC1-2, corresponding to the APC with hatches open. On this scale, the inclusion of shifting is seen to result in a large relative drop in error rate for MBT1, a smaller drop for ADU1, and an even smaller drop for APC1.

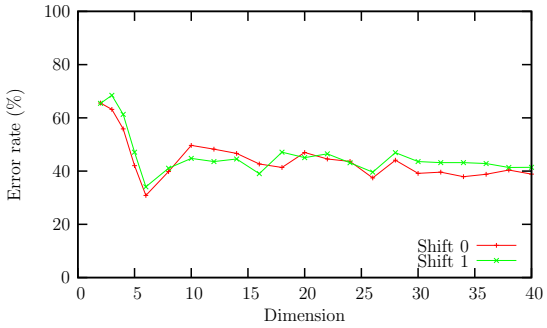
Summary results for Group 2 are given in figure 6.30b. The error rate for ADU1-6 is very close to zero. This data set corresponds to the vehicle's tracking radar lowered and the engine running. APC1-3 and ADU1-7 are the next lowest. The error rate for MBT1-2 stands out as being particularly high. The inclusion of shifting results in a large relative decrease in error rate for this data set. The next two highest error rates, MBT1-6 and ADU1-4, also show a reduction with shifting. The data sets with the lowest error rates (APC1-3, ADU1-6, and ADU1-7) show either no change with shifting or an increase.

The summary results for Group 3 are plotted in figures 6.30c and 6.30d. The MBT2 and MBT3 data sets, in figure 6.30c, have similar error rates that are up to 20 percentage points better than random chance. Approximately 10 percentage points better than random is MBT4-3 in figure 6.30d. Also yielding results better than random are ADU2-1 and ADU3-1 in figure 6.30d, with ADU2-1 being the lowest in this group. The effect of performing shifting is mixed, with both increases and decreases in error rate. The magnitude of any change is usually small.

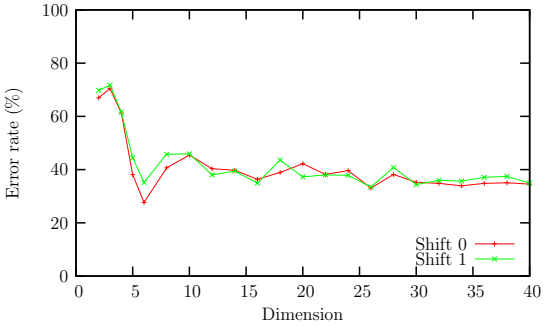
Summary results for the data sets in Group 4 are given in figures 6.30e through 6.30g. All of the data sets for MBT2, MBT3, and MBT4, but not MBT5, have error rates better than random. For example, in figure 6.30e MBT2-4 and MBT4-2 are approximately 10 percentage points better than random and MBT2-6 is approximately 30 percentage points better. The results for ADU3 are better than random but tend to require shifting of the input images to achieve this. ADU3-4 in figure 6.30f is the exception to this: it scores better than random without requiring shifting. Overall, the effect of shifting is mixed. For



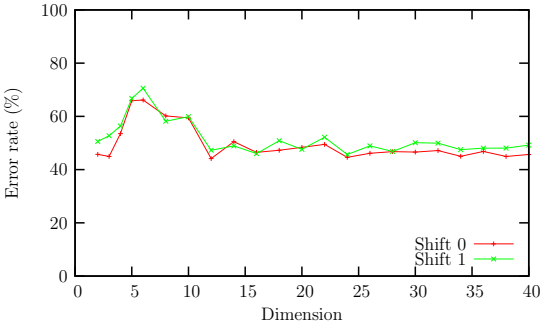
(a) *MBT2-4*



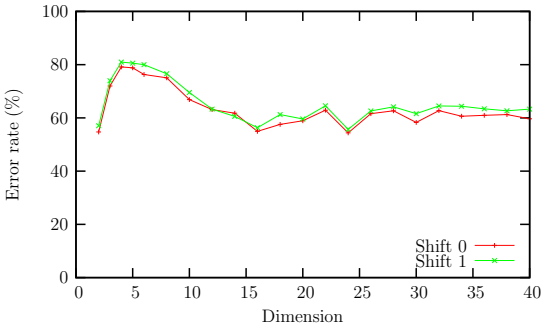
(b) *MBT2-5*



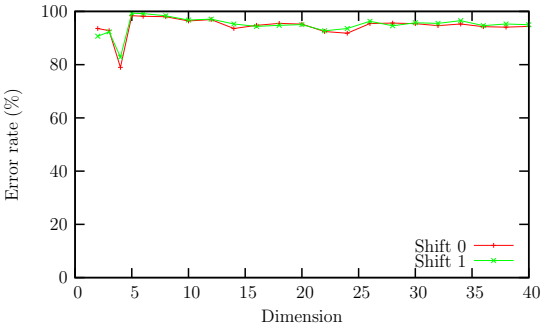
(c) *MBT2-6*



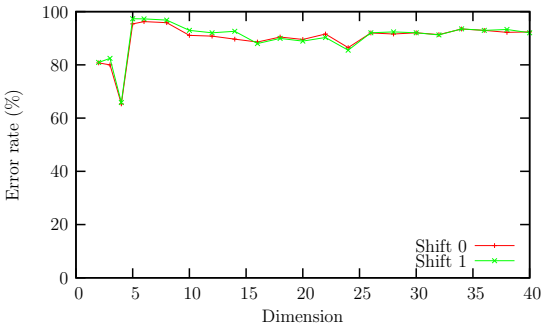
(d) *MBT4-1*



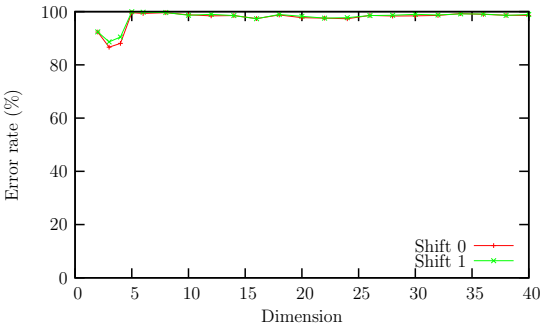
(e) *MBT4-2*



(f) *MBT5-4*

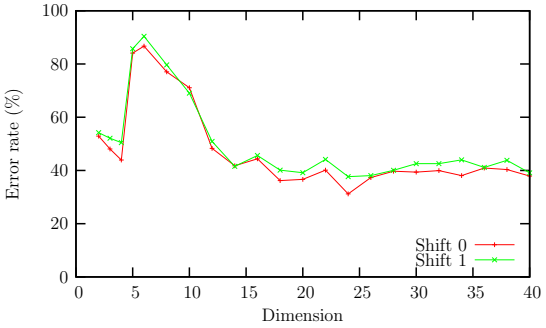


(g) *MBT5-5*

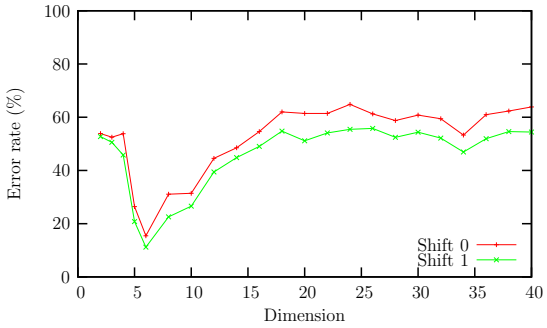


(h) *MBT5-6*

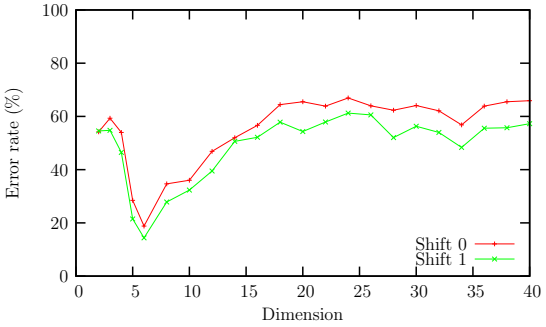
Figure 6.28: *Group 4 test results, with shifting, for RBF projection trained on Box-Cox transformed data and used to project Box-Cox transformed data.*



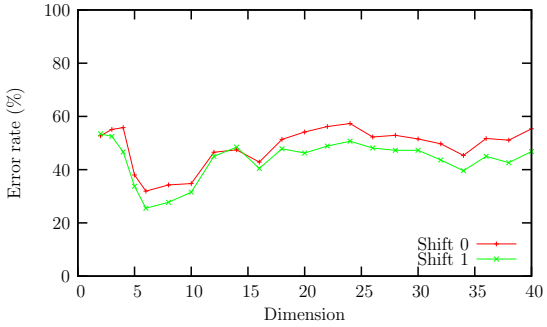
(i) *MBT4-4*



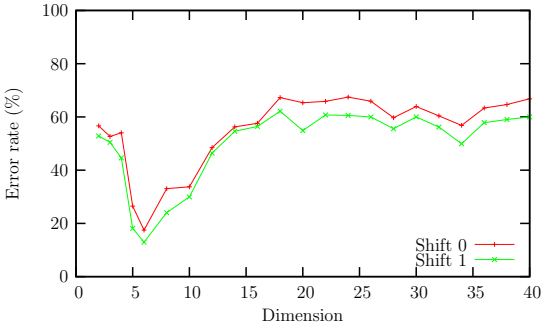
(j) *ADU3-2*



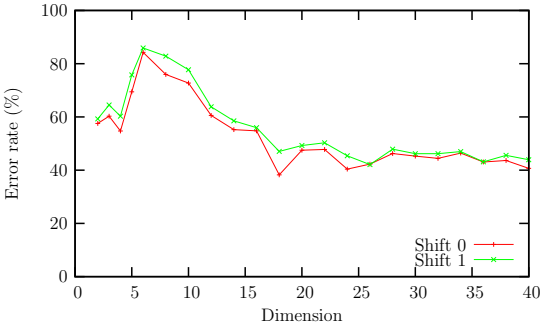
(k) *ADU3-3*



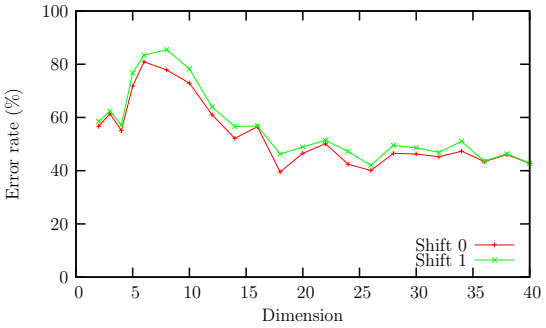
(l) *ADU3-4*



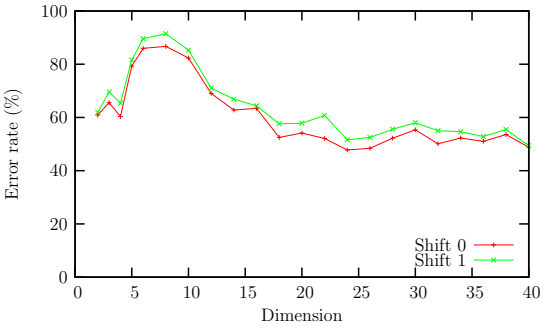
(m) *ADU3-5*



(n) *MBT3-7*

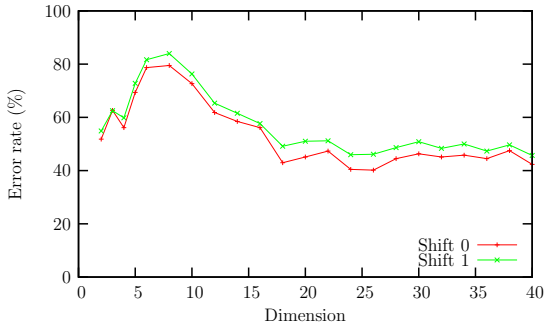


(o) *MBT3-8*



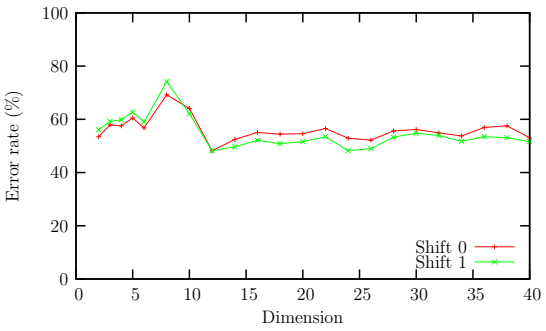
(p) *MBT3-9*

Figure 6.28: *Group 4 test results, with shifting, for RBF projection trained on Box-Cox transformed data and used to project Box-Cox transformed data (cont.)*

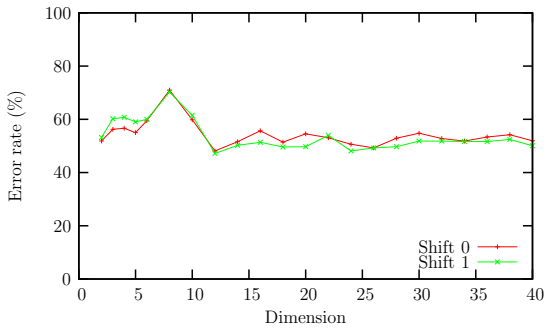


(q) *MBT3-10*

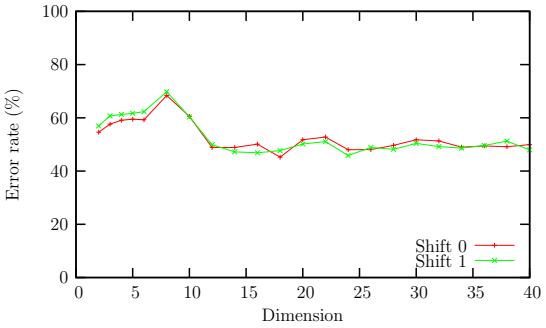
Figure 6.28: *Group 4 test results, with shifting, for RBF projection trained on Box-Cox transformed data and used to project Box-Cox transformed data (cont.)*



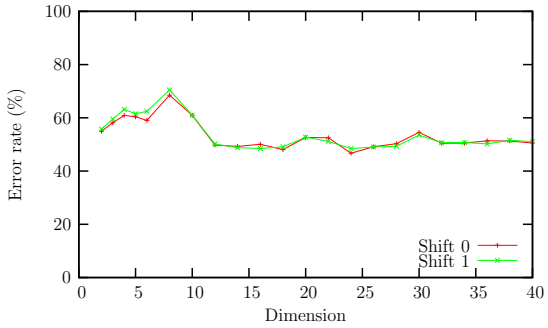
(a) *MBT3-1*



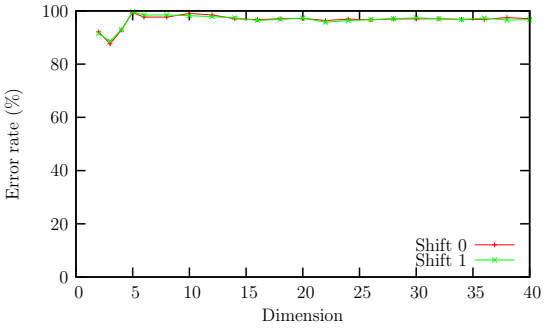
(b) *MBT3-2*



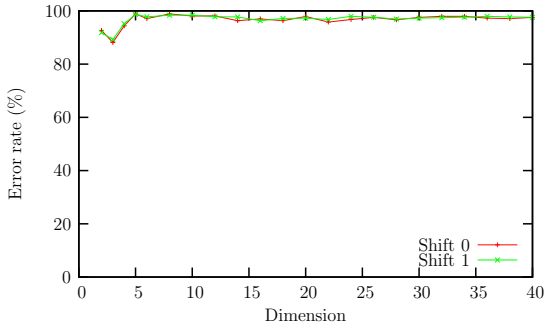
(c) *MBT3-4*



(d) *MBT3-5*

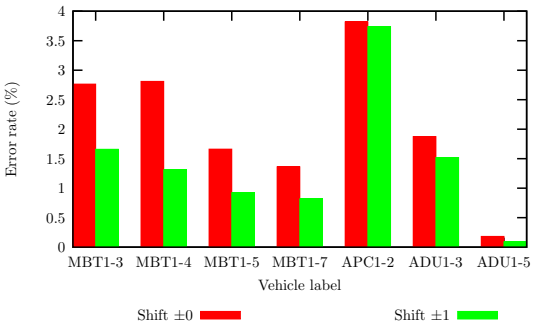


(e) *MBT5-1*

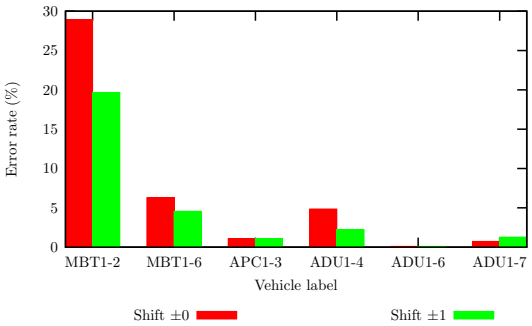


(f) *MBT5-2*

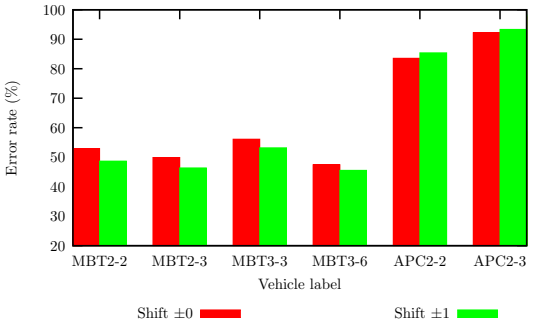
Figure 6.29: *Group 5 test results, with shifting, for RBF projection trained on Box-Cox transformed data and used to project Box-Cox transformed data.*



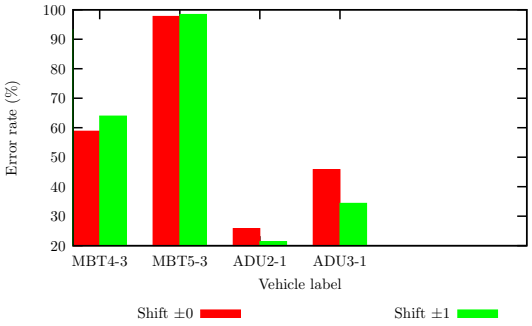
(a) *Group 1*



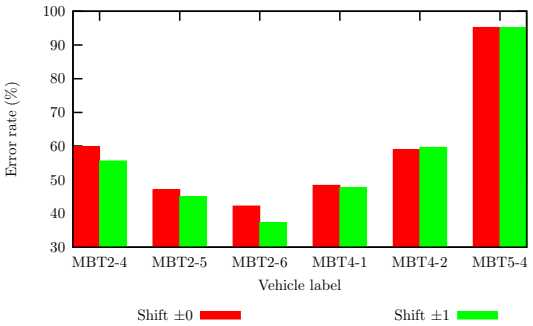
(b) *Group 2*



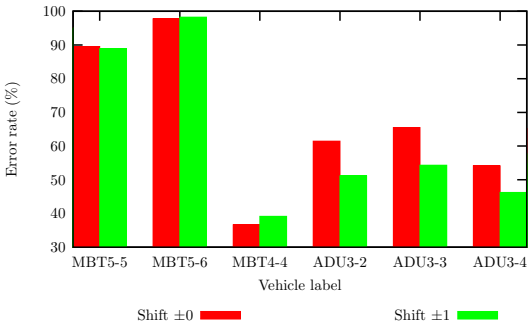
(c) *Group 3*



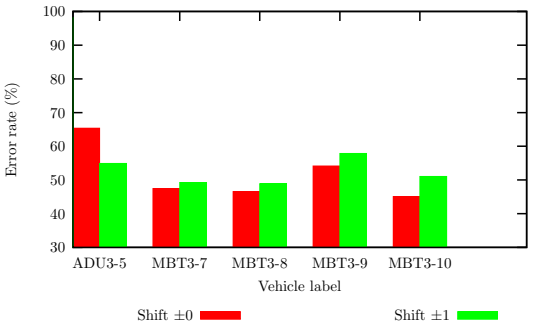
(d) *Group 3 (cont.)*



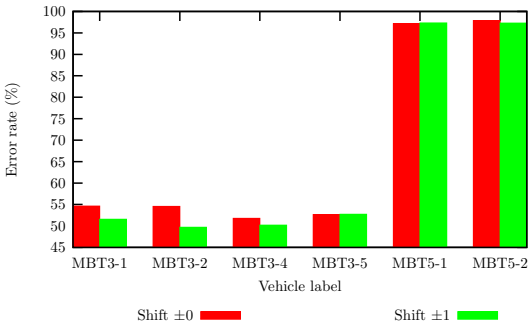
(e) *Group 4*



(f) *Group 4 (cont.)*



(g) *Group 4 (cont.)*



(h) *Group 5*

Figure 6.30: *Error rate from the 1-NNR using 20 features calculated by the RBFN trained using the Box-Cox transformed data and applied to Box-Cox transformed test data. The larger groups (3 and 4) are split across subfigures, but the y-range is kept constant.*

example, MBT2-4, MBT2-5, and MBT2-6 in figure 6.30e show a small decrease in error rate with shifting; MBT4-1 and MBT4-2, also in figure 6.30e, show a negligible change, and MBT3-7, MBT3-8, MBT3-9, and MBT3-10 in figure 6.30g show a small increase.

Finally, summary results for Group 5 are given in figure 6.30h. The error rates for the MBT3 data sets are similar and approximately 15–20 percentage points better than random. The error rates for the MBT5 data sets are also similar, but much worse than random. The inclusion of shifting causes a small decrease in the error rate for MBT3-1, MBT3-2, and MBT3-4, but no change for MBT3-5.

6.6.3 Discussion

For Groups 1 and 2, the error rate typically falls very rapidly with increasing dimension and falls to a very low value. For the other groups, a local minimum or maximum is frequently encountered between 5 and 10 dimensions. This local stationary point is usually sharp and located at 6 dimensions. This is also the number of dimensions at which the decrease in error rate usually levels off in Groups 1 and 2. It is clear that as the number of features extracted increases, for Groups 1 and 2, 6 dimensions marks the point at which no further features are useful for discrimination. These first 6 features are also useful for many of the non-training vehicles in Groups 3–5. The difference relative to Groups 1 and 2 is that the addition of extra features in Groups 3–5 is harmful and adds confusion, rather than being harmless or benign to discrimination. Some non-training vehicles, however, do require the additional features to achieve better than random error rates. There is no 'magical', generalizing set of 6 features that works across all vehicles. Nevertheless, something interesting, in terms of features for classification, does happen around 6 dimensions.

6.7 Summary

The effects of shifting the input test images prior to feature extraction are mixed, with the error rate decreasing for some data sets and increasing for others. This demonstrates that the feature extraction is not robust to the position of the vehicle within the image chip.

For both the unsupervised and supervised loss functions, the error rates within Groups 1 and 2 are lower than those within Groups 3–5. This reflects the division between groups that comprise only vehicles used for training and groups that comprise only vehicles not used for training. Training the supervised loss function with $\rho = 0$ packs discriminatory information into fewer features than for $\rho > 0$, but does not improve upon the error rate that is attainable using $\rho > 0$ and more features.

Training the extracted features to maximize the separation of the classes in the training data reduces the disparity between many of the different configurations of the same vehicle in the test data, with the exception of MBT1-2 in Group 2. This is at the expense, however, of an overall increase in the error rate. Furthermore, there are also then some data sets in Groups 3 and 4, which comprise only vehicles not present during training, that have error rates lower than for MBT1-2 in Group 2, which is a training vehicle.

The attempt to train the feature extraction stage to learn the Box-Cox transformation clearly was not as successful as simply applying the Box-Cox transformation as a separate pre-processing step prior to feature extraction. The latter produces much lower error rates for Groups 1 and 2 than any of the other approaches. It usually achieves this by using only a 6 dimensional feature space. The error rate then levels off abruptly, with increasing number of features producing no further decrease in error rate.

The levelling off of error rate with dimension for Groups 1 and 2 when using the Box-Cox transformation as a pre-processing step is in common with the results from the other RBFN approaches. For these two groups, all approaches typically show little change in error rate as the number of features increases above a certain value. The responses for Groups 3–5, however, are very different. Many data sets in these groups show a local minimum or maximum centred about 6 dimensions. The addition of extra dimensions for the other approaches generally results in little change in performance, even for the non-training-vehicle data sets.

Chapter 7

Support vector machine

7.1 Introduction

Support Vector Machines (SVMs) are kernel-based algorithms, and are currently the focus of much attention for regression and classification. As such they offer an important technique for comparison with the RBFN approach. This chapter describes experiments using a Support Vector Classifier (SVC).

There is now a wide range of SVM software available and links to many can be found on the World Wide Web (WWW) [58, 111]. Compiling a shortlist of candidate packages and then evaluating each of them to arrive at a preferred choice for a particular application would be a mini-project in itself. Clearly an important requirement for a desirable package is its performance, but other considerations for this particular work included:

- licensing conditions,
- ease of use (especially to non-SVM-expert users),
- multi-class capability,
- compatibility with a GNU/Linux environment.

The SVM software chosen was the C-SVC, using an RBF kernel, in LIBSVM [14]. LIBSVM's license is unrestrictive and it can even be used in commercial products, its design is motivated by the desire to simplify the use of SVMs as a tool, it supports multiple classes via a built-in one-vs-one voting algorithm [51]; and it works on a GNU/Linux system. LIBSVM won the EUNITE [37] worldwide competition on electricity load prediction and two of three IJCNN [53] competitions. It has also been used to generate results for a number of papers, including a benchmarking exercise [80] where the SVM results were compared with those obtained from a number of other regression and classification

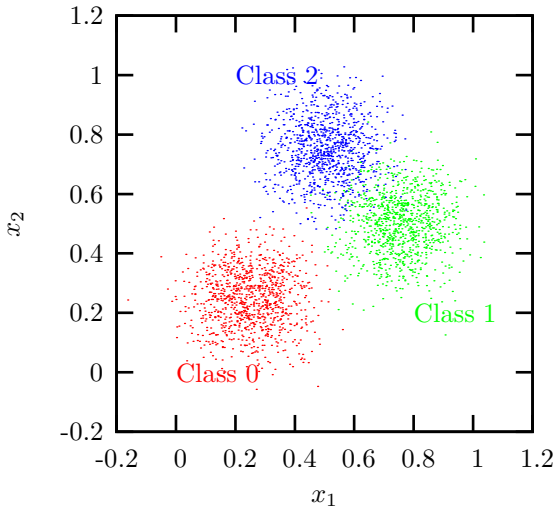


Figure 7.1: *Training data comprising 1000 samples per class of the three-class illustration problem.*

techniques and LIBSVM was found generally to perform well on all data sets. LIBSVM includes tools for scaling the input data, training and cross-validation, and classification.

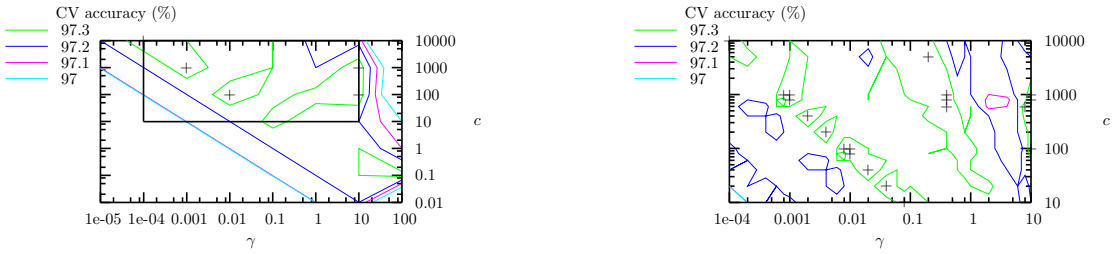
To visualize the behaviour of the software, experiments were first performed on an illustrative problem comprising three classes in two dimensions with Gaussian noise. These are described in section 7.2. Experiments were then conducted on the ISAR data, as described in section 7.3, with the results described in section 7.4. A summary and conclusions are finally presented in section 7.5.

7.2 Illustration problem

7.2.1 Introduction

Initial explorations with LIBSVM were performed on an artificially constructed problem comprising three classes in two dimensions: *class 0* with mean $[0.25, 0.25]^{\mathrm{T}}$, *class 1* with mean $[0.75, 0.5]^{\mathrm{T}}$, and *class 2* with mean $[0.5, 0.75]^{\mathrm{T}}$. Pseudo-random Gaussian noise of variance 0.1 was added to each dimension of each class. This simple data set facilitated easy visualization and validation of the various stages in LIBSVM.

The training data set comprised 1000 points for each class. This is shown graphically in figure 7.1. The data were contrived to have a noticeable overlap between classes 1 and 2, with class 0 reasonably well separated from the other two.



(a) *Coarse grid.* (b) *Fine grid.*

Figure 7.2: *Contour plots of LIBSVM cross-validation accuracy on illustration problem. The region of the coarse-grid search used for the fine-grid search is marked by a box. The locations of the peaks are marked by '+'. The peak value achieved in both was* 97.37%. *Many of the peak values on the fine grid can be seen to occupy a ridge between* $(10^{-4}, 10000)$ *and* $(0.1, 10)$.

7.2.2 Model selection by cross-validation

For the RBF kernel, LIBSVM has two free parameters to fix prior to obtaining an SVM model. These parameters are γ, the RBF width; and c, the cost. LIBSVM has a built-in k-fold cross-validation algorithm, leaving the user only to specify the value of k. By using the cross-validation accuracy as an estimate of the generalization error, the pair of parameters are chosen that can be expected to have good performance on future data. In the limit as k increases, the cross-validation error tends towards the leave-one-out (LOO) error. The LOO error is an almost unbiased estimate of the generalization error [119, p.417], however, it rapidly becomes computationally expensive to calculate for large training sets. The value $k = 5$ has been found [1] to be a good splitting strategy for SVM model selection, indeed, this is the default value used in LIBSVM. A 'traditional default' [36, p.484] value of k is $k = 10$, and this was used for these experiments.

The procedure for exploring the potential parameter space was to perform a search over a relatively coarse 'grid' of (γ, c). The search was subsequently refined using smaller increments in regions of interest as appropriate. A contour plot of the cross-validation accuracies obtained from the coarse grid search is shown in figure 7.2a. The resulting surface does not reveal a unique maximum; rather, it exhibits a number of potential regions of interest. A subregion that contained these regions of interest was identified for a fine grid search. The resulting contour plot using the finer grid is shown in figure 7.2b.

It appears, from figure 7.2b, that a cross-validation accuracy greater than 97.37% will not readily be achieved. An apparently reasonable choice of $(\gamma = 0.01, c = 100)$ was thus made and an SVM trained using these values. The resulting SVM used 313 of the training

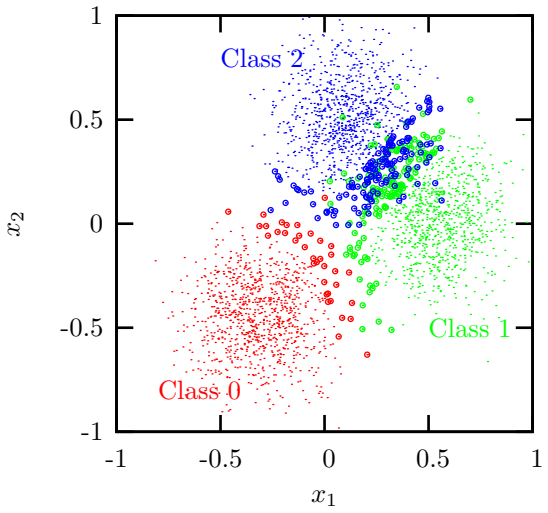


Figure 7.3: *Scaled training data. The 313 samples used as support vectors, by the model* ($\gamma = 0.01, c = 100$) *chosen via cross-validation on the data set, are shown ringed.*

Table 7.1: *Classification results for 3000 test points using LIBSVM trained via cross-validation. The overall classification accuracy is 96.8%.*

	Predicted class			
Class	0	1	2	$\%_{correct}$
0	**992**	3	5	99.2
1	2	**952**	46	95.2
2	3	36	**961**	96.1

samples as support vectors: 36 were from class 0, 142 from class 1, and 135 from class 2. The greater concentration of support vectors in classes 1 and 2 reflects the greater overlap between these two classes. The training data are shown in figure 7.3 with the points that were used as support vectors shown ringed. In this figure the training data can be seen to have been scaled into the interval $[-1, 1]$.

The resulting classification performance on 1000 test samples from each class (generated from the same distribution as the training data) is given in table 7.1. As expected from the data distributions, the greatest confusion is between classes 1 and 2.

7.2.3 Model selection by separate validation set

A validation data set, again comprising 1000 samples from each class, was generated in the same way as the previously used training and test data sets. The same grid search over parameters γ and c was conducted, only this time the measure of classification accuracy for performing model selection was calculated against this new validation data set. The contour plot of classification accuracy for the validation data set, using a coarse

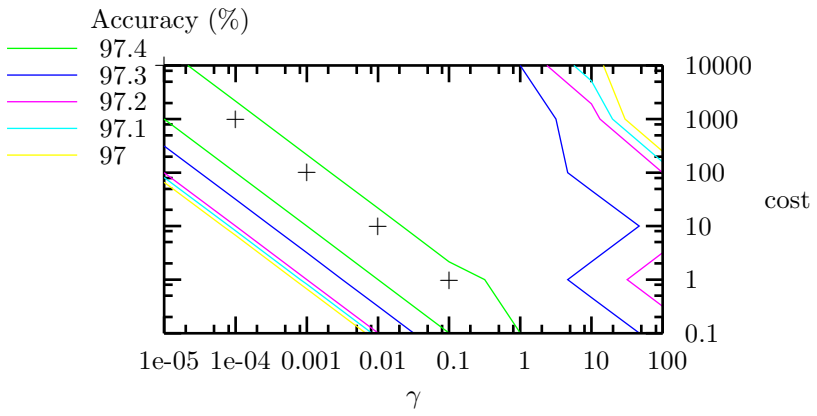


Figure 7.4: *Contour plot of classification accuracy on the validation data set using a relatively coarse grid. Even with the coarse grid, a ridge of peak values (97.43%, marked by '+') can be made out.*

grid, is shown in figure 7.4. Comparing the contour plot of figure 7.4, obtained using the validation data set, with that of figure 7.2a, obtained using cross-validation on the training data set, the surface obtained using a validation data set appears somewhat less complex, with the ridge of peak values readily evident.

Owing to the simpler topology of the validation-set surface, a further, finer, grid search was not conducted and the values ($\gamma = 0.1, c = 1$) were used for the next LIBSVM model. The resulting model used 605 support vectors[1]: 101 from class 0, 256 from class 1, and 248 from class 2. As with the previous model, significantly many more samples from classes 1 and 2 were taken as support vectors than from class 0 reflecting the greater overlap between classes 1 and 2. These support vectors along with the scaled training data are shown in figure 7.5.

The resulting classification performances on the test data are given in table 7.2. The results are nearly identical to those of table 7.1, with the overall classification performance being only 0.1 percentage points greater.

Rather than train on 90% of the training data and test on the remaining 10%, then repeat this ten times to obtain a single cross-validation estimate of classifier performance, the use of a validation data set allows training on 100% of the training set and testing on 100% of the validation set just once to yield an estimate of performance. As well as this yielding a computational speed advantage, the slight difference in vehicle configurations between training and validation data will mean that the chosen model parameters are less

[1]Nearly twice the number used by the previous model.

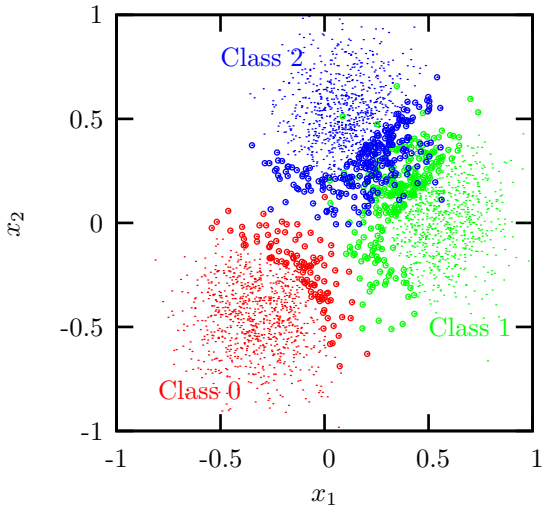


Figure 7.5: *Scaled training data. The 605 samples used as support vectors, by the model* $(\gamma = 0.1, c = 1)$ *chosen via classification performance on a separate validation set, are shown ringed.*

Table 7.2: *Classification results on the test data using LIBSVM training on a separate validation data set. The overall classification accuracy is 96.9%.*

	Predicted class			
Class	0	1	2	$\%_{correct}$
0	**994**	1	5	99.4
1	2	**951**	47	95.1
2	3	36	**961**	96.1

finely tuned to the configurations present in the training data. Furthermore, the smoother surface obtained by using a separate validation data set simplifies parameter selection.

7.2.4 On the number of support vectors and the resulting decision boundary complexity

The parameters selected by cross-validation and those selected using independent validation data gave rise to classifiers with very similar performances on the test data. However, the model selected using the validation data used twice as many support vectors. Thus, using the same training data, it is possible to obtain two SVMs that demonstrate very similar classification performances using very different numbers of support vectors.

The previous contour plots for model selection considered only the estimated classification performance. By also displaying the number of support vectors used, SVMs can be selected that use the fewest support vectors for a given level of classification performance. The model selection stage was conducted again, but this time the number of support vectors for each SVM was recorded in addition to the cross-validation accuracy. The resultant graph is shown in figure 7.6. The model previously chosen by cross-validation was ($\gamma = 0.01, c = 100$) and this is still evident as a peak value in figure 7.6, despite a widening of the range over which (γ, c) were explored. This figure illustrates the topology of the (coloured) surface representing the number of support vectors for each SVM.

The ridge of peak values of validation-set performance was slightly offset from the corresponding ridge obtained from cross-validation, which explains the slightly different parameters chosen for the two respective SVMs. This nudged the chosen SVM in the direction of a steep rise in the number of support vectors and explains the large difference in the numbers of support vectors used in the two models. Furthermore, the ridge of peak classification performance in figure 7.6 runs parallel to this sharp rise in the number of support vectors, so there is no unique set of parameters (γ, c) that combines peak estimated performance with a smaller number of support vectors. The ridge of peak values relates to classification accuracies that exceed the 97.3% contour; the 97.2% contour is much wider and suggests that performance will not be overly sensitive to small changes in the selected model parameters.

To contrast the performance of the previous model chosen by cross-validation, a new model was selected using the opposite criterion: minimization of the number of support vectors. Thus, the parameters ($\gamma = 0.001, c = 10^{12}$) were selected, reducing the number of support vectors by more than half whilst accepting a drop in cross-validation accuracy of less than 0.2%. The resultant support vectors used to construct this SVM are shown in figure 7.7. There are now very few support vectors used to describe the boundary between Class 0 and the other two classes. This class is most readily separable from the other

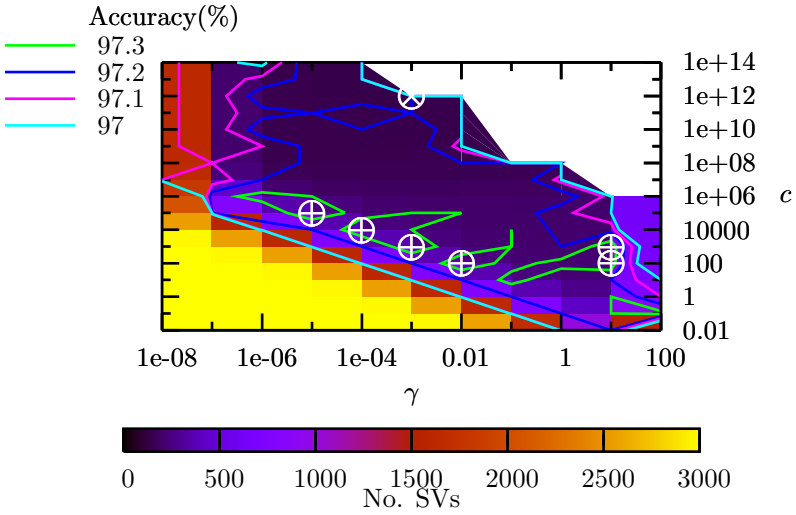


Figure 7.6: *Cross-validation accuracy contours and number of support vectors over an extended parameter search. The peak cross-validation accuracy is 97.37%, marked with* $\oplus$*, and the minimum number of support vectors is 138, marked with* $\otimes$*. The uncoloured region reflects runs terminated due to elapsed time.*

Table 7.3: *Classification results on the test data using LIBSVM training by cross-validation and choosing the model yielding the smallest set of support vectors. The overall classification accuracy is 96.8%.*

	Predicted class			
Class	0	1	2	$\%_{correct}$
0	**989**	3	8	98.9
1	2	**952**	46	95.2
2	3	33	**964**	96.4

two and so, on the face of it, this would seem to be an appealing result as it appears to exemplify the often touted sparsity of the SVM solution.

Accepting that the final arbiter of a classifier's performance is its performance on unseen test data, the classification results, on the test data set, of this new model are presented in table 7.3. The overall classification performance of 96.8%, identical to that in table 7.1, offers no evidence either way that the reduction in model complexity has been either detrimental or favourable to classifier generalization. All that can be said is that, on these specific test data, overall performance has not been decreased for the significant reduction in the number of support vectors used.

Ultimately, because we know the distributions giving rise to the three classes — specifically Gaussian with common variance and each class having equal *a priori* probability — we can readily compare the empirically-determined decision boundaries of the SVMs with the theoretically-optimal decision boundaries. The optimal decision boundary between

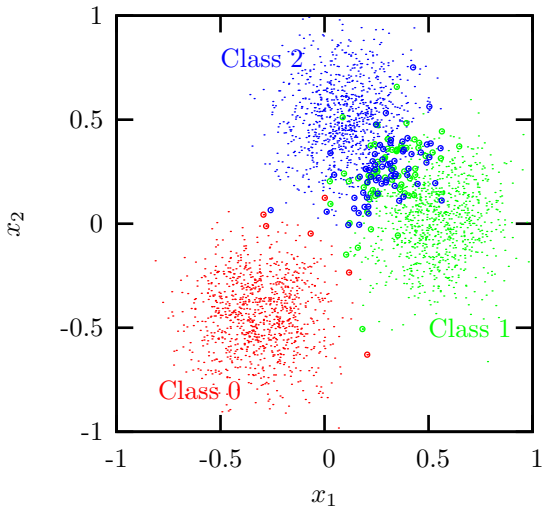


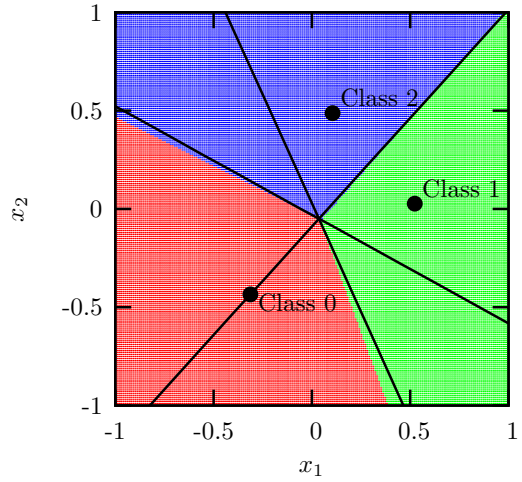
Figure 7.7: *Scaled training data. The 138 samples used as support vectors, by the model* $(\gamma = 0.001, c = 10^{12})$ *chosen to use the fewest support vectors, are shown ringed.*

two such distributions is normal to the line between the two means and intersects that line at its midpoint. The optimal decision boundaries for multiple such classes are simply the resultant piecewise linear decision boundaries of each two-class problem.

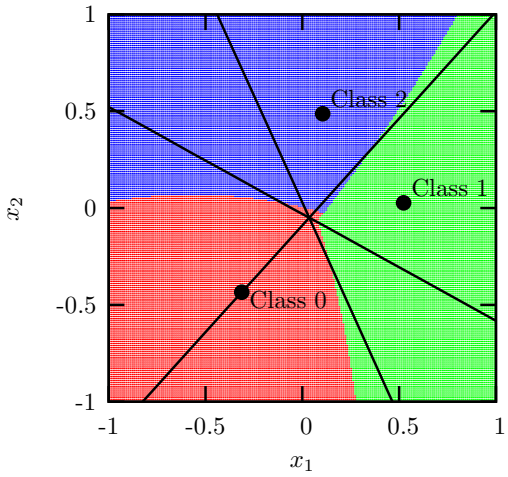
The decision boundaries for the original model obtained via cross-validation are shown in figure 7.8a. The empirical decision boundaries are clearly close to optimal, with the 1/2 boundary being closest. The attempt to obtain a more sparse solution, on the other hand, resulted in the decision boundaries shown in figure 7.8b. The empirically determined boundaries have deviated away from the optimal boundaries. The 0/2 boundary, in particular, has become more complex with noticeable curvature. Thus, despite the two SVMs exhibiting equal overall performance on the test data, the model chosen to yield the highest cross-validation accuracy is unarguably closer to demonstrating the property of best generalization than is the model chosen to yield the fewest support vectors.

Finally, we consider the decision boundaries as determined by the model chosen on the basis of peak classification performance on a separate validation data set. Recall that the use of this validation data set facilitated faster model selection because of negating the requirement to perform cross-validation, at the expense of the chosen model using 605 support vectors. The empirically determined decision boundaries are shown related to the optimal boundaries in figure 7.8c. The empirical decision boundaries appear to be as simple (straight) and as close to the optimal as were those of the 313 support vector SVM.

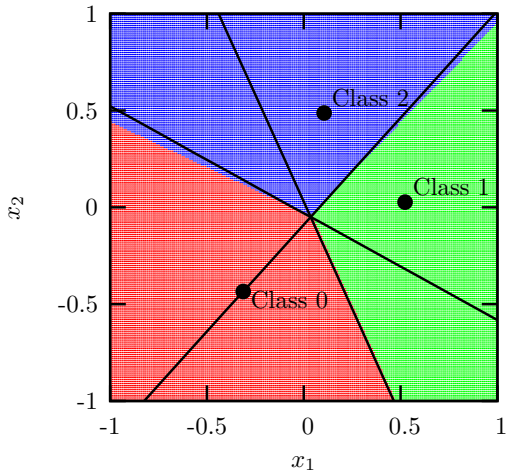
Having established that the support vectors used in the SVM models chosen to offer best classification performance did indeed define simple decision boundaries that lay very



(a) *313 support vectors*



(b) *138 support vectors*



(c) *605 support vectors*

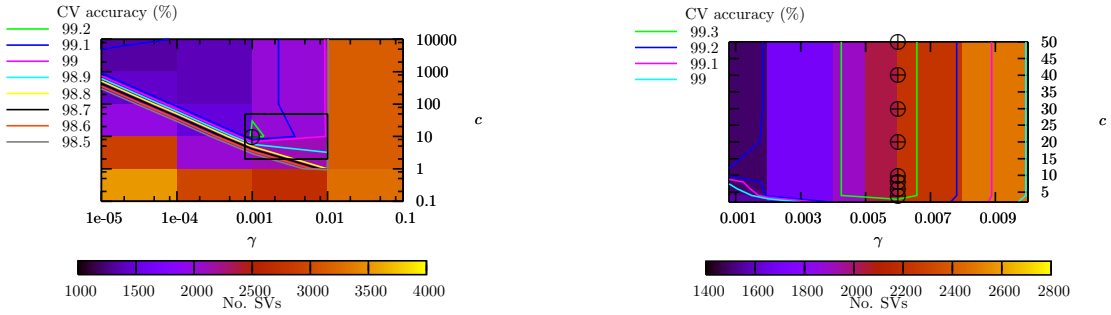
Figure 7.8: *Color coded illustration-problem decision-regions empirically determined by different SVMs that used different numbers of support vectors. The mean of each class is labelled, and the theoretically-optimal decision boundary for each constituent two-class problem is marked by a solid black line.*

close to the best possible decision boundaries, it would be satisfying to be able to further provide a solid basis for the number of support vectors used. It is known that SVMs often scale poorly with the size of the training set [95]. For finite kernels, such as polynomial, the dimension of the feature space provides an upper bound on the number of support vectors [91], but this bound does not exist for infinite kernels such as the Gaussian. Furthermore, it has been established for the Gaussian RBF kernel that the fraction of support vectors tends to twice the Bayes risk for the L1-SVM [107, 108], i.e. the number of support vectors scales linearly with the training set size.

The fraction of support vectors is an upper bound of the LOO estimator of the generalization error [10]. Specifically, the number of LOO errors does not exceed the number of essential support vectors [119, p.418]. The corollary of this is that the number of essential support vectors is greater than or equal to the number of LOO errors. Note that the essential support vectors comprise the intersection of all possible sets of support vectors [119, p.413] and would appear to be mutually linearly independent [35]. Given that SVMs generally produce solutions with a greater number of support vectors than are strictly necessary [35], we can expect to obtain, in practice, more support vectors than the 'bare minimum' number of essential support vectors that would be indicated by the LOO estimation of the generalization error. Now, we have not performed LOO estimation on the illustration-problem training data, but we do know that the decision boundary obtained by the 313-SVM (figure 7.8a) is close to the Bayesian optimal boundary and provides a good estimate of the true generalization error. We will now use this estimate to compare the number of support vectors used with the number predicted, or bounded, from theory.

The 313-SVM producing the decision boundaries in figure 7.8a is built from the support vectors marked in figure 7.3. There are 142 support vectors in Class 1 and 135 support vectors in Class 2, a total of around 280. By eye we estimate that around 20 support vectors in each class are actually contributing to the decision boundaries between these two classes and Class 0. This leaves 240 support vectors across the two classes supporting the mutual decision boundary. The error rate between classes 1 and 2, for this SVM, is approximately 4% (table 7.1). Taking this to be a reasonable estimate of the Bayes risk, by Steinwart [108] we would expect around 8% of the 2000 samples to be used as support vectors (giving 160). In light of this, admittedly informal estimate, our use of ≈ 280 support vectors appears reasonable. We could not expect fewer than 4% of 2000 samples to be used as (essential) support vectors (giving 80) from Vapnik's bound on the number of essential support vectors [119, p.418]

Attempts to reduce the number of support vectors have included the selective removal of examples from the training set such that separable distributions of training examples



(a) *Coarse grid.* (b) *Fine grid.*

Figure 7.9: *Contour plot of cross-validation accuracy obtained by LIBSVM with ISAR data. The coloured surface represents the number of support vectors used when the SVM is trained on the full training set. The contours show the levels of cross-validation accuracy. The locations of the peak values of cross-validation accuracy,* 99.24% *for the coarse grid and* 99.35% *for the fine grid, are marked by* $\oplus$ *and the subregion of the coarse grid selected for the finer grid search is indicated by the black box.*

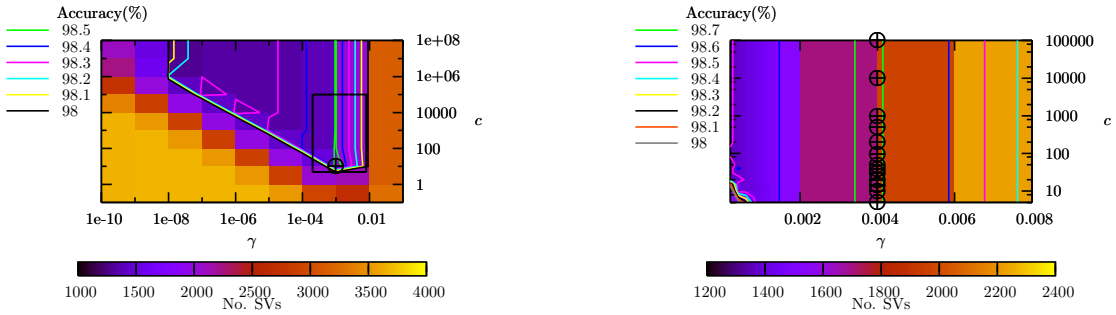
remain without modifying the location of the decision boundary [2]. The derivation of a separable training set reduces the Bayes risk to zero and the number of support vectors scales less than linearly with the training set size. Another approach has been to identify and remove support vectors that are linearly dependent in feature space [35]. Clearly the removal of such support vectors will leave the decision boundary unchanged.

7.3 Inverse synthetic aperture radar data

7.3.1 Model selection by cross-validation

Performing 10-fold cross-validation on the training data set that comprised three data sets (one from each of the three classes APC, MBT, and ADU) realized better than 99% accuracy. The contour plot of this cross-validation performance, which also indicates the number of support vectors when trained on the full training set, is shown in figure 7.9a. The number of support vectors rises sharply outside of the region of 'good' parameters. A subregion around the peak value marked by $\oplus$ was selected for closer examination using a finer grid of (γ, c) and the resultant plot for cross-validation accuracy is shown in figure 7.9b. In figure 7.9b the peak performance is insensitive to the value of c over the range examined. The number of support vectors exhibits similar insensitivity.

Based on figure 7.9b, the parameters chosen for the final SVM were $(\gamma = 0.006, c = 10)$, with considerable freedom apparently available in the choice of c. This SVM used 2103 support vectors, equating to 57.2% of the training set, with 673 from the MBT class, 609



(a) *Coarse grid.* (b) *Fine grid.*

Figure 7.10: *Contour plot of classification accuracy on the small validation data set for SVMs trained on the small training set. The coloured surface represents the number of support vectors used. The peak value of classification accuracy, marked by* $\oplus$*, was* 98.59% *for the coarse grid and* 98.71% *for the fine grid. The black box outlines the region selected for the finer grid search.*

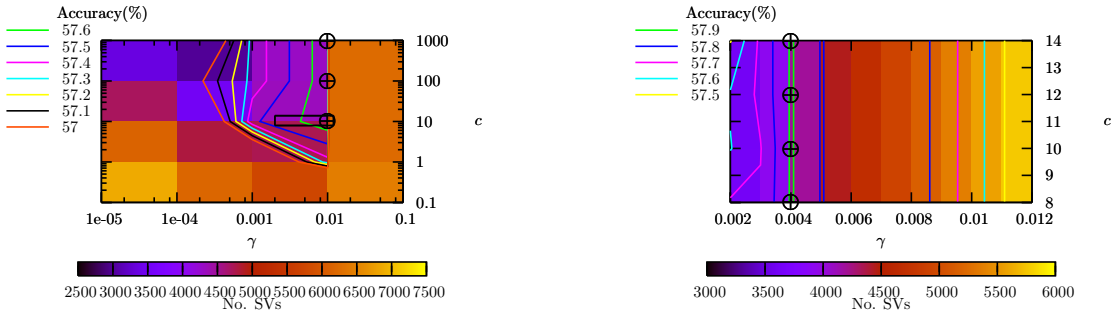
from the APC class, and 821 from the ADU class. This SVM was then used to classify the various partitions (Groups 1–5) of the test data. The results are presented and discussed in section 7.4.

7.3.2 Model selection by separate validation set

Small training set and small validation set

Performing model selection based on the classification accuracy on a separate validation data set offers the potential of improved generalization performance on unseen data and reduced training (model selection) time. The coarse grid search, training on the small training set and calculating the classifier's performance on the small validation set, is shown in figure 7.10a. The peak value for classification performance obtained in the coarse grid search was 98.59%. The location of this value is marked by $\oplus$ in figure 7.10a and the subregion selected for a finer grid search is identified by the black box. The model selection results for this finer grid search are shown in figure 7.10b. The peak performance on the fine grid was 98.71%, with insensitivity to the value of c.

The model selected from figure 7.10b was ($\gamma = 0.004, c = 10$). This yielded an SVM that used 1833 support vectors (49.8% of the training set), with 609 belonging to the MBT class, 524 to the APC class, and 700 to the ADU class. Comparing this model with that previously obtained, via cross-validation, the estimate of the best model has been moved in the direction of smaller γ. This is also in the direction of decreasing number of support vectors. Recall that the model is being trained on exactly the same training data in both scenarios, it is only the data set upon which the SVM's performance is being



(a) *Coarse grid.* (b) *Fine grid.*

Figure 7.11: *Contour plot of classification accuracy on the extended validation data set for SVMs trained on the extended training set. The coloured surface represents the number of support vectors used. The peak value of classification accuracy, marked by* $\oplus$*, was* 57.67% *for the coarse grid and* 57.91% *for the fine grid. The black box outlines the region selected for the finer grid search.*

assessed that is different. Thus, if the two model selection methods had suggested the same parameters (γ, c) then the two SVMs would have been identical. The SVM trained using these parameters was then used to classify the five groups of test data. The results are presented and discussed in section 7.4.

Extended training set and extended validation set

Finally, LIBSVM was trained on the extended training set with model selection being based on classification performance obtained on the extended validation set. Thus this scenario trained the algorithm on a wider variety of data as well as choosing parameters based on performance against a wider variety of data. The coarse grid search contour plot is shown in figure 7.11a. The peak performance was 57.67%, marked by $\oplus$, and the region selected for a finer grid search is marked by the black box. The much lower classification performance reflects the presence in the new validation data set of vehicles not represented during training.

As both training and validation were performed on the larger data sets, computation time increased significantly for these experiments. Hence, the size of the region over which the finer grid search was conducted was kept as small as possible. The fine grid search results are shown in figure 7.11b, where the peak performance of 57.91% is marked by $\oplus$. Thus the parameters selected for the final SVM model were $(\gamma = 0.004, c = 10)$. This resulted in an SVM using 4060 support vectors (57.9% of the training set), with 1306 from the MBT class, 1318 from the APC class, and 1436 from the ADU class. The results are presented and discussed in section 7.4.

7.4 Classification results on the radar data

The LIBSVM classification results for the Group 1 and Group 2 test data are presented in figures 7.12a and 7.12b, respectively. All of the vehicles in these test groups were represented during classifier training. For the classifier trained using the small training set, with model selection performed either via cross-validation (red) or by separate validation set (green), error rates within each vehicle label are consistent. With the exception of MBT1-2 (in figure 7.12b) all error rates, for training using the small training set, are very low ($< 5\%$), sometimes vanishingly so. With the inclusion of extra vehicles in the training set (blue) error rates rise sharply. The single exception is again MBT1-2 in figure 7.12b, which shows a halving of the error rate.

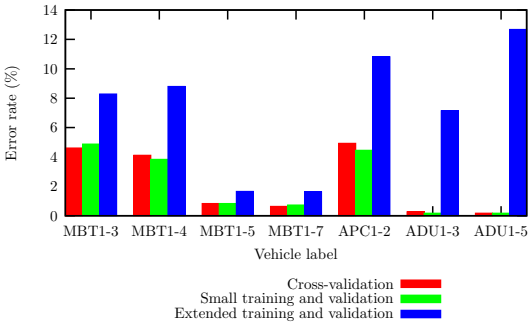
The LIBSVM classification results for Groups 3 and 4 are shown in figures 7.12c–7.12g, respectively. None of the vehicles in these groups were represented when the classifier was trained using the small training set. A few of the vehicles were represented when the extended training set was used. With the exception of the results for ADU3 all error rates are high when training used only the small training set. It turns out that ADU3 is a ZSU-23-4, the same vehicle type as ADU1 used in the small training set. As previously noted, the use of a separate validation data set to choose model parameters yielded a classifier as good as when using cross-validation. Vehicles in these two groups which were added with the extended training set are MBT2 and APC2. These vehicles appear in figures 7.12c and 7.12e. Error rates against these vehicles drop markedly with the vehicles' inclusion during training. Error rates for ADU2 and ADU3 are very high with the extended training data, and low using the small training data. The classification performances for the other vehicles remains very poor.

The vehicles in Group 5 comprise only MBT3 and MBT5. The LIBSVM results for this group are shown in figure 7.12h. All error rates are very high.

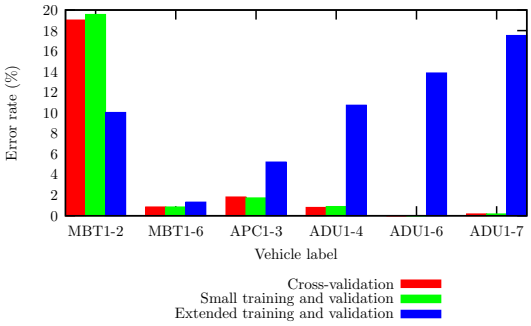
7.5 Summary and conclusions

The three-class, two-dimensional illustration problem demonstrated that the use of a separate validation data set for model selection was as good as cross-validation on the training set. Furthermore, using a separate validation data set provided a large speed boost. The decision boundaries were shown to be very close to the theoretical optimum and the number of support vectors used was shown to be consistent with theoretical expectation.

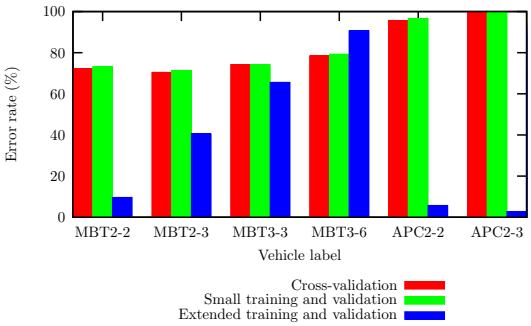
Using the ISAR data, the use of the separate validation set for model selection was again shown to produce classification results that were as good as with cross-validation. This gave confidence that cross-validation could be avoided for performing model selection



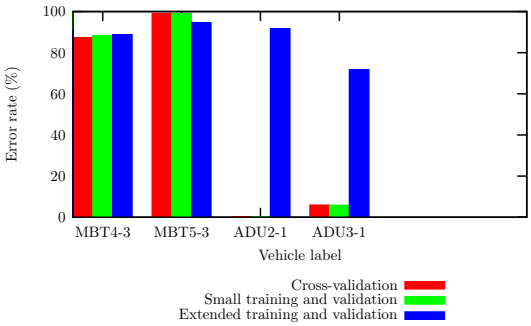
(a) *Group 1*



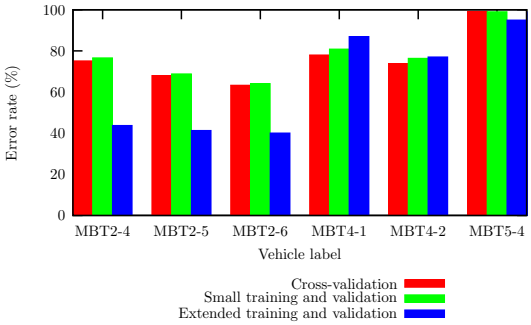
(b) *Group 2*



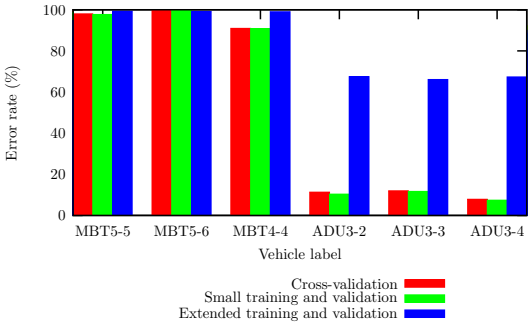
(c) *Group 3*



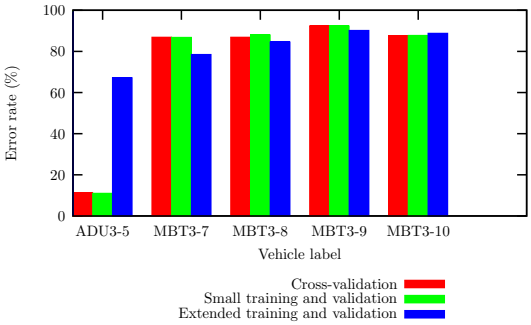
(d) *Group 3*



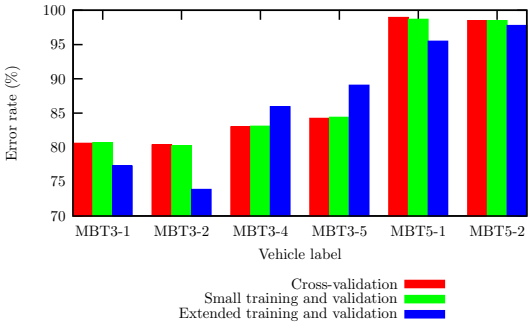
(e) *Group 4*



(f) *Group 4*



(g) *Group 4*



(h) *Group 5*

Figure 7.12: *LIBSVM error rates.*

on the extended training set, important for keeping computation time to a minimum.

Using only the small training set, classification performance was very good on vehicles represented during training, but was very poor on vehicles not represented during training. There were a small number of exceptions: MBT1-2 had an anomolously high error rate for Groups 1 and 2; ADU3 had an anomolously low error rate for Groups 3 and 4, although this turned out to be the same vehicle type represented during training by ADU1; the error rate for ADU2 was vanishingly small. This was difficult to explain.

The addition of the extra training data resulted in substantial increases in error rates in Groups 1 and 2, with simultaneous substantial falls in error rates for the vehicles in Groups 3 and 4 that were newly included during training. The latter, however, did not drop anywhere near as low as the error rates seen in Groups 1 and 2 when using the small training set.

Thus, for narrowly defined problems, essentially where the class label is synonymous with a single vehicle type, the SVM performance was excellent. In this light, rather than viewing the very low error rate on ADU2 as a successful assignment to the ADU class, it can be viewed as highlighting the risk of false positives as it was 'wrongly' assigned to the 'ADU1' class (ADU1 being the vehicle type represented in training). When the classification problem was widened through the inclusion of more vehicle types within generic classes, classification performances against the new vehicle types were improved, but were not as good as performances on the original training vehicles when they were the sole representatives of their class. Furthermore, the performances on these original training vehicles when the 'definition' of a class was widened were not as good as when they were they sole representatives of their classes. These facts suggest that it is probably much better to have an n-class problem for n vehicle types than to attempt to formulate a smaller number of generic classes.

Chapter 8

Comparison of techniques

8.1 Introduction

This chapter draws together a comparison of the techniques described in the preceding chapters. The results are drawn together into three categories: classification performed using only the small training group comprising one vehicle per class, presented in section 8.2; classification performed using the extended training group comprising two vehicles per class, presented in section 8.3; and classification performed involving the Box-Cox transformation, presented in section 8.4.

Not all techniques from previous chapters are relevant in all categories. Most of the previously applied techniques are relevant to the small-training-group category, with the notable exception of anything involving the Box-Cox transformation, which is considered separately. Techniques compared within the extended-training-set category have been restricted to those that make use of class labels during training. Only results using the small training group are included within the Box-Cox transformation category, as the main purpose of those experiments was to assess the effect of the Box-Cox transformation rather than additional training data.

All subfigures within any given figure of results for the five groups are plotted using the same y-axis range. This facilitates comparison between data sets within each group.

Finally, a summary and conclusions of this chapter is given in section 8.5.

8.2 Small training group

Most of the previously applied techniques are relevant to this category. The results compared are:

Full dim The class exemplars were provided by the small training group. Classification was performed in the full-dimensional measurement-space. These results were

previously quoted in section 5.4, in figure 5.14.

PCA The class exemplars were provided by the small training group. Classification was performed in the 30-dimensional PCA feature-space determined by training on the small training group. These results were previously quoted in section 5.4, in figure 5.14.

LDA The class exemplars were provided by the small training group. Classification was performed in the LDA feature-space determined by training on the small training group. These results were previously quoted in section 5.4, in figure 5.14.

SVM The SVM classifier was trained on the small training group. Model parameters were selected based on classification performance against the small validation group. These results were previously quoted in section 7.4, in figure 7.12.

RBF(U) The RBFN was trained using the small training group and the unsupervised loss function. Model selection was based on the stress calculated against the small validation group. The class exemplars were provided by the small training group. Classification was performed in 20-dimensional RBF feature space. These results were previously quoted in section 6.2.2, in figure 6.6.

RBF(S) The RBFN was trained using the small training group and the supervised loss function ($\rho = 0$). Model selection was based on the stress calculated against the small validation group. The class exemplars were provided by the small training group. Classification was performed in 20-dimensional RBF feature space. These results were previously quoted in section 6.3.2, in figure 6.12.

RBF(R) The RBFN was trained using the rescaled inter-class distances and unsupervised loss function on the small training group. Model selection was based on the stress calculated against the small validation group. The class exemplars were provided by the small training group. Classification was performed in the 20-dimensional RBF feature space. These results were previously quoted in section 6.4.2, in figure 6.18.

RBFN The RBFN was trained to perform classification rather than feature extraction. The network was trained in a directly comparable manner to the SVM classifier: it was trained on the small training set and final model parameters were selected based on classification performance on the small validation set. These results have not previously been quoted and are given only for a limited subset of the test data to demonstrate the expected similarity between the SVM and RBFN classifiers.

All figures within this section use the following colour scheme. Results obtained with no shifting of the input images are coloured red and with a shift of ±1 pixels are coloured green. Shifting was not applicable to the SVM and so those results are only available without shifting and so are coloured red.

The questions being posed are, given a single representative vehicle for each class:

- How well did each technique extract features that generalized across target variability?
- How did classification in a reduced dimensional space compare to classification using the full dimensional space of the original images?

8.2.1 Group 1

The summary of results for the data sets comprising Group 1 is shown in figure 8.1.

The SVM classifier was clearly the best. Even though shifting was not implemented for this classifier, it performed at least as well as the 1-NNR in full-dimensional measurement-space with shifting. Usually it did far better. MBT1-3 (figure 8.1a) and MBT1-4 (figure 8.1b) boasted an error rate of one half to one third of the best of the rest. Thus the SVM demonstrated increased robustness to the toolboxes being opened, relative to any of the other techniques. For MBT1-5 (figure 8.1c) and MBT1-7 (figure 8.1d) it rivaled the error rate achieved using full-dimensional data, which was very low anyway. SVM error rates against ADU1-3 (figure 8.1f) and ADU1-5 (figure 8.1g) were vanishingly small, indicating a robustness to rotation of the ZSU turret.

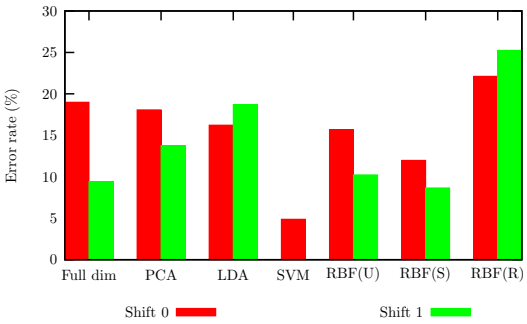
For a majority of the data sets — MBT1-5 (figure 8.1c), MBT1-7 (figure 8.1d), APC1-2 (figure 8.1e), and ADU1-5 (figure 8.1g) — classification in LDA and RBF(R) feature-space was far worse than for other techniques. It is interesting that these two techniques were a linear and a nonlinear way, respectively, of increasing the separability of the classes (subject to some constraint).

Of the RBF techniques, the supervised loss function, RBF(S), with $\rho = 0$ always performed best. Thus, for the Group 1 data, training the nonlinear RBFN projection to capture structure that reflects extant inter-class separability within the training data, generalized better than attempting to increase the separability of the classes.

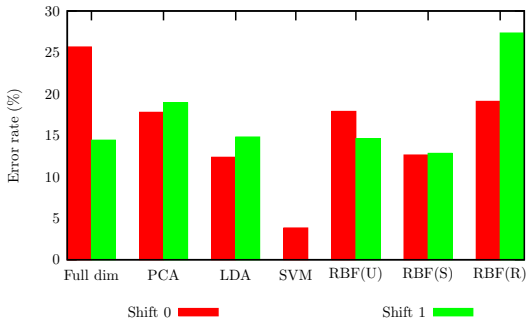
8.2.2 Group 2

The summary of results for data sets comprising Group 2 is shown in figure 8.2.

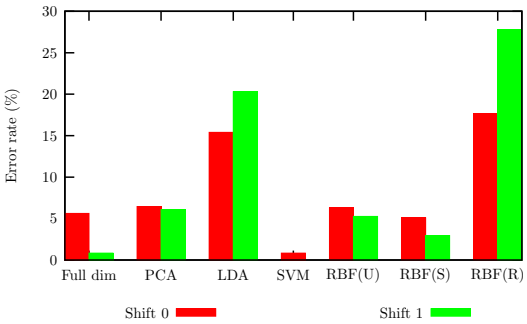
As with Group 1, the SVM classifier is the best overall. Sometimes by a slim margin, often by a considerable margin. For MBT1-6 (figure 8.2b) and ADU1-4 (figure 8.2d) it



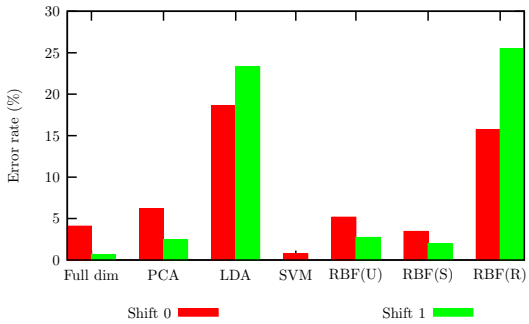
(a) *MBT1-3*



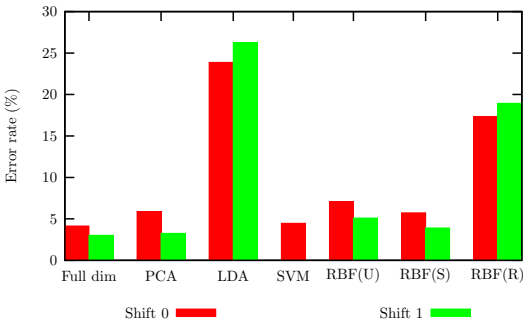
(b) *MBT1-4*



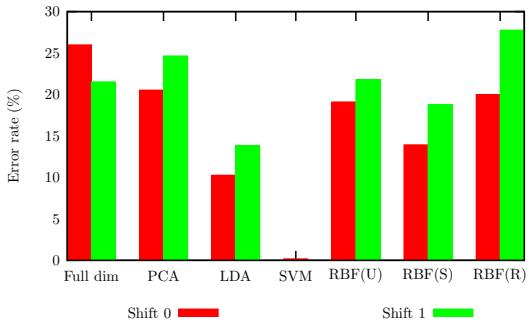
(c) *MBT1-5*



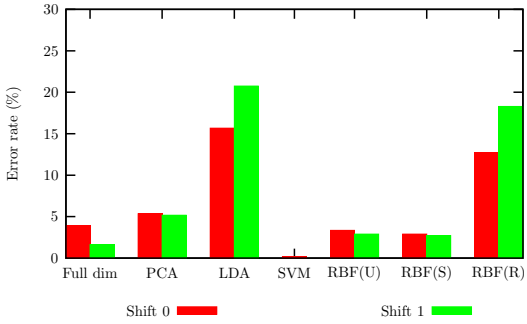
(d) *MBT1-7*



(e) *APC1-2*

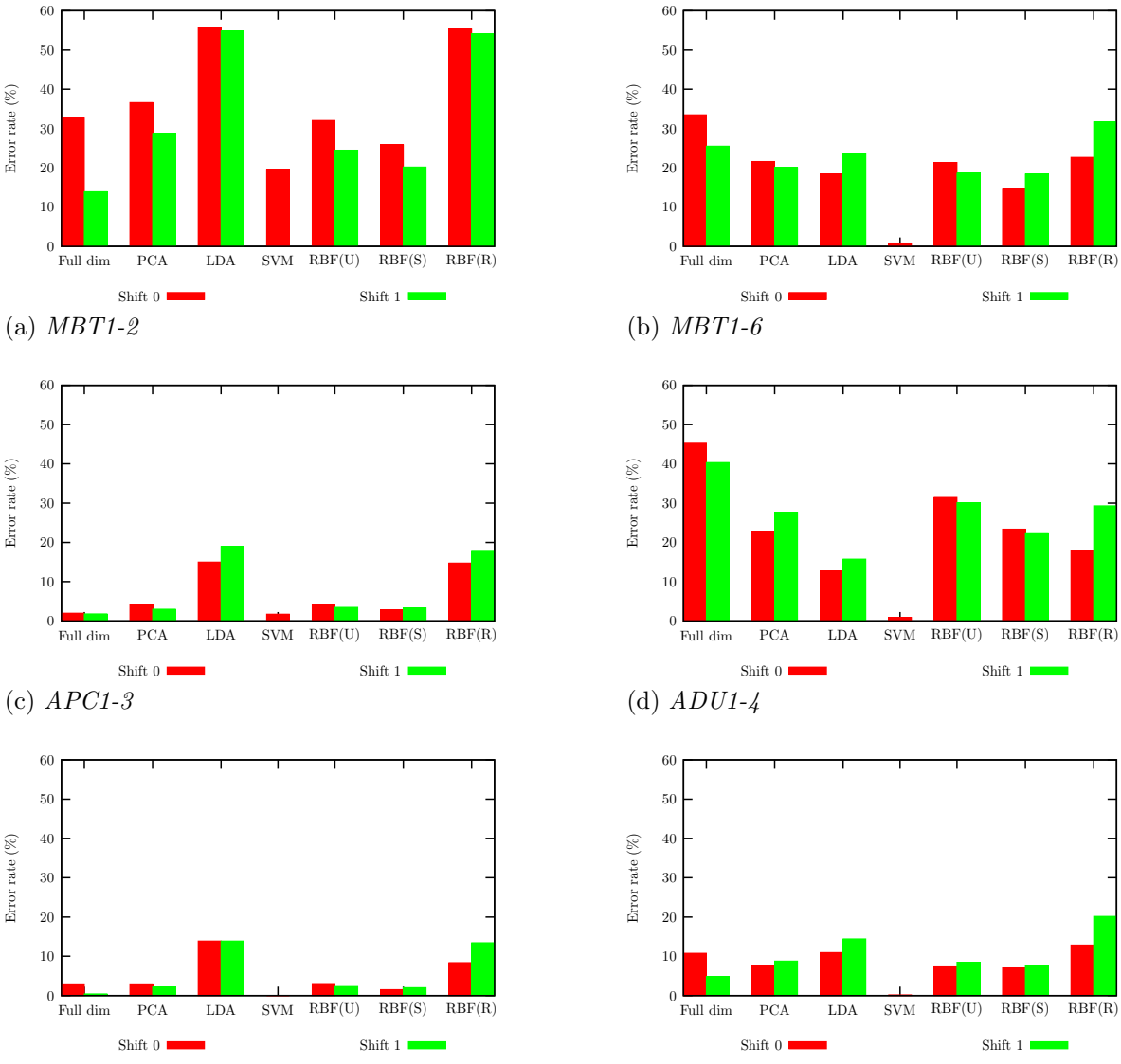


(f) *ADU1-3*



(g) *ADU1-5*

Figure 8.1: *Group 1, comparison of the small-training-group results. RBF(U) denotes RBF feature space obtained using the unsupervised loss function and projecting to 20 dimensions. RBF(S) denotes RBF feature space obtained using the supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using the rescaled inter-class distances and projecting to 20 dimensions.*



(a) *MBT1-2*

(b) *MBT1-6*

(c) *APC1-3*

(d) *ADU1-4*

(e) *ADU1-6*

(f) *ADU1-7*

Figure 8.2: *Group 2, comparison of the small-training-group results. RBF(U) denotes RBF feature space obtained using the unsupervised loss function and projecting to 20 dimensions. RBF(S) denotes RBF feature space obtained using the supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using the rescaled inter-class distances and projecting to 20 dimensions.*

particularly casts the other methods in a poor light. MBT1-6 is the training vehicle with camouflage netting draped over it, which modifies the radar return somewhat over the entire vehicle extent. ADU1-4 is the training ZSU-23-4 with the turret rotated through 90°. The SVM thus demonstrated particular robustness, relative to the other techniques, to these significant deviations from training conditions. The error rates for ADU1-6 (figure 8.2e) and ADU1-7 (figure 8.2f) were vanishingly small for the SVM. Only for MBT1-2 (figure 8.2a) was the SVM slightly improved upon by the full-dimensional results with shifting. Even for this data set, the SVM result can arguably be viewed favourably given that it was achieved without any shifting of the input image. This indicates a robustness to the precise position of the target within the image not possessed by any of the other methods.

In half the data sets — namely MBT1-2 (figure 8.2a), APC1-3 (figure 8.2c), and ADU1-6 (figure 8.2e) — the LDA and RBF(R) feature-spaces once again yielded error rates similar to one another and that stood well above the error rates demonstrated by the other techniques. Thus, again, attempting to obtain features that increase the separability between the training classes seemed to result in worse generalization to variations in target configuration than features that sought to capture extant separability between those same training classes.

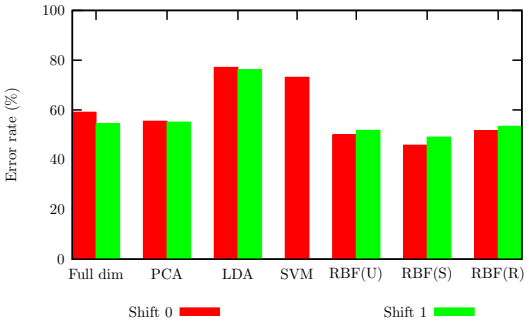
In most of the data sets within Group 2, with the exception of ADU1-4 (figure 8.2d), the RBF(R) was the worst of the RBF results. The margin was often considerable. The RBF(S) was generally the best of the RBF results, although its margin relative to the RBF(U) was generally small. The one exception to the latter was ADU1-4 where RBF(S) with no shift gave a higher error rate than RBF(R) with no shift. However, RBF(S) both with no shift and with shift of ± 1 pixels gave lower error rates than RBF(R) with shift of ± 1.

8.2.3 Group 3

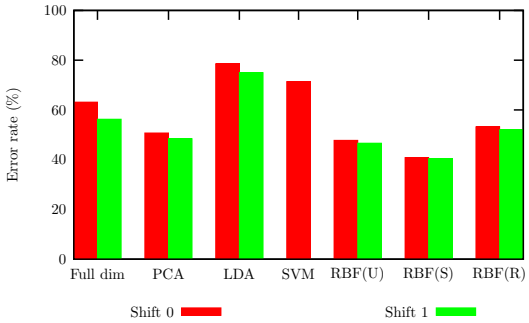
The summaries of results for data sets comprising Group 3 are shown in figure 8.3.

The error rates for MBT2, figures 8.3a and 8.3b, are generally better than would be obtained by random chance. The exceptions are the results for LDA and SVM. The results for APC2, figures 8.3e and 8.3f, are only better than random for classification performed using the full-dimension images. ADU2-1, in figure 8.3i, and ADU3-1, in figure 8.3j, are generally just better than random (usually aided by shifting), but the error rates from LDA and SVM are much better than random.

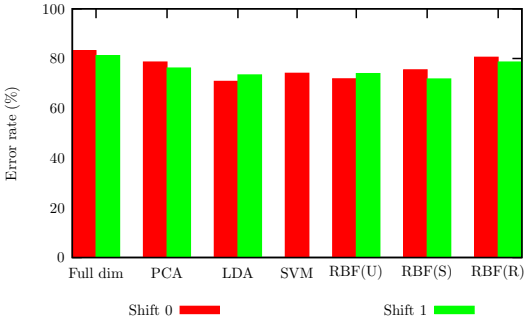
Recalling that ADU3-1 is the same vehicle type, and very similar configuration, as ADU1 used in training, it seems that both LDA and SVM are robust to whatever difference is present in the ADU3 data that caused the performance obtained by the other techniques



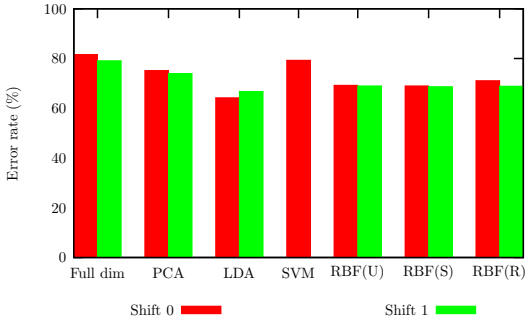
(a) *MBT2-2*



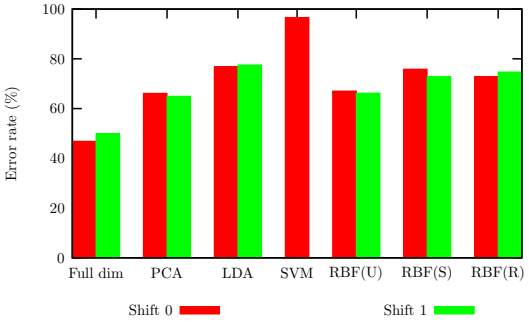
(b) *MBT2-3*



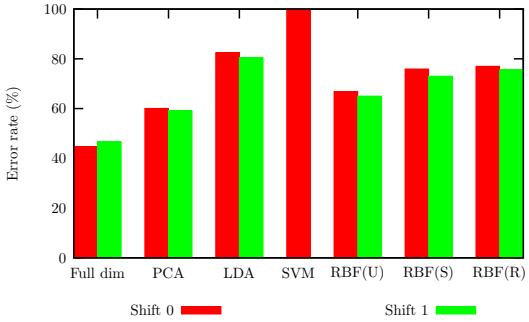
(c) *MBT3-3*



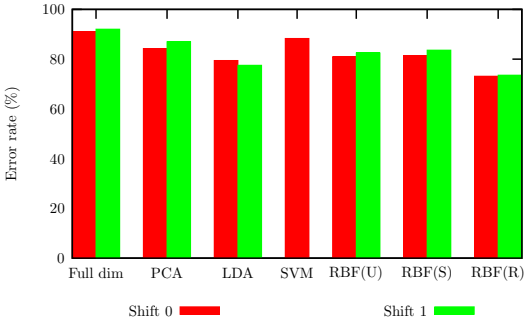
(d) *MBT3-6*



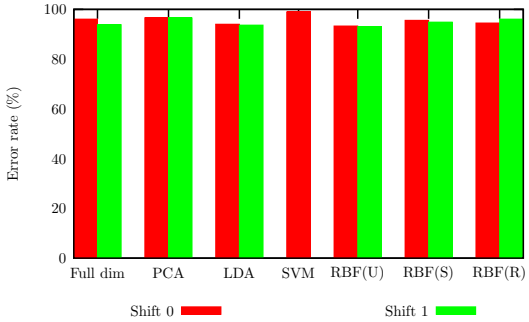
(e) *APC2-2*



(f) *APC2-3*

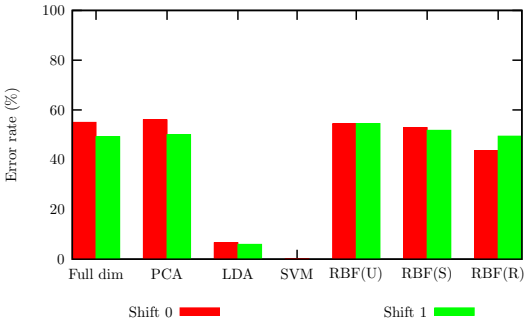


(g) *MBT4-3*

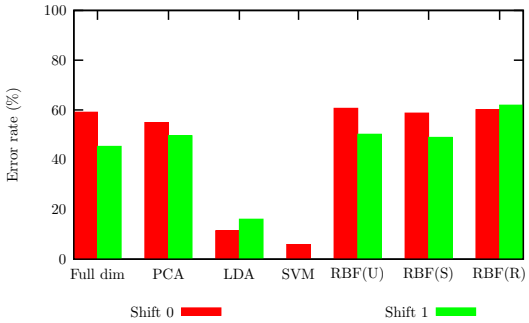


(h) *MBT5-3*

Figure 8.3: *Group 3, comparison of the small-training-group results. RBF(U) denotes RBF feature space obtained using the unsupervised loss function and projecting to 20 dimensions. RBF(S) denotes RBF feature space obtained using the supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using the rescaled inter-class distances and projecting to 20 dimensions.*



(i) *ADU2-1*



(j) *ADU3-1*

Figure 8.3: *Group 3, comparison of the small-training-group results. RBF(U) denotes RBF feature space obtained using the unsupervised loss function and projecting to 20 dimensions. RBF(S) denotes RBF feature space obtained using the supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using the rescaled inter-class distances and projecting to 20 dimensions (cont.)*

to be poor. A visual comparison of ISAR images from the two vehicles does suggest a reason for the different classification performances between them. Sample averaged images for each, at the same orientation angle, are shown in figure 8.4. Despite there being many similarities in the features between the two, including regions of bright radar returns, relative to the provided reference mark (+) the two vehicles are clearly offset from one another within the image frame. Both images are after the application of the automatic recentering algorithm.

8.2.4 Group 4

The summary of results for data sets comprising Group 4 is shown in figure 8.5.

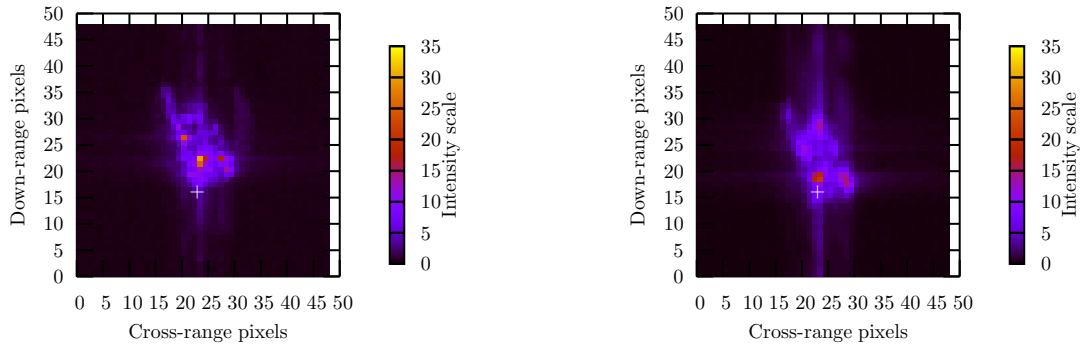
The error rates from most techniques on MBT2, in figures 8.5a, 8.5b, and 8.5c, are generally better than random. The exceptions are the error rates from LDA and SVM, which tend to be at, or above, the level of random chance.

The results for ADU3, in figures 8.5j–8.5m, are generally worse than random with the exception of LDA and SVM. These two techniques produce performances that are very much improved from the level of random chance. Thus, LDA, and especially SVM, seemed most capable of 'realizing' that ADU3 is the same class as ADU1.

8.2.5 Group 5

The summary of results for data sets comprising Group 5 is shown in figure 8.6.

There are two vehicle types, MBT3 and MBT5, present in this group. The vehicles were imaged at varying RDA. The error rates from all techniques applied to these data sets



(a) *ADU1-1* (b) *ADU3-1*

Figure 8.4: *Comparison of ADU1-1 and ADU3-1 averaged images. The same reference point within the image frame is marked in both with a '+'.*

are approximately at the level of, or worse than, random chance. There is no discernible change in performance with RDA.

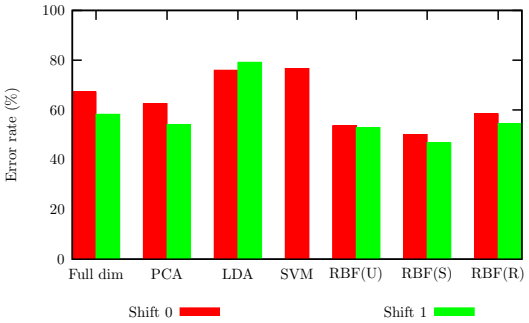
8.2.6 Radial basis function network trained as a classifier

The SVM is an optimized classifier that stands alone amongst the other methods, which are optimized in some sense for feature extraction. It is included because of its close connection to the radial basis function network as well as its current status as a well-regarded classifier. As such it provides an indicative level of achievable classification performance on the data.

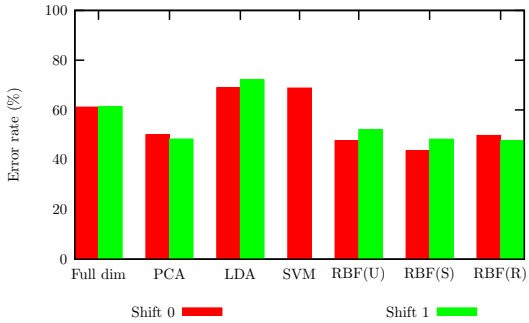
The classification results for Group 1 are repeated in figure 8.7 including the classification results from the RBFN classifier. As expected, the classification performances of the SVM and RBFN are similar.

8.3 Extended training group

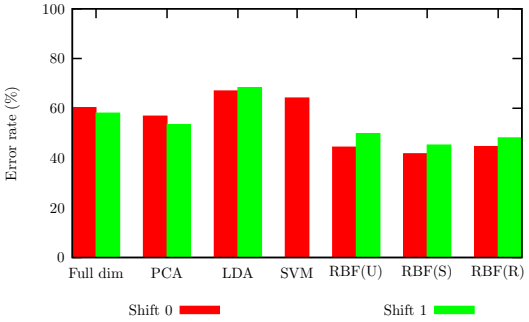
Techniques considered within this category are restricted to those that make use of the class labels of the training data. Only by being aware of the classes associated with disparate regions of measurement space can a transformation of the data be expected to project such disparate regions onto the same region of feature space. If the extracted features are successful in achieving this merging, the two, previously disparate, data distributions will be equally well represented by a single set of class exemplars. To clarify this with an example: The original representative of the MBT class was MBT1 (specifically MBT1-1 as listed in table 4.1) and it has been seen that classification performance against



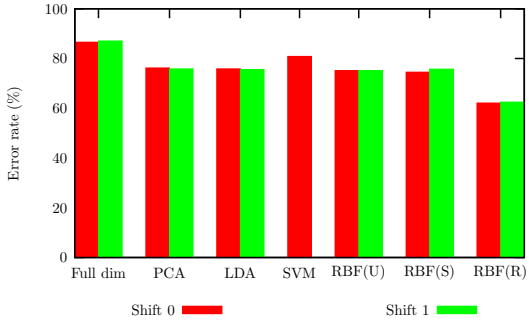
(a) *MBT2-4*



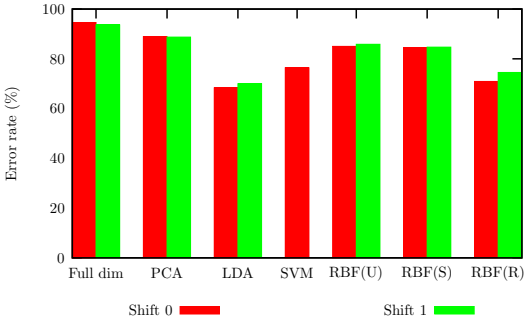
(b) *MBT2-5*



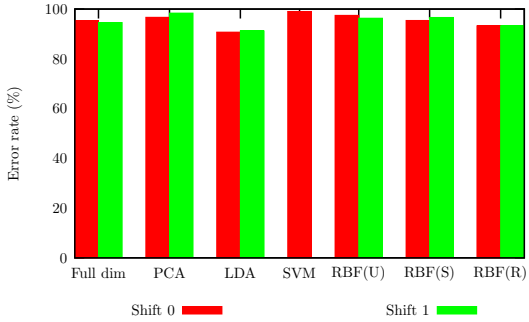
(c) *MBT2-6*



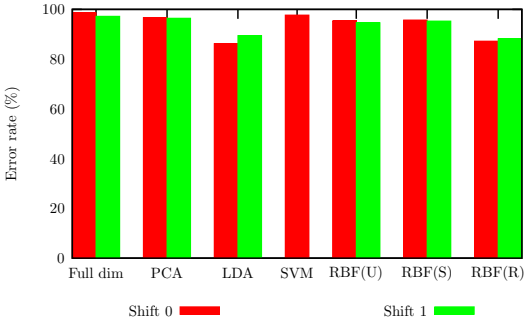
(d) *MBT4-1*



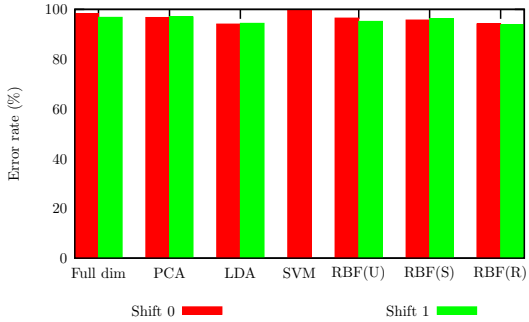
(e) *MBT4-2*



(f) *MBT5-4*

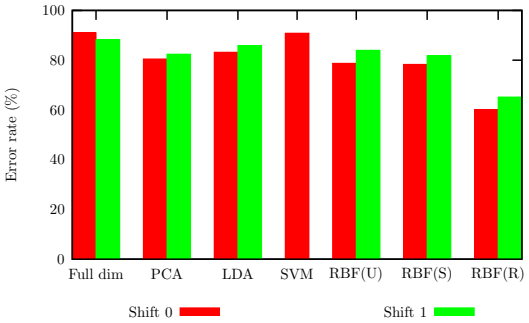


(g) *MBT5-5*

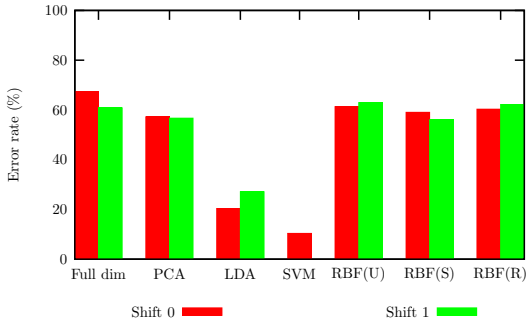


(h) *MBT5-6*

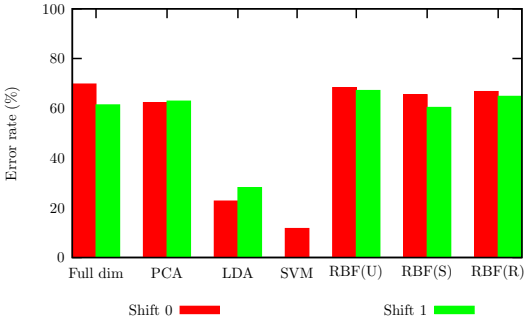
Figure 8.5: *Group 4, comparison of the small-training-group results. RBF(U) denotes RBF feature space obtained using the unsupervised loss function and projecting to 20 dimensions. RBF(S) denotes RBF feature space obtained using the supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using the rescaled inter-class distances and projecting to 20 dimensions.*



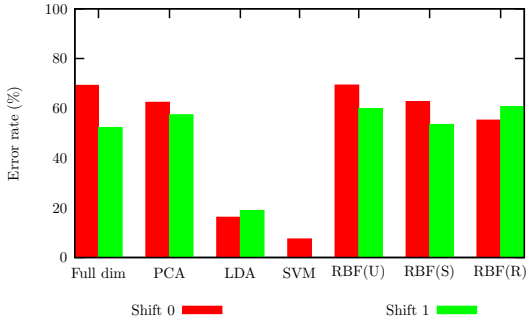
(i) *MBT4-4*



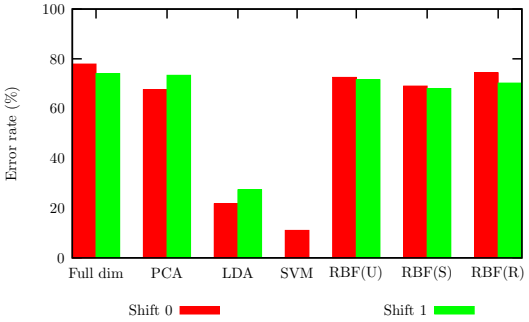
(j) *ADU3-2*



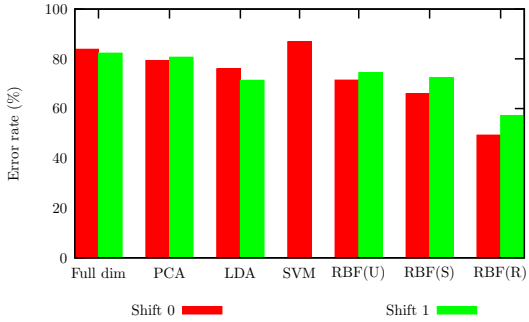
(k) *ADU3-3*



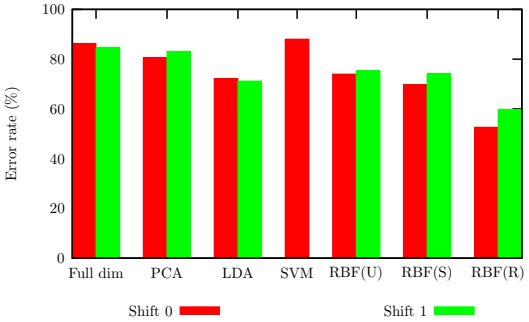
(l) *ADU3-4*



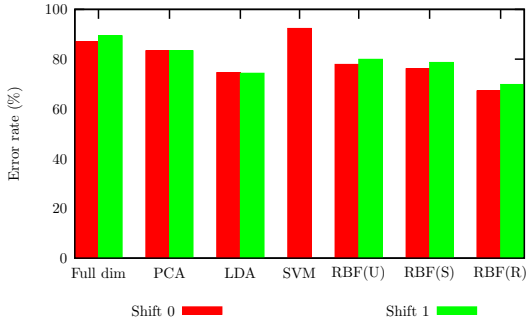
(m) *ADU3-5*



(n) *MBT3-7*

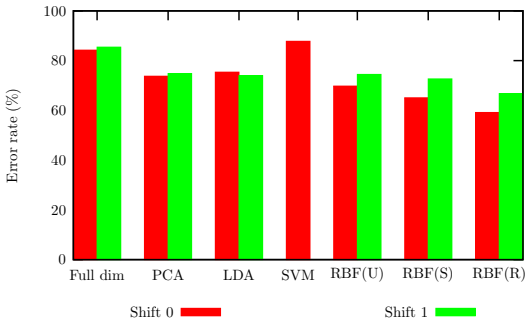


(o) *MBT3-8*



(p) *MBT3-9*

Figure 8.5: *Group 4, comparison of the small-training-group results. RBF(U) denotes RBF feature space obtained using the unsupervised loss function and projecting to 20 dimensions. RBF(S) denotes RBF feature space obtained using the supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using the rescaled inter-class distances and projecting to 20 dimensions (cont.)*



(q) *MBT3-10*

Figure 8.5: *Group 4, comparison of the small-training-group results. RBF(U) denotes RBF feature space obtained using the unsupervised loss function and projecting to 20 dimensions. RBF(S) denotes RBF feature space obtained using the supervised loss function with* $\rho = 0$ *and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using the rescaled inter-class distances and projecting to 20 dimensions (cont.)*

Table 8.1: *Summary of the data sets used for training the feature extraction, validating model parameters for feature extraction, and providing the class exemplars for the subsequent classification.*

Partition	Data set
Training	MBT1-1, APC1-1, ADU1-1, MBT2-1, ADU1-2, APC2-1
Validation	MBT1-8, APC1-4, ADU1-8, MBT2-7, MBT3-11, MBT5-7
Class exemplars	MBT1-1, APC1-1, ADU1-1

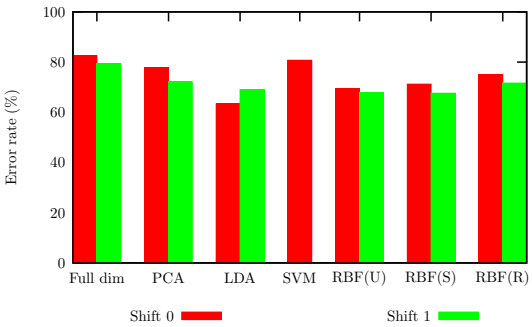
other MBT vehicles, such as MBT2, is generally poor. If a feature space could be obtained in which the regions for MBT1 and MBT2 coincided, then the use of the previous MBT1 exemplars (MBT1-1) in the classifier would produce improved results against MBT2.

For the feature-extraction methods, training using the extended training group, by combining the data listed in tables 4.1 and 4.8, but using only the small training group (table 4.1 alone) to provide class exemplars for the classifier allows an examination of whether the extracted features lead to a feature-space in which the small training group is more representative of a wider range of vehicle types. The data sets are summarized in table 8.1. The results compared are thus from:

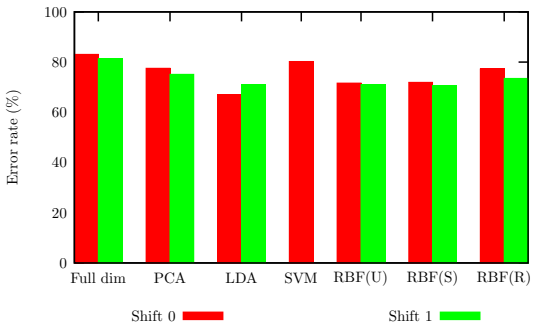
Full dim The class exemplars were provided by the extended training group. Classification was performed in the full-dimensional measurement-space. These results have not been previously quoted.

LDA The class exemplars were provided by the small training group. Classification was performed in the LDA feature-space determined by training on the extended training group. These results have not been previously quoted.

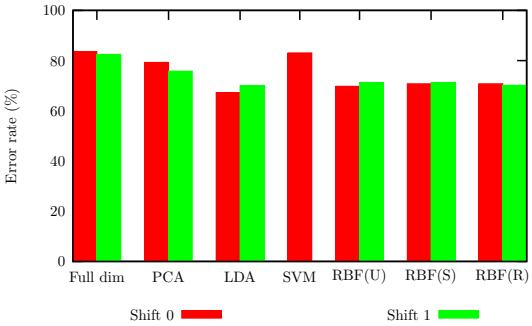
SVM The SVM classifier was trained on the extended training group. Model parameters



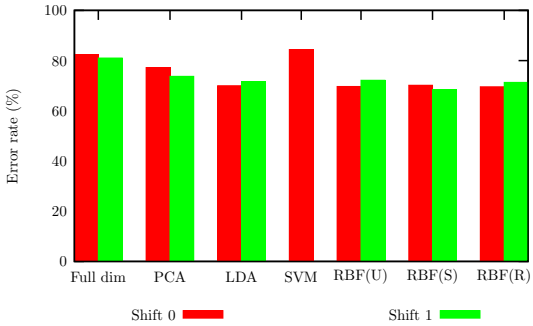
(a) *MBT3-1*



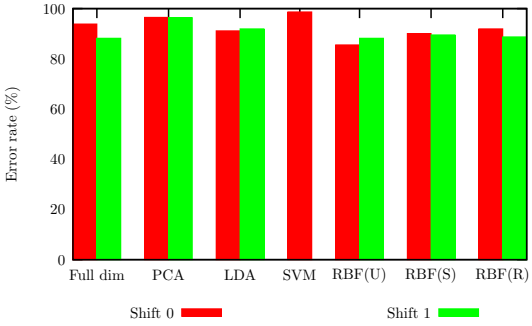
(b) *MBT3-2*



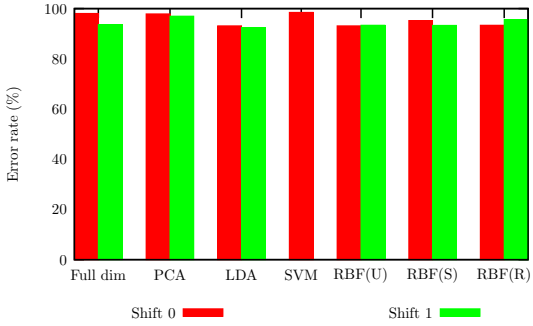
(c) *MBT3-4*



(d) *MBT3-5*

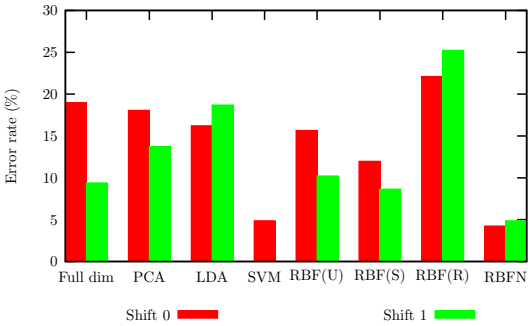


(e) *MBT5-1*

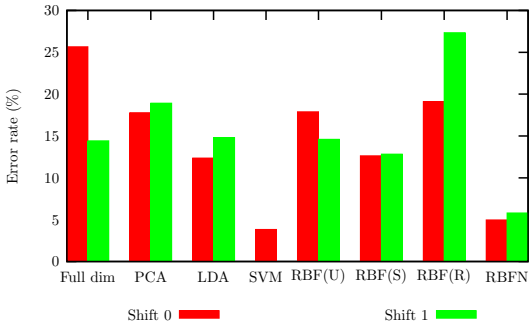


(f) *MBT5-2*

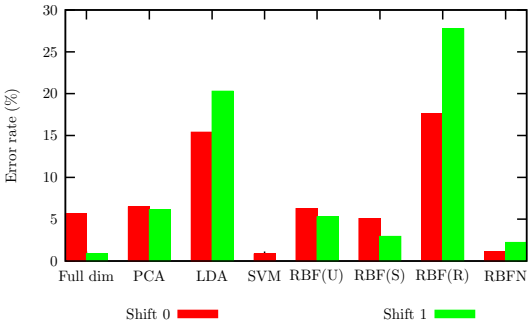
Figure 8.6: *Group 5, comparison of the small-training-group results. RBF(U) denotes RBF feature space obtained using the unsupervised loss function and projecting to 20 dimensions. RBF(S) denotes RBF feature space obtained using the supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using the rescaled inter-class distances and projecting to 20 dimensions.*



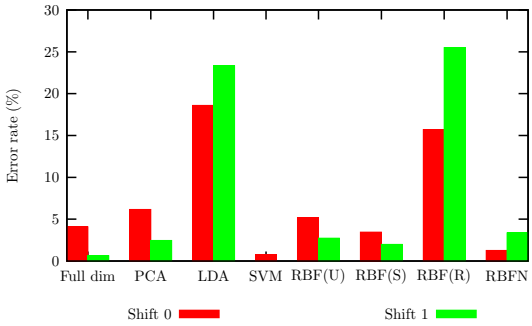
(a) *MBT1-3*



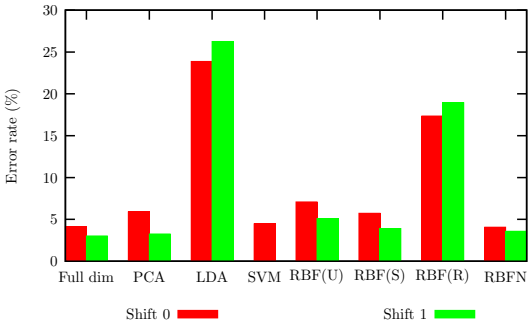
(b) *MBT1-4*



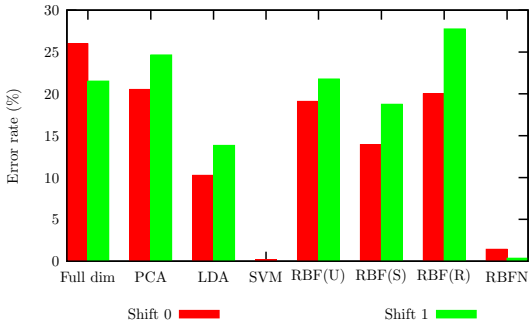
(c) *MBT1-5*



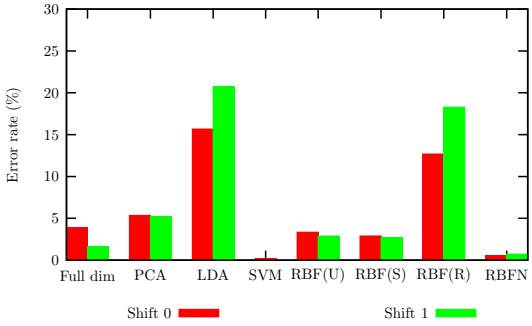
(d) *MBT1-7*



(e) *APC1-2*



(f) *ADU1-3*



(g) *ADU1-5*

Figure 8.7: *Group 1, comparison of the small-training-group results including RBFN classifier. RBF(U) denotes RBF feature space obtained using the unsupervised loss function and projecting to 20 dimensions. RBF(S) denotes RBF feature space obtained using the supervised loss function with* $\rho = 0$ *and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using the rescaled inter-class distances and projecting to 20 dimensions. RBFN denotes the RBFN used to perform classification.*

were selected based on classification performance against the extended validation group. These results were previously quoted in section 7.4, in figure 7.12.

RBF(S) The RBFN was trained using the extended training group and the supervised loss function ($\rho = 0$). Model selection was based on the stress calculated against the extended validation group. The class exemplars were provided by the small training group. Classification was performed in 20-dimensional RBF feature-space. These results have not been previously quoted.

RBF(R) The RBFN was trained using the rescaled inter-class distances and unsupervised loss function on the extended training group. Model selection was based on the stress calculated against the extended validation group. The class exemplars were provided by the small training group. Classification was performed in 20-dimensional RBF feature space. These results have not been previously quoted.

All figures within this section use the following colour scheme. Results using the extended data sets without shifting are shown in blue, and with shifting are shown in purple. For comparison, results from the previous section using only the small data sets are also given without shifting, in red, and with shifting, in green. In this way, comparisons can readily be made both to assess the relative performance of the different techniques using the additional training data and how these performances changed for each technique relative to the small-training-group results.

The questions being posed are, given the availability of additional vehicles in the training data:

- How well did each technique extract features within the data that reduced unwanted variability?
- How did this compare to classification performed in measurement space, with no feature extraction performed?
- How did this relate to the previous scenario using the small training group?

Viewed this way, it is clear why classification in feature space only used class exemplars from the small training group. The goal was to assess the feature-extraction. Using the additional vehicles also as class exemplars for the classifier would mask the effect of the feature-extraction. Not having separate feature extraction stages, the full-dimension and SVM classifiers were trained using the extended training data. As such they provide a comparison with classification performed with the benefit of the additional training data during classifier training.

8.3.1 Group 1

The summary of results for data sets comprising Group 1 is shown in figure 8.8.

For the majority of the data sets, the error rate for the LDA-trained classifier using the extra data was 2–3 times higher than that using the small training group. The data sets with this higher error rate were the MBT1 test data: MBT1-3 (figure 8.8a), MBT1-4 (figure 8.8b), MBT1-5 (figure 8.8c), MBT1-7 (figure 8.8d); and the APC1 test data: APC1-2 (figure 8.8e). Only for ADU1: ADU1-3 (figure 8.8f) and ADU1-5 (figure 8.8g) did LDA show a slight decrease in error rate with the addition of the extra training data.

With the exception of LDA applied to these latter two data sets, the general trend for all techniques was for the error rate to increase with the additional training data, with LDA being the most severely affected. The most consistent between the two training scenarios was the full-dimension results, where classification was performed in the original measurement space. In fact, the full-dimension error rates revealed very similar classification performance obtained using the small and extended training groups. Clearly the additional class exemplars were sufficiently different from these test data to not interfere with the 1-NNR decision boundaries in measurement space.

The relative increases in SVM error rates were considerable when the extra training data were added. Given the vanishingly small SVM error rate for ADU1-3 (figure 8.8f) and ADU1-5 (figure 8.8g) when trained using the small training group, the relative increase in error rate with the additional training data was large indeed. This large increase in error rate is surprising given that the additional ADU representative in the extended training group was another example of ADU1 (table 4.8).

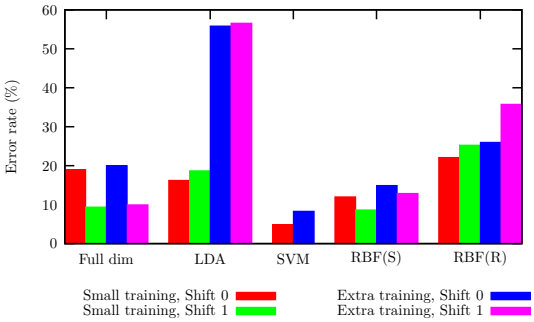
This suggests that attempting to improve the generalization capability of a classifier across extended operating conditions for one class, i.e. adding additional vehicle types to a class, can reduce the generalization capability against another class. The addition of extra training data added enough 'confusion' that the SVM sometimes no longer performed as well as the full-dimension classifier, as seen with APC1-2 (figure 8.8e) and ADU1-5 (figure 8.8g), although it generally compared quite well.

Of the RBF implementations, that using the supervised loss function with $\rho = 0$ demonstrated a lower error rate than that using rescaled inter-class distances. This was true when using either the small or extended training group.

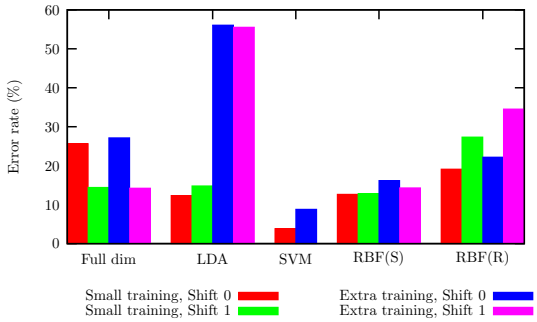
8.3.2 Group 2

The summary of results for data sets comprising Group 2 is shown in figure 8.9.

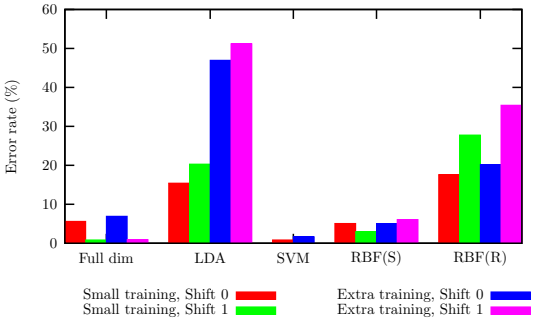
LDA once again showed considerable increase in error rate on some data sets, by more than a factor of two for MBT1-6 (figure 8.9b) and APC1-3 (figure 8.9c), upon the inclusion



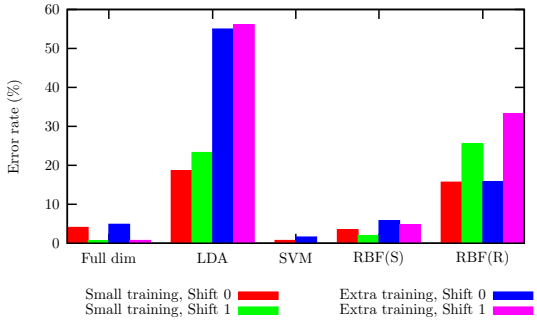
(a) *MBT1-3*



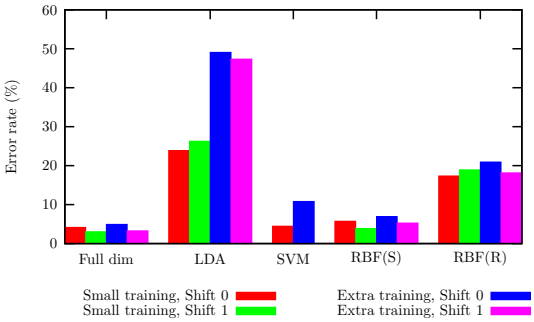
(b) *MBT1-4*



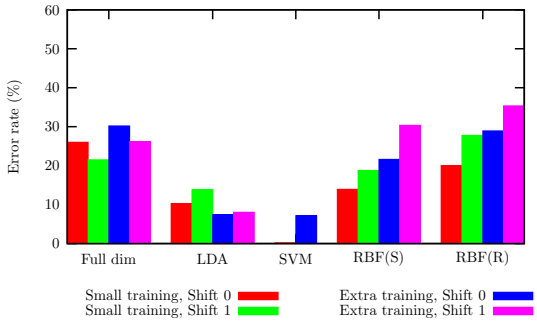
(c) *MBT1-5*



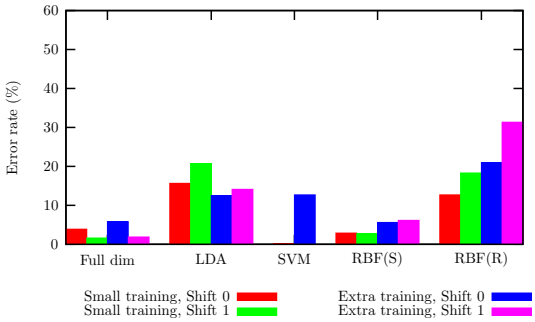
(d) *MBT1-7*



(e) *APC1-2*

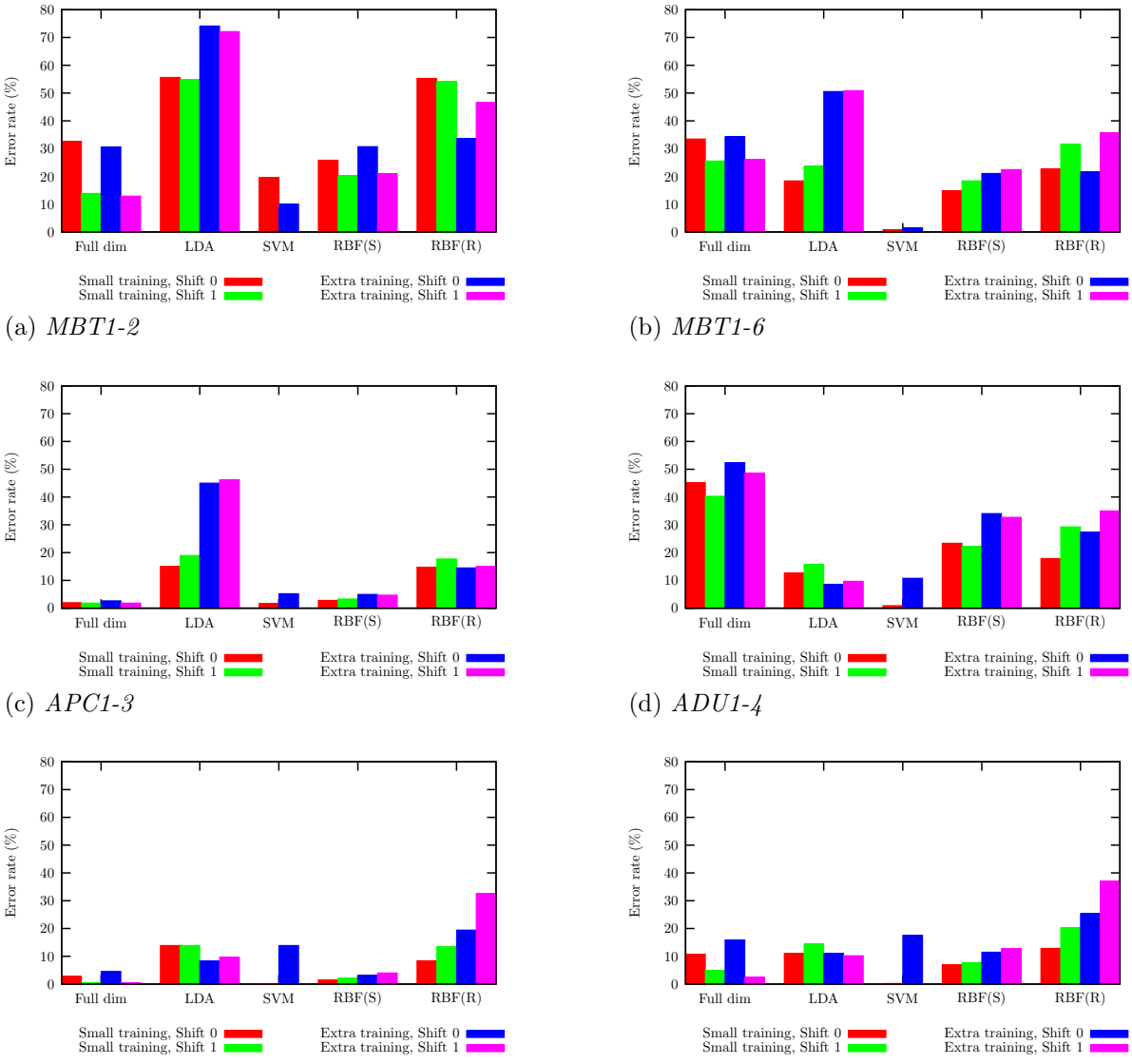


(f) *ADU1-3*



(g) *ADU1-5*

Figure 8.8: *Group 1, comparison of the extended-training-group results. Relevant small-training-group results are included for reference. RBF(S) denotes RBF feature space obtained using supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using rescaled inter-class distances and projecting to 20 dimensions.*



(a) *MBT1-2*

(b) *MBT1-6*

(c) *APC1-3*

(d) *ADU1-4*

(e) *ADU1-6*

(f) *ADU1-7*

Figure 8.9: *Group 2, comparison of the extended-training-group results. Relevant small-training-group results are included for reference. RBF(S) denotes RBF feature space obtained using supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using rescaled inter-class distances and projecting to 20 dimensions.*

of the additional training data. Such increases turned LDA into the worst performer for those data sets and for MBT1-2 (figure 8.9a).

Broadly speaking, the tendency was, once again, for error rates to increase when the extra training data were added. The main exception was MBT1-2 (figure 8.9a), which tended to have the highest initial error rates. Both SVM and RBF(R) saw a reduction in error rate on this data set with the inclusion of the extra training data.

Other than for MBT1, the error rates for SVM demonstrated very large relative increases in error rate upon the inclusion of the extra training data. For data sets APC1-3 (figure 8.9c), ADU1-6 (figure 8.9e), and ADU1-7 (figure 8.9f), this raised the SVM error rates above those for the full-dimension classifier, whereas before they had been similar or lower.

Of the supervised RBF approaches, the RBF(S) was, again, generally the better performing. This applied both to the small and extended training group scenarios.

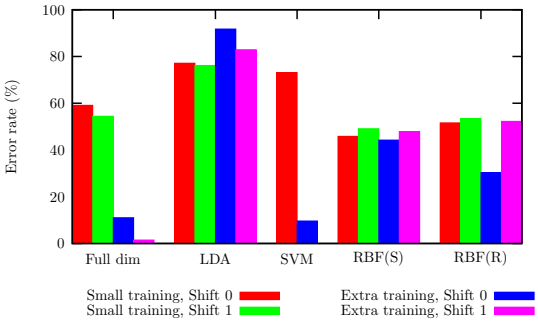
8.3.3 Group 3

The summary of results for data sets comprising Group 3 is shown in figure 8.10.

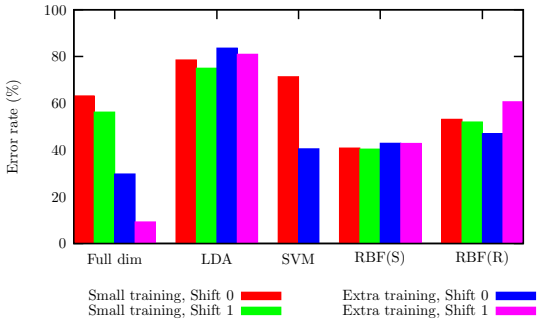
The results for the two MBT2 data sets are shown in figures 8.10a and 8.10b. This vehicle was one added with the extended training data. The two classifiers that directly use the additional training data, the full-dimension and SVM classifiers, are the only techniques to show a clear drop in error rate with the extra data. With the exception of the RBF(R)-with-no-shifting applied to MBT2-2, none of the techniques for feature extraction define a projection that increases the similarity between MBT1 and MBT2. However, both the RBF(S) and RBF(R) features did yield error rates better than random when trained using only the small training set.

APC2 was another vehicle included in the extended training data and results for APC2-2 and APC2-3 are given in figures 8.10e and 8.10f respectively. Previously, using only the small training set, the only approach that gave results better than random was classification performed with the original images. With the extra data, the full-dimension, SVM, and RBF-based classifiers all show clear falls in error rate. This brings the error rate below the level of random chance for the SVM and RBF-based approaches. The SVM classifier, in particular, previously performed very poorly on this vehicle and now performs very well with the additional training data.

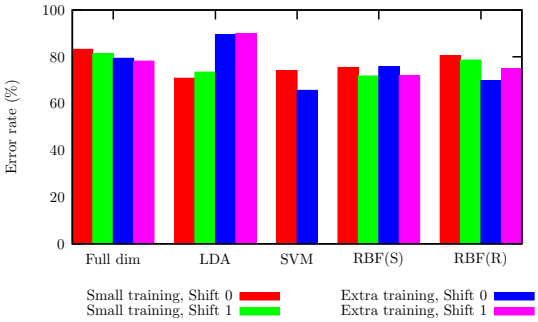
Using the small training set, the error rates for ADU2-1 in figure 8.10i and ADU3-1 in figure 8.10j were approximately at the level of random chance, or worse, for most techniques. The features calculated by LDA did give rise to low error rates when trained on the small training set; these error rates become vanishingly small when LDA is trained using the additional training data. However, the error rates for these data sets using the



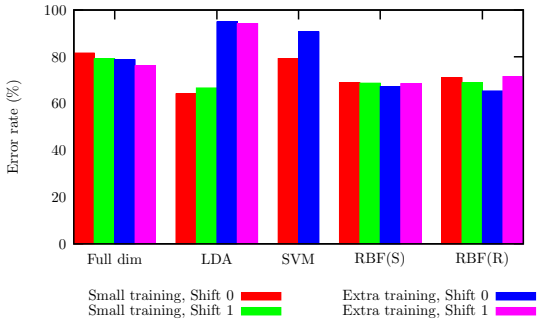
(a) *MBT2-2*



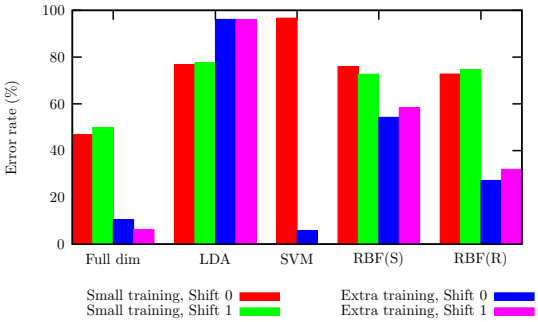
(b) *MBT2-3*



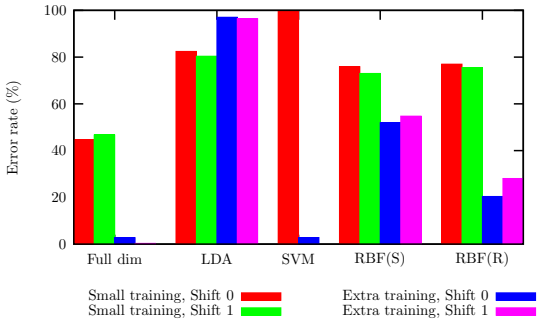
(c) *MBT3-3*



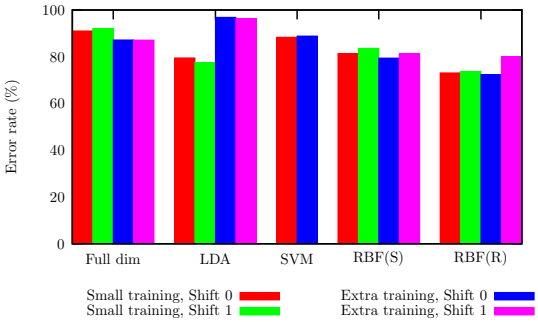
(d) *MBT3-6*



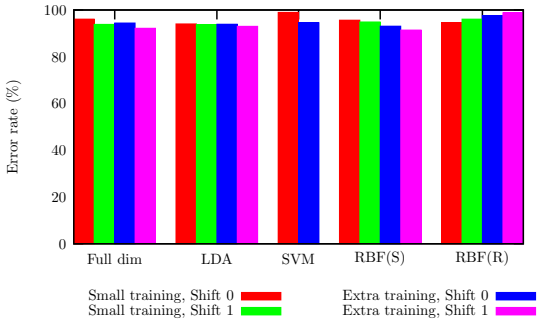
(e) *APC2-2*



(f) *APC2-3*

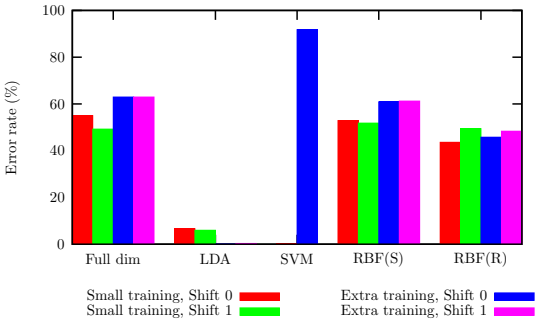


(g) *MBT4-3*

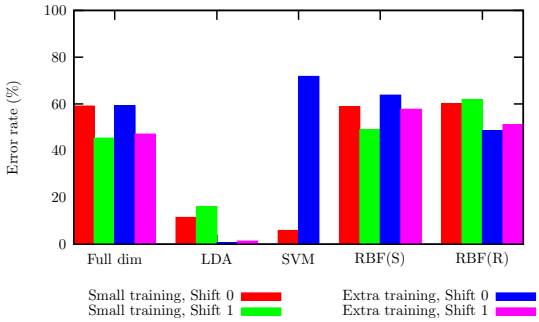


(h) *MBT5-3*

Figure 8.10: *Group 3, comparison of the extended-training-group results. Relevant small-training-group results are included for reference. RBF(S) denotes RBF feature space obtained using supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using rescaled inter-class distances and projecting to 20 dimensions.*



(i) *ADU2-1*



(j) *ADU3-1*

Figure 8.10: *Group 3, comparison of the extended-training-group results. Relevant small-training-group results are included for reference. RBF(S) denotes RBF feature space obtained using supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using rescaled inter-class distances and projecting to 20 dimensions (cont.)*

SVM classifier were previously vanishingly small or very low when trained on the small training set and become high when the additional training data are used. This increase takes the error rate for the SVM classifier well above the level of random chance.

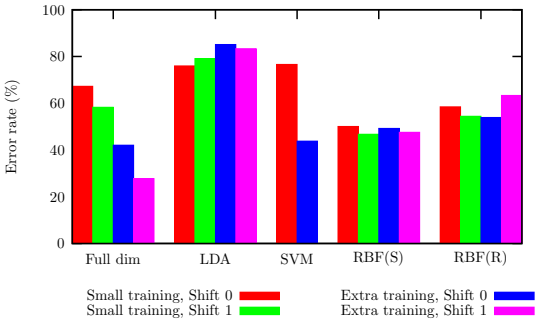
The error rates for MBT3 in figures 8.10c and 8.10d, MBT4 in figure 8.10g, and MBT5 in figure 8.10h remain broadly at or above the level of random chance.

8.3.4 Group 4

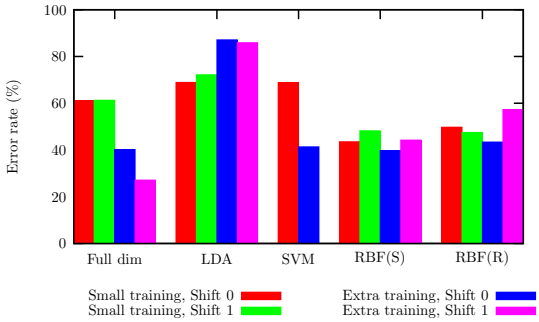
The summary of results for data sets comprising Group 4 is shown in figure 8.11.

Error rates for the MBT2 data sets are shown in figures 8.11a, 8.11b, and 8.11c. Using the small training set, the results better than random were those for classification performed using the original images and using RBF-derived features; classification using LDA-derived features and using the SVM classifier gave results worse than random. The addition of the extra training data gives rise to a big drop in error rate for the full-dimension and SVM classifiers, with the error rates for both becoming much lower than for random chance. The results for classification using the RBF-derived features show negligible change with the additional training data, but at least remain better than random.

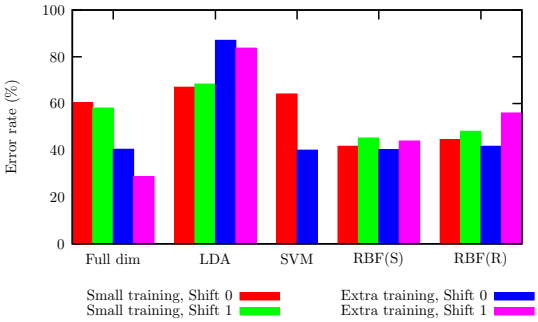
The results for ADU3, in figures 8.11j through 8.11m, show that using the small training set, only the results using LDA and the SVM were better than random. With the addition of the extra training data, the error rate for the LDA features greatly decreases and the error rate for the SVM classifier greatly increases. This increase in the SVM error rate brings its performance approximately to the level of the other techniques. The other techniques show little change with the addition of the extra training data.



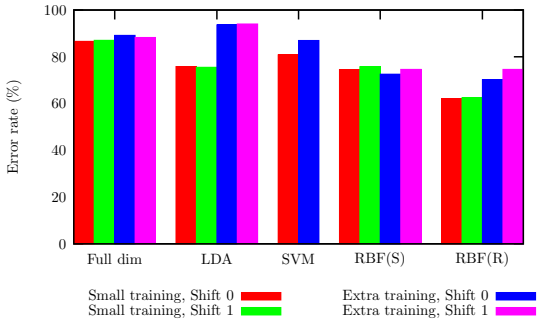
(a) *MBT2-4*



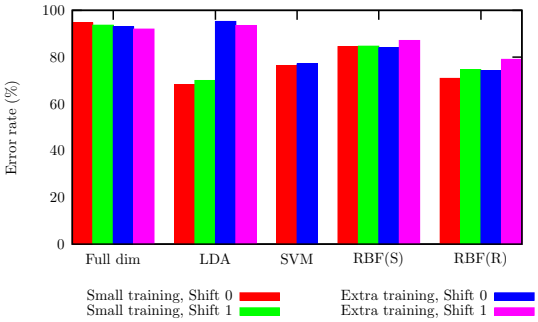
(b) *MBT2-5*



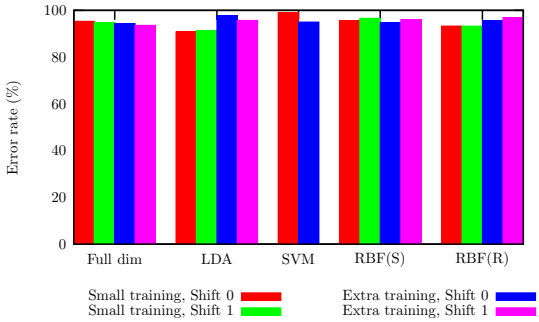
(c) *MBT2-6*



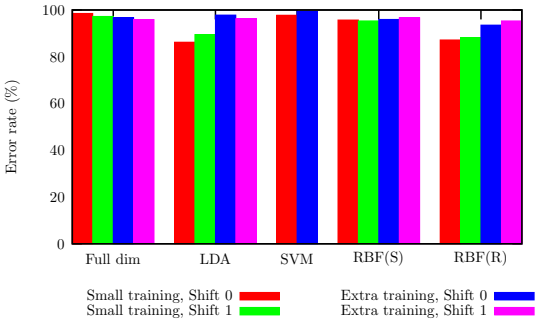
(d) *MBT4-1*



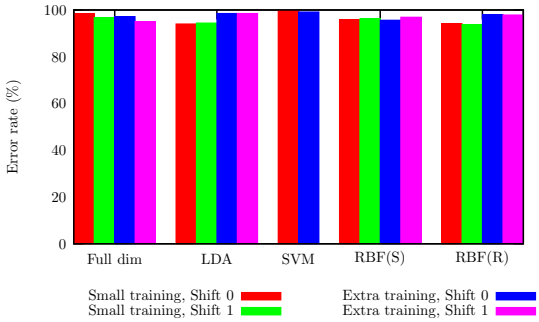
(e) *MBT4-2*



(f) *MBT5-4*

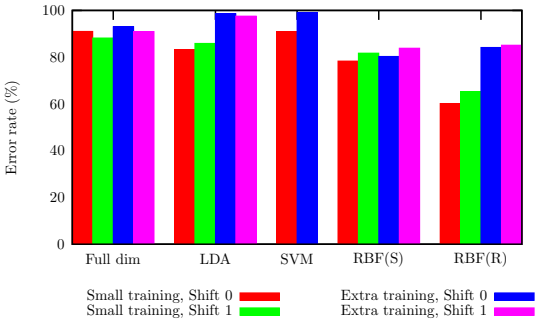


(g) *MBT5-5*

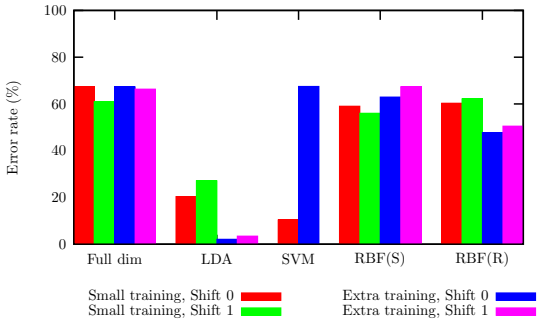


(h) *MBT5-6*

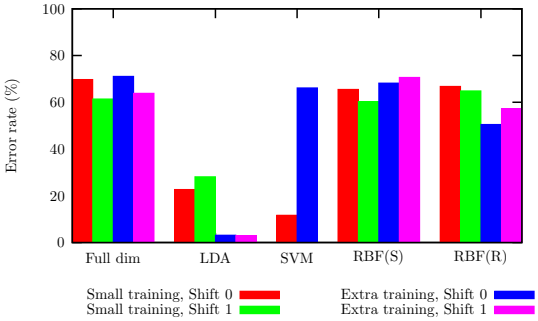
Figure 8.11: *Group 4, comparison of the extended-training-group results. Relevant small-training-group results are included for reference. RBF(S) denotes RBF feature space obtained using supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using rescaled inter-class distances and projecting to 20 dimensions.*



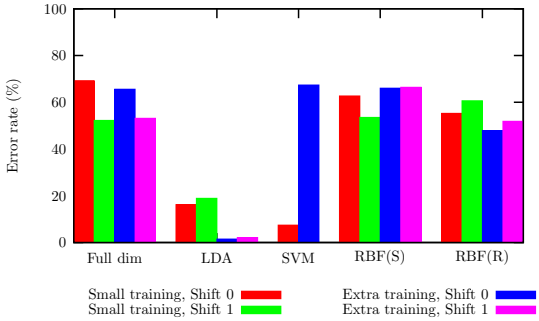
(i) *MBT4-4*



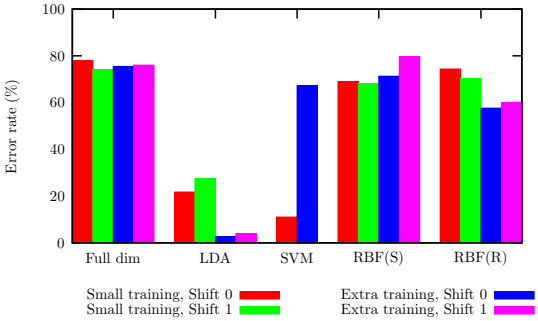
(j) *ADU3-2*



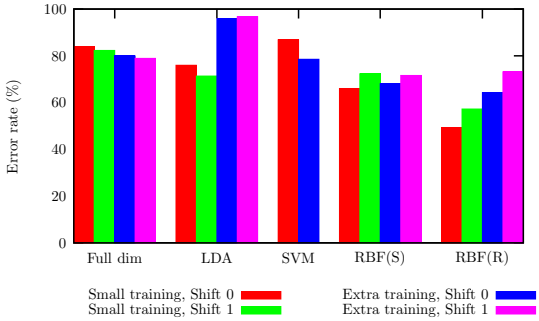
(k) *ADU3-3*



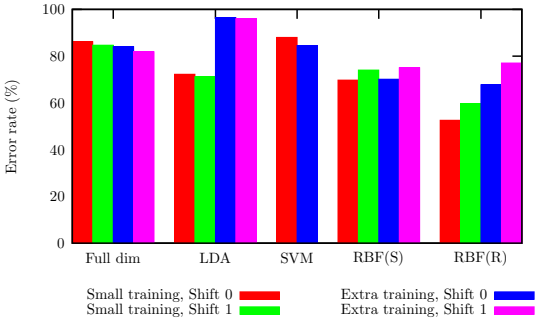
(l) *ADU3-4*



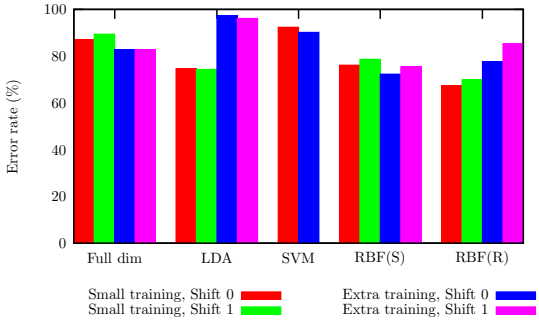
(m) *ADU3-5*



(n) *MBT3-7*

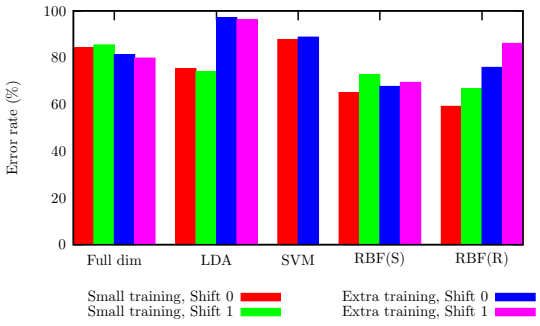


(o) *MBT3-8*



(p) *MBT3-9*

Figure 8.11: *Group 4, comparison of the extended-training-group results. Relevant small-training-group results are included for reference. RBF(S) denotes RBF feature space obtained using supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using rescaled inter-class distances and projecting to 20 dimensions (cont.)*



(q) *MBT3-10*

Figure 8.11: *Group 4, comparison of the extended-training-group results. Relevant small-training-group results are included for reference. RBF(S) denotes RBF feature space obtained using supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using rescaled inter-class distances and projecting to 20 dimensions (cont.)*

For MBT3, in figures 8.11n through 8.11q, all methods produce error rates worse than random, with the exception of classification using the RBF(R)-derived features. However, even classification using the RBR(R)-derived features shows an increase in error rate both as shifting and as the additional training data are included. This is such that when the additional training data are used as well as shifting, the error rate is always worse than random.

Error rates for the other vehicles, MBT4 and MBT5, are all approximately at, or worse than, the level of random chance.

8.3.5 Group 5

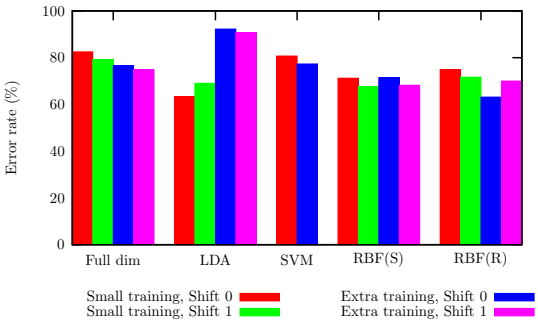
The summaries of results for data sets comprising Group 5 are shown in figure 8.12.

All error rates within this group are at or above the level for random chance. No improvement is seen with the inclusion of the additional training data.

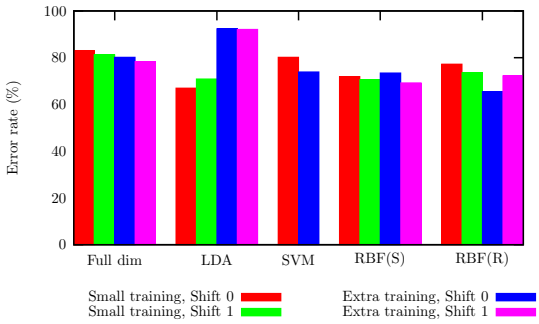
8.4 Box-Cox transformation

The techniques considered within this category concern different applications of the Box-Cox transformation. All experiments used only the small training group. The techniques compared are from:

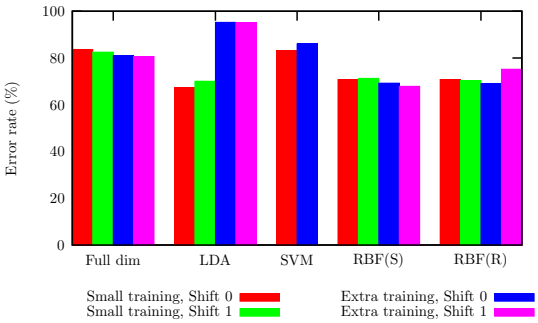
Full dim The class exemplars were provided by the small training group. Classification was performed in the full-dimensional measurement-space. The Box-Cox transformation was applied to all data samples prior to classification using the Euclidean metric. These results have not been previously quoted, although results are also



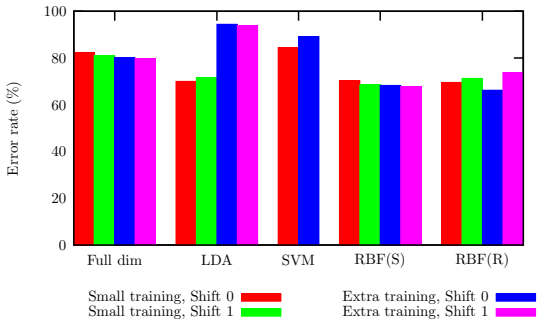
(a) *MBT3-1*



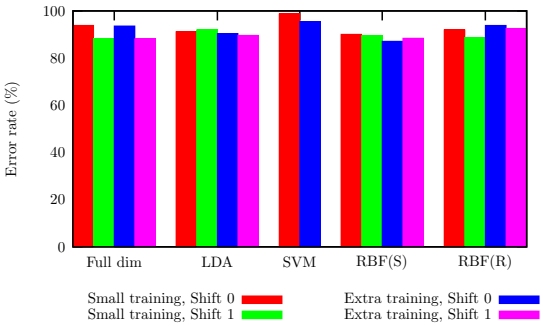
(b) *MBT3-2*



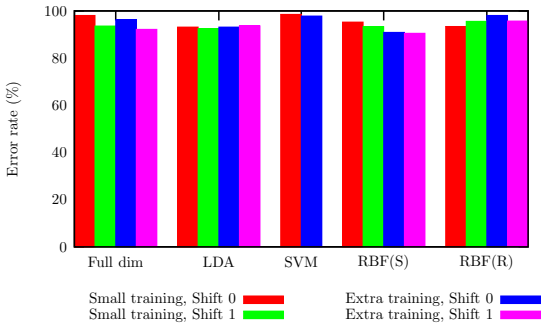
(c) *MBT3-4*



(d) *MBT3-5*



(e) *MBT5-1*



(f) *MBT5-2*

Figure 8.12: *Group 5, comparison of the extended-training-group results. Relevant small-training-group results are included for reference. RBF(S) denotes RBF feature space obtained using supervised loss function with $\rho = 0$ and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using rescaled inter-class distances and projecting to 20 dimensions.*

provided for the full-dimension classifier without the Box-Cox transformation and these results have previously been quoted in section 5.4, in figure 5.14.

RBF(U) The RBFN was trained using the small training group and the unsupervised loss function. The Euclidean distance metric was used to calculate the dissimilarities. Model selection was based on the stress calculated against the small validation group. The class exemplars were provided by the small training group. Classification was performed in 20-dimensional RBF feature space. The classifier used the Euclidean metric. The Box-Cox transformation was never applied to the data. These results were previously quoted in section 6.2.2, in figure 6.6.

RBF(BM) The RBFN was trained using the small training group and the unsupervised loss function. The Box-Cox metric was used only to calculate the input dissimilarities. Model selection was based on the stress calculated against the small validation group. The class exemplars were provided by the small training group. Classification was performed in 20-dimensional RBF feature space. The classifier used the Euclidean metric. These results were previously quoted in section 6.5.2 in figure 6.24.

RBF(BD) The RBFN was trained using the small training group and the unsupervised loss function. All data (training, validation, and test) were pre-processed by applying the Box-Cox transformation. The Euclidean metric was then used both for training the RBFN and for classification. Model selection was based on the stress calculated against the small validation group. The class exemplars were provided by the small training group. Classification was performed in 20-dimensional RBF feature space. These results were previously quoted in section 6.6.2 in figure 6.30.

Thus, RBF(U) made no use of the Box-Cox transformation, RBF(BM) attempted to learn the Box-Cox transformation as part of the feature-extraction process, and RBF(BD) acted on data globally-transformed by Box-Cox and so did not have to 'learn' the transformation itself.

All figures within this section use the following colour scheme. Results obtained using the Box-Cox transformation are coloured in blue and purple. Results obtained without any use of the Box-Cox transformation are coloured red and green. In this way, comparisons can readily be made both to assess the relative performances of the different Box-Cox methods and how they compare to the methods that do not use the Box-Cox transformation.

Questions being posed in this category are:

- What benefit was there in using the Box-Cox transformation?

- How well did the RBFN learn the Box-Cox transformation as a part of the feature-extraction process?
- How well did this generalize?
- How did learning the Box-Cox transformation compare to explicitly applying it to the data?

8.4.1 Group 1

The summary of results for data sets comprising Group 1 is shown in figure 8.13.

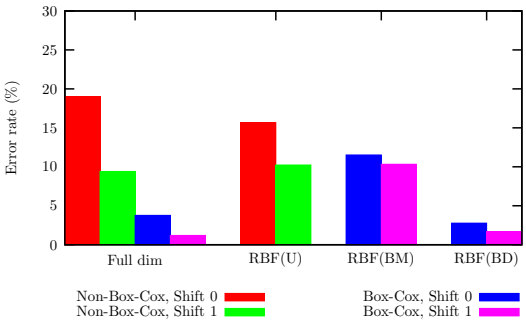
Considering first the full-dimension results; the effect on the error rate of performing the classification on Box-Cox transformed data, rather than the original measurements, is often dramatic. MBT1-4 (figure 8.13b) and ADU1-3 (figure 8.13f) see an order of magnitude drop in error rate. For MBT1-3 (figure 8.13a), MBT1-5 (figure 8.13c) and MBT1-7 (figure 8.13d), error rates become a small fraction of the original values. In particular, error rates for MBT1-5 and MBT1-7 with shift ± 1, and ADU1-5 (figure 8.13g) for both shift 0 and shift ± 1, become vanishingly small. For APC1-2 (figure 8.13e) all techniques yield similar performances.

The error rates for full-dimension classification using the original (non-Box-Cox-transformed) data and RBF(U) are broadly similar within each data set. This is also true for comparing between the RBF(U) and the RBF(BM), which used the Box-Cox metric for deriving inter-sample dissimilaries. Another pair of results which are broadly similar are those of the Box-Cox full-dimension and RBF(BD). These correspond to classification performed in original measurement space, and in a feature space determined by the RBFN trained using the unsupervised loss function, but with all input data (both training and test) pre-processed by application of the Box-Cox transformation in each case. These latter two cases yielded the lowest error rates by a considerable margin. Thus the best performance, suggesting the best robustness to the variations in target configuration, is obtained by globally applying the Box-Cox transformation to all input data. Subsequent RBFN feature extraction utilizing 20 features then leads to classification performance about as good as that obtained using the full-dimension data.

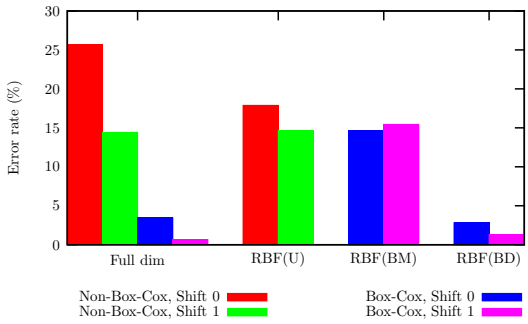
8.4.2 Group 2

The summary of results for data sets comprising Group 2 is shown in figure 8.14.

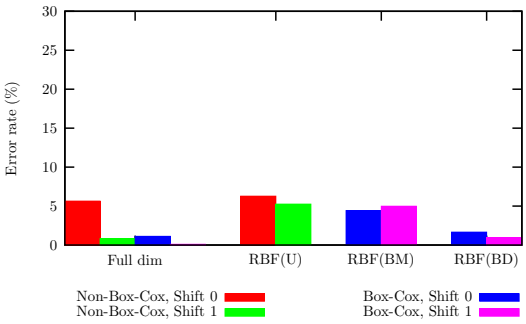
Other than for MBT1-2 (figure 8.14a), which again presents a somewhat anomalous case, the same observations made for Group 1 apply also to this group. The RBF(U)



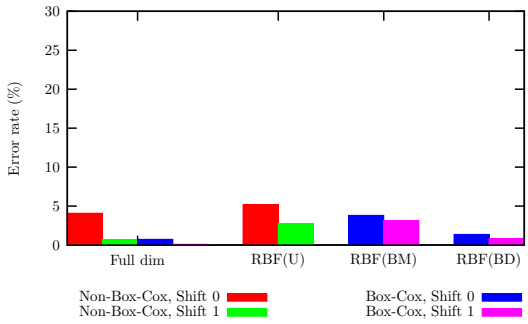
(a) *MBT1-3*



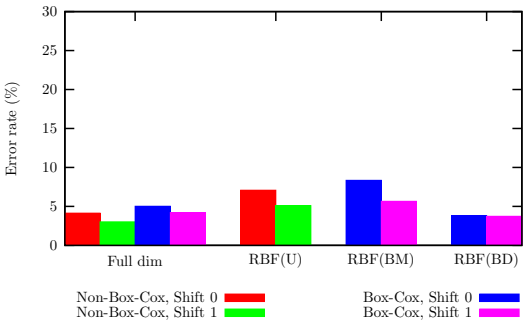
(b) *MBT1-4*



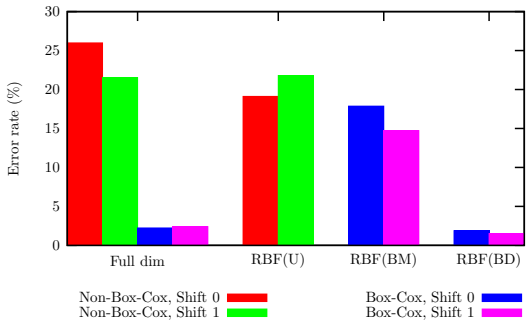
(c) *MBT1-5*



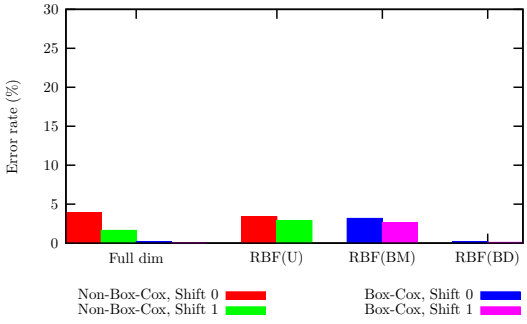
(d) *MBT1-7*



(e) *APC1-2*

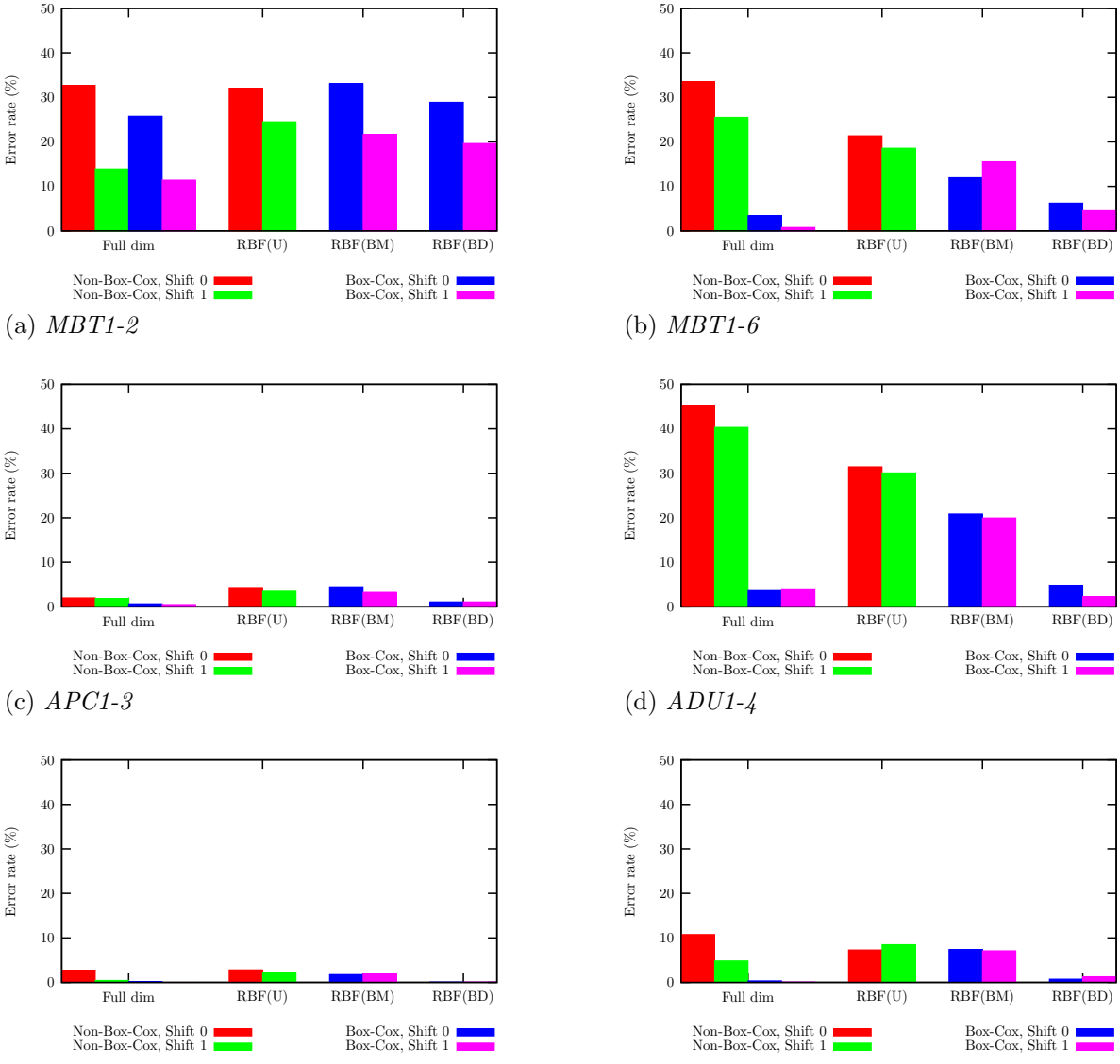


(f) *ADU1-3*



(g) *ADU1-5*

Figure 8.13: *Group 1, comparison of Box-Cox results. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, and RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. All projections were to 20 dimensions.*



(a) *MBT1-2*

(b) *MBT1-6*

(c) *APC1-3*

(d) *ADU1-4*

(e) *ADU1-6*

(f) *ADU1-7*

Figure 8.14: *Group 2, comparison of Box-Cox results. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, and RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. All projections were to 20 dimensions.*

largely attains the same error rates as achieved using the full-dimension non-Box-Cox-transformed data. Using the Box-Cox metric to derive inter-sample distances, RBF(BM), typically achieves little or no reduction in error rate relative to RBF(U), which uses the Euclidean metric. The lowest error rates come from using data that have been pre-processed by the Box-Cox transformation, whether classification is performed in the full-dimension space or the RBF feature space (RBF(BD)).

The MBT1-2 results are atypical for the group. The application of the Box-Cox transformation makes little difference to the error rate. This is in contrast even to the other MBT1 data set in the group, MBT1-6 in figure 8.14b, which starts with a similar error rate using the full-dimension data but this drops by a large factor after the Box-Cox transformation is applied.

8.4.3 Group 3

The summary of results for data sets comprising Group 3 is shown in figure 8.15.

Broadly, the RBF(U) and RBF(BM) features produce similar error rates, as do the full-dimension with Box-Cox and the RBF(BD) features. Some data sets show a drop in error rate with the application of the Box-Cox transformation and some show an increase.

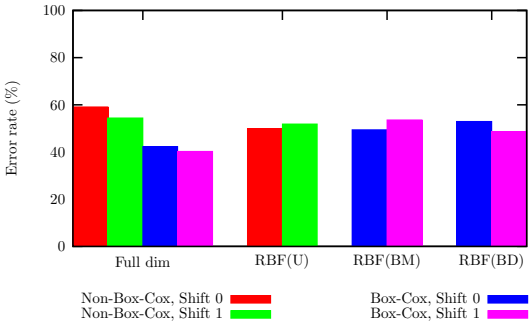
The data sets for MBT2, in figures 8.15a and 8.15b, achieve error rates lower than random chance for all techniques. The application of the Box-Cox transformation lowers the error rate for the full-dimension data, but the similarity of the RBF(U) and RBF(BD) error rates shows that the Box-Cox transformation does not noticeably affect performance when classification is performed on RBF-derived features.

For the MBT3 data sets, in figures 8.15c and 8.15d, the error rates obtained when the Box-Cox transformation is not applied are approximately at the level of, or worse than, random chance. Applying the Box-Cox transformation and then classifying either in the full-dimension space or RBF-derived feature space lowers the error rate to better than random.

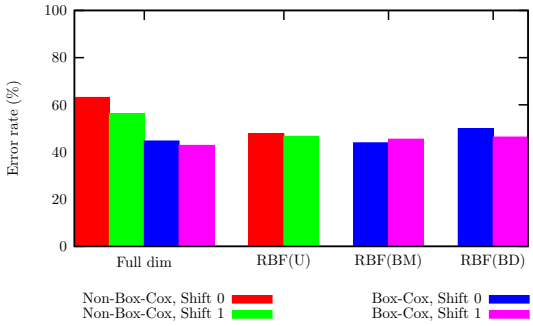
APC2, in figures 8.15e and 8.15f, has error rates better than random for classification in full-dimension space without the Box-Cox transformation applied. Application of the Box-Cox transformation, however, causes this error rate to rise well above the level of random chance. All other techniques produce results approximately at, or worse than, the level of random chance.

MBT4-3, in figure 8.15g, has a full-dimension error-rate without the Box-Cox transformation that is worse than would be obtained by random. The application of the Box-Cox transformation reduces this error rate below that of random-chance. Comparing the RBF(U) and RBF(BD) results shows a similar effect.

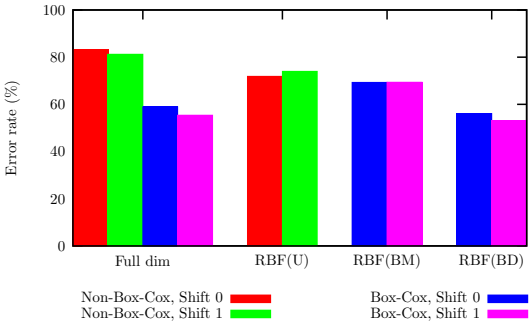
For ADU2-1, in figure 8.15i, all error rates are better than random. Applying the Box-



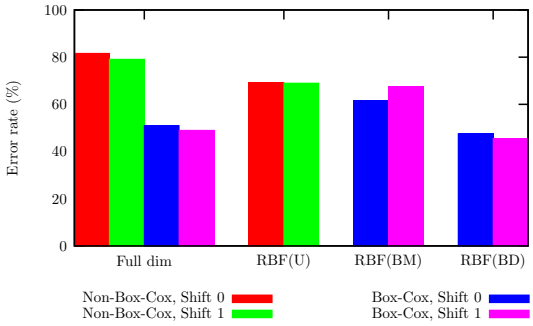
(a) *MBT2-2*



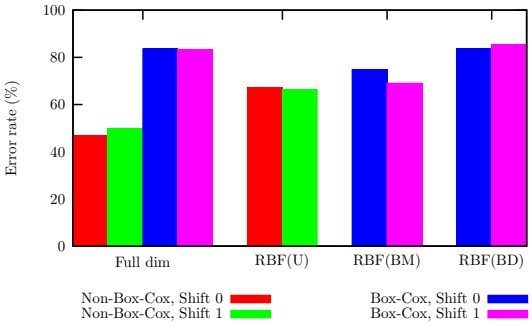
(b) *MBT2-3*



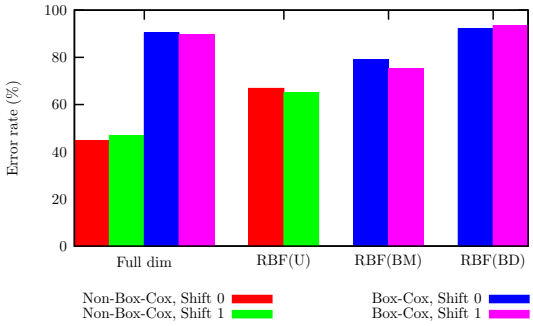
(c) *MBT3-3*



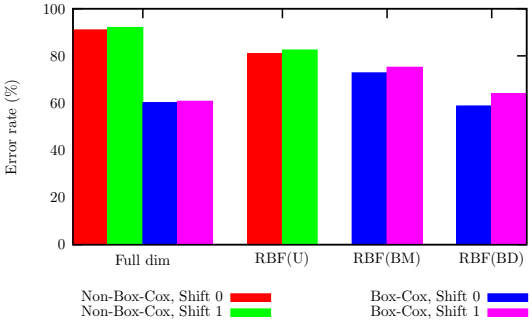
(d) *MBT3-6*



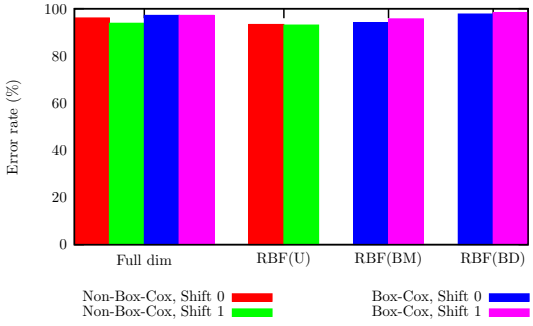
(e) *APC2-2*



(f) *APC2-3*

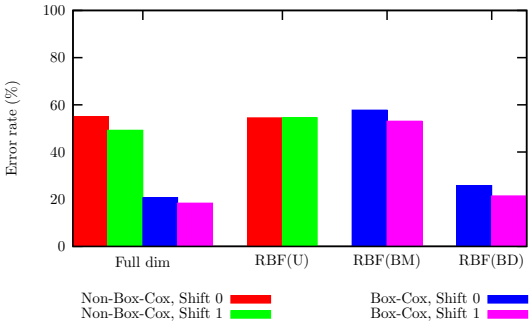


(g) *MBT4-3*

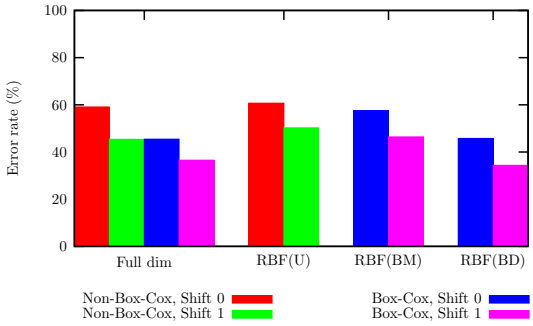


(h) *MBT5-3*

Figure 8.15: *Group 3, comparison of Box-Cox results. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, and RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. All projections were to 20 dimensions.*



(i) *ADU2-1*



(j) *ADU3-1*

Figure 8.15: *Group 3, comparison of Box-Cox results. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, and RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. All projections were to 20 dimensions (cont.)*

Cox transformation reduces the error rate both for classification using full-dimension data and RBF-derived features. A similar response is obtained for ADU3-1, in figure 8.15j, although the reduction in error rate is smaller.

8.4.4 Group 4

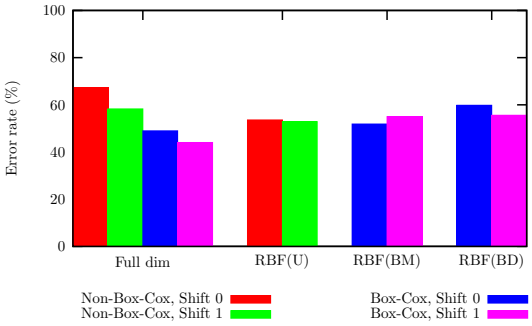
The summary of results for data sets comprising Group 4 is shown in figure 8.16.

The MBT2 results, shown in figures 8.16a through 8.16c, are better than random for all techniques. The application of the Box-Cox transformation improves classification results using the full-dimension data, but not using the RBF-derived features.

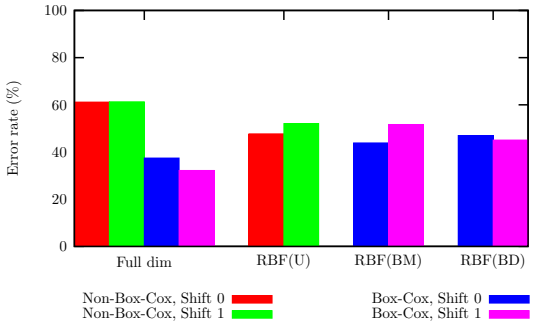
MBT4, in figures 8.16d, 8.16e, and 8.16i, has error rates worse than random when the Box-Cox transformation is not applied. When the Box-Cox transformation is applied, the error rate drops to better than random both for classification performed in the full-dimension space and classification performed using the RBF-derived features (RBF(BD)).

MBT3, in figures 8.16n through 8.16q, has error rates at or above the level of random chance when the Box-Cox transformation is not applied. When the Box-Cox transformation is applied, error rates fall to below random both for the full-dimension and RBF-feature-based (RBF(BD)) classifiers.

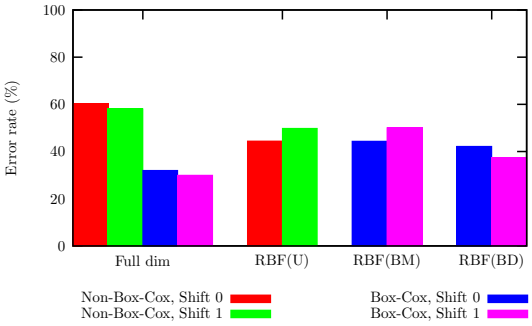
Results for the other data sets are approximately at, or above, the level of random chance. Broadly, for this group, if the application of the Box-Cox transformation results in a reduction in the error rate for a given data set then the RBF-derived features trained using the Box-Cox metric (RBF(BM)) fail to reflect this. The evidence is that the RBF(BM) error rates are closer to the RBF(U) error rates than the RBF(BD) error rates.



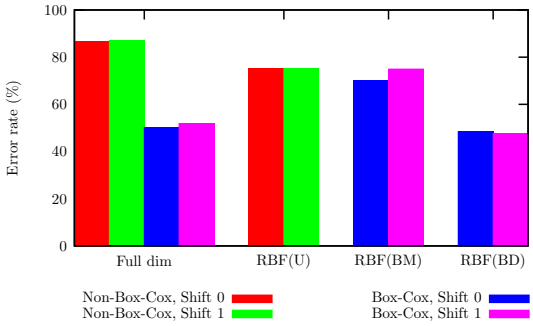
(a) *MBT2-4*



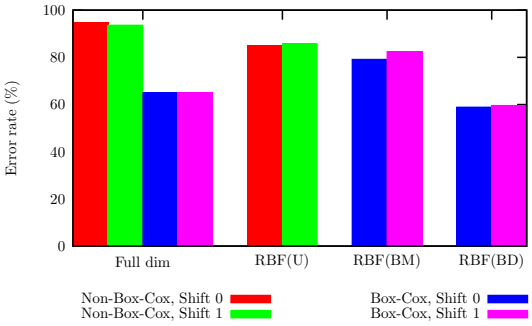
(b) *MBT2-5*



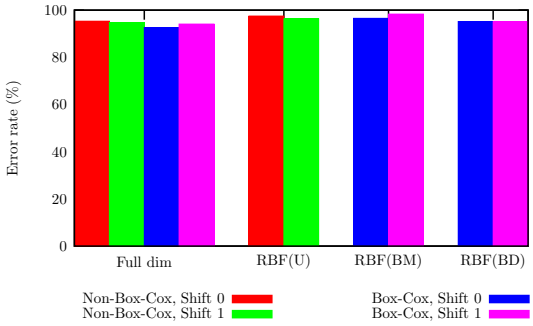
(c) *MBT2-6*



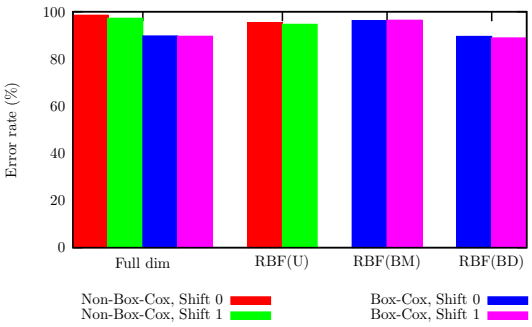
(d) *MBT4-1*



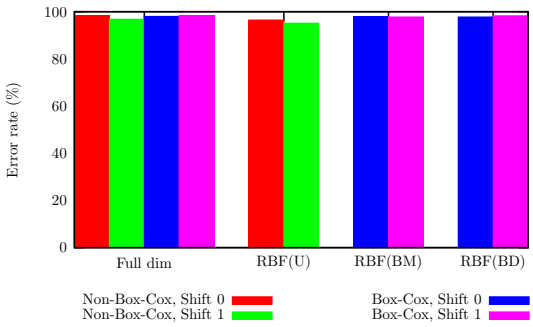
(e) *MBT4-2*



(f) *MBT5-4*

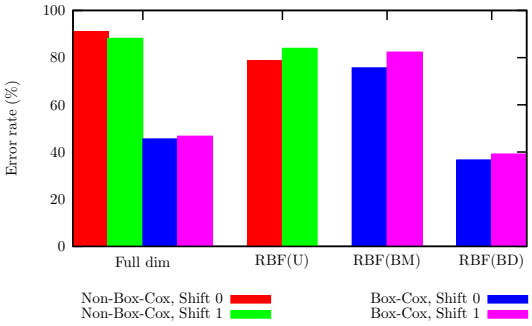


(g) *MBT5-5*

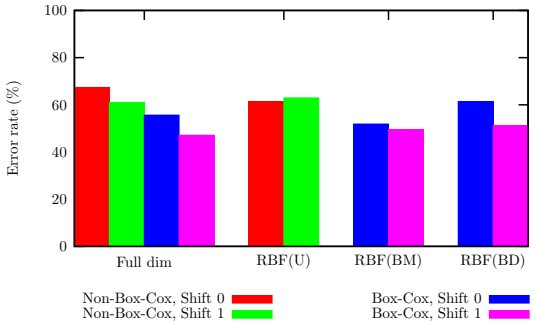


(h) *MBT5-6*

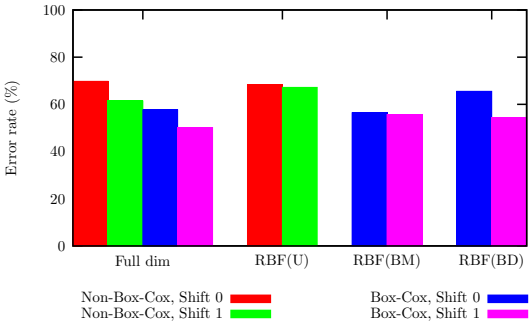
Figure 8.16: *Group 4, comparison of Box-Cox results. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, and RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. All projections were to 20 dimensions.*



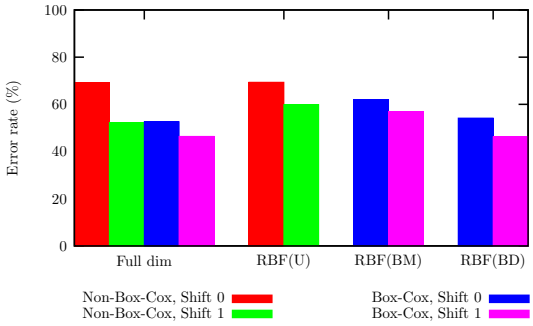
(i) *MBT4-4*



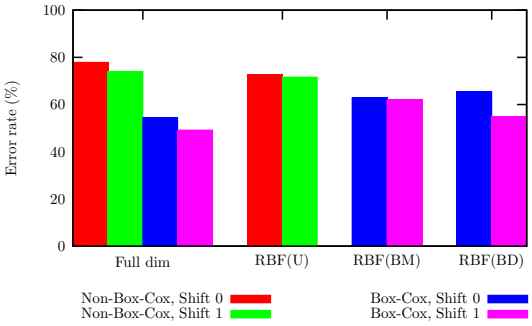
(j) *ADU3-2*



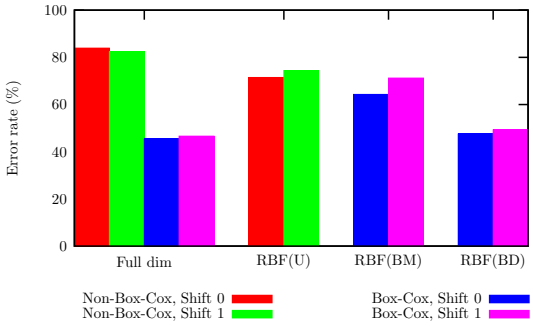
(k) *ADU3-3*



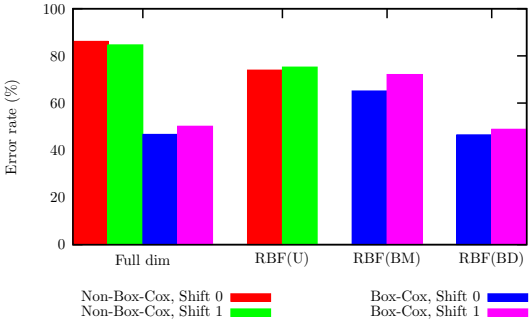
(l) *ADU3-4*



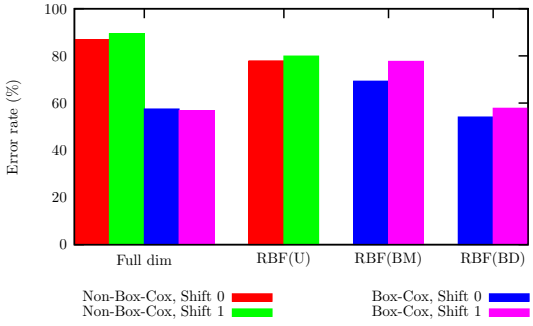
(m) *ADU3-5*



(n) *MBT3-7*

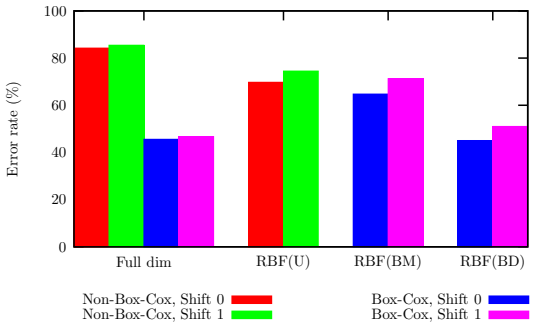


(o) *MBT3-8*



(p) *MBT3-9*

Figure 8.16: *Group 4, comparison of Box-Cox results. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, and RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. All projections were to 20 dimensions (cont.)*



(q) *MBT3-10*

Figure 8.16: *Group 4, comparison of Box-Cox results. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, and RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. All projections were to 20 dimensions (cont.)*

8.4.5 Group 5

The summary of results for data sets comprising Group 5 is shown in figure 8.17.

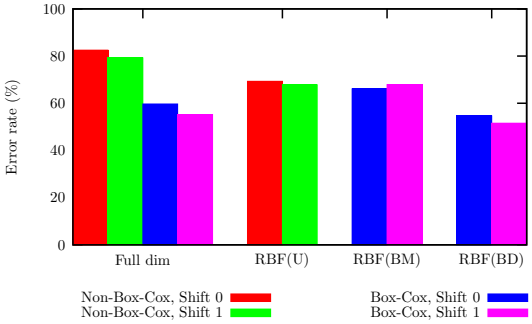
As with Groups 3 and 4, data sets of MBT3 (figures 8.17a–8.17d) showed some improvement upon the application of the Box-Cox transformation. Without the Box-Cox transformation, error rates are at approximately the level of random chance. With the transformation, they become better than random. The error rates for classification performed on full-dimension data after application of the Box-Cox transformation and for classification performed on RBF-derived features after application of the Box-Cox transformation (RBF(BD)) are similar for all MBT3 data sets.

All error rates for MBT5 (figures 8.17e and 8.17f) are much worse than random, even when the Box-Cox transformation is applied.

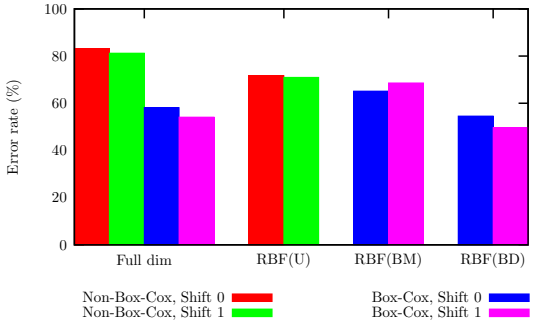
8.5 Summary and conclusions

Using the small training set, and when test vehicle types are represented during training (i.e. Groups 1 and 2), the SVM provids the best classification results, usually even better than using 1-NNR with the full-dimension data. The techniques that attempt to extract features that maximize separation of the classes do not seem to generalize well across configuration changes. The RBFN with supervision, RBF(S), generalizes the best out of the RBF variants. The SVM demonstrates good robustness to the position of the vehicle within the image as it generally performs at least as well as the full-dimension data with shifting.

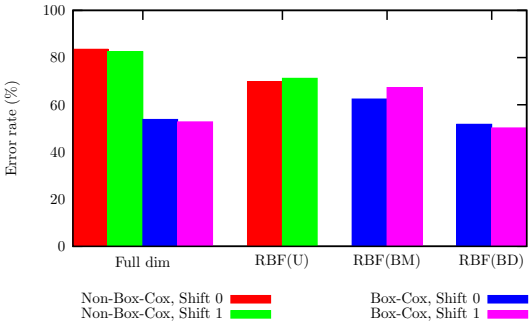
When vehicles are not represented during training, no method can really be said to



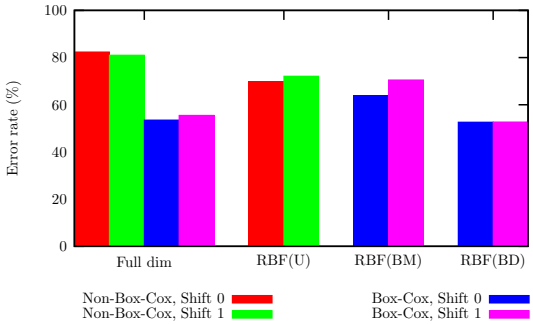
(a) *MBT3-1*



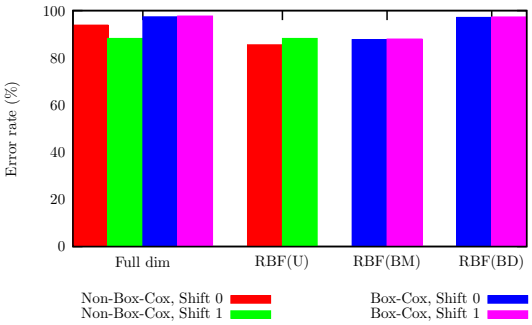
(b) *MBT3-2*



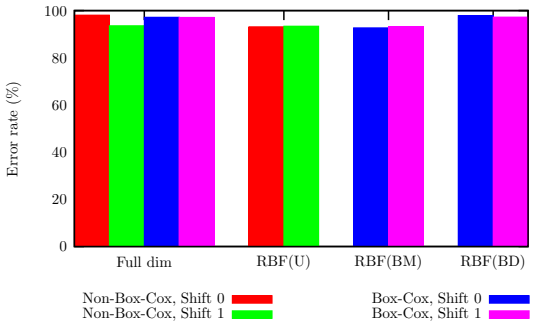
(c) *MBT3-4*



(d) *MBT3-5*



(e) *MBT5-1*



(f) *MBT5-2*

Figure 8.17: *Group 5, comparison of Box-Cox results. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, and RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. All projections were to 20 dimensions.*

perform well. Two exceptions are LDA and especially SVM against ADU2 and ADU3 (Groups 3–5). Although ADU3 is the same vehicle type as ADU1 (used in training) and so understandable in that light, ADU2 remains an anomaly. An offset between the vehicle positions of ADU1 and ADU3 is a possible explanation of the performances against ADU3 in Groups 3 and 4, especially given the previous conclusion that the SVM is robust to offsets of vehicle position.

Using the extended training set, when vehicles were previously represented during training (Groups 1 and 2), error rates for all feature-based techniques, including SVM, typically rise. The SVM does, however, remain the best feature-based method. The full-dimension results are the most consistent between the two training-sets, demonstrating that the full-dimension classifier is still generally able to pick out the correct match despite the introduction of new (different vehicle-type) class exemplars.

The corollary of this is that it seems to be feature extraction that has a problem when a class is represented by multiple, different vehicle types. This further reinforces a conclusion of the previous chapter that it is probably much better to have one class per vehicle type, rather than generic classes containing different vehicle types.

For Groups 3–5, previously exclusively comprising vehicle types not present during training, the addition of MBT2 and APC2 to the training data leads to large reductions in error rates on those vehicle types for both the full-dimension classifier and the SVM. It typically does not benefit the RBF-feature-based classifier. Although the RBF(R) error rate does drop somewhat for APC2, disappointingly there is no clear benefit to the MBT2 results. Thus, even though the RBF(R) is trained to maximize class separability relative to within-class distances, it does not seem to cancel out the unwanted difference(s) between MBT1 and MBT2.

It seems that the presence of a vehicle type when training the classifier is crucial to achieving good performance. Including a vehicle type when training the feature extraction stage, but then omitting it from the set of classifier exemplars in the hope that the feature space will compensate does not seem to bear fruit.

The Box-Cox transformation is very beneficial to classification performance, but the RBFN does not demonstrate that it learns to perform the Box-Cox transformation as part of the nonlinear feature-extraction transformation. Performing an explicit Box-Cox transformation essentially defines the transformation for all of input space and thereby, by definition, generalizes the transformation across all input space for all novel data points. In contrast, attempting to train the RBFN to perform the Box-Cox transformation results in it only being defined, at best, within the subspace of input space spanned by the training data. After an explicit Box-Cox transformation applied to the input data, the RBFN performs an efficient reduction of dimensionality and retains the level of classification

performance achieved with the full-dimension data.

Chapter 9

Summary

This thesis introduced the problem at hand as being the classification of radar (ISAR) images of military vehicles. Specifically, the aim was to improve generalization capability in the classification of the images. Particular challenges in this problem domain include the wide variety of vehicle types, the wide variety of configurations of a given vehicle type, and the difficulty of obtaining measurements representative of all such variations. The primary goal of this work was to investigate a technique for performing feature extraction, essentially projecting the high-dimension image data to a lower-dimension feature space, to compensate for differences between training and testing conditions.

A progression of techniques for performing data-driven feature extraction was presented in chapter 2. This started with some common linear methods, particularly PCA, before describing approaches to generalize PCA to include nonlinear transformations of the data. Limitations of such approaches were discussed, such as the difficulty applying them to novel measurements. Neural networks, with their property of universal approximation were introduced. The chapter concluded with a technique for performing MDS using an RBFN. The latter technique was adopted for use in this thesis for the reasons outlined in section 2.8.

The theory behind the chosen RBFN was described in chapter 3 along with schemes for modifying the transformation learned by the network. These modifications centred around the variation of which distances between points were important and how the distances between points were calculated.

Chapter 4 outlined the aim of the research and design of the experiments and experimental method. This chapter also described the radar data used and how the test data were partitioned into different groups that reflected the extent of difference from training conditions. How the data were pre-processed prior to use was also described.

Chapter 5 gave baseline results obtained using two common linear methods of feature extraction — PCA and LDA. The effects of the different optimization criteria of these

two methods were shown in the projections of the training data to low-dimension feature space. The baseline classification results showed that generalizing to new vehicle types was a far harder problem than generalizing across configuration changes for a given vehicle. The inclusion of shifting the position of the vehicle within an image was shown to improve classification using the images, but was often detrimental to classification performed in feature space.

A method for choosing the model parameters of the RBFN was validated, and the distributions of the distances between the data points and their nearest RBF centres were shown to be useful indicators of the required RBF bandwidths. Also it was shown that, provided enough reasonably-placed centres were used and their bandwidths were large enough, the quality of the projection, as measured by the stress, was not critically dependent upon the chosen model parameter values. The final choice of model parameters was based on the stress calculated for the full (unaveraged) validation data set, which was shown to be similar to the stress calculated for the full training data. It was shown that teaching the RBFN that distances between points belonging to different classes should not be smaller than distances between points belonging to the same class lead to a feature space in which the separability of the training classes was visibly improved. Anecdotal evidence was provided in support of the notion that the RBFN was extracting physically meaningful features from the data.

Initial classification results using RBFN-features, presented in chapter 6, as in chapter 5, showed that classification was very hard when the test vehicles were not represented during training. Classification against vehicles present during training, but with configuration changes, was much easier. Furthermore, a plausible explanation for the rank ordering of results was offered under some conditions. Some improvement with an allowed shift of ± 1 pixels in the vehicle position was seen, but this was far from universal, and further shifting was seen to be detrimental. When the number of features (dimensionality of feature space) was severely limited, it was shown to be advantageous to train the RBFN to concentrate on inter-class structure only. This advantage was shown to fade as the number of available features for describing the data increased. Despite the favourable appearance of the separability of the classes when using the rescaled inter-class distance, as shown in chapter 5, this was shown not to generalize well across either vehicle configurations nor different vehicle types. The best results came from performing the Box-Cox transformation globally on the data and then simply training the RBFN with the unsupervised loss function and the Euclidean distance metric.

The application of an SVM to an illustration problem, to demonstrate its behaviour, and to the ISAR data was presented in chapter 7. With the availability of separate validation data, these were used to perform model selection and shown to yield models

as good as those obtained via cross-validation on the training data, yet with the speed advantage of not having to perform cross-validation. The decision boundaries obtained from the SVM were shown to be very close to optimal and the number of support vectors used was shown to be reasonably consistent with theoretical expectation. Classification performance was shown to be very good when testing on the same vehicles used during training, with one vehicle representing a single class. Performance was typically very poor when the test and training vehicles were different types. The addition of new vehicle types to the training data did improve classification performance against those newly-added vehicle types, however performance against the vehicles represented by the small training group dropped.

A comparison of the preceding techniques was then drawn together in chapter 8 under the following categories: classification performed using only the small training group comprising one vehicle per class, classification performed using the extended training group comprising two vehicles per class, and classification that in some manner used the Box-Cox transformation. Using the small training set, the SVM was shown to perform best out of all techniques, usually better than the 1-NNR with the full-dimension image data. The feature extraction algorithms trained to maximize the separation of the classes were seen to generalize worst. The RBFN trained to concentrate on inter-class structure, but without attempting to increase class separability, was shown to generalize best out of the RBFN variants. Classification performed in full-dimension image space responded best to the inclusion of shifting, yet the SVM did not have shifting explicitly included and performed at least as well. This was taken to indicate that the SVM was robust, at least to some degree, to the exact positioning of the vehicles within the image frame. Error rates were shown to rise against the vehicle types represented in the original, small training set, upon the inclusion of additional representatives of a generic class when classification was performed using extracted features. In agreement with previous research, the application of the Box-Cox transformation was shown to be very beneficial to classification performance of the radar images. Furthermore, this was best applied as a preprocessing step rather than learned as part of the feature extraction stage.

Thus, the main contributions of this thesis are as follows.

- We have taken an approach for performing data-driven, nonlinear feature extraction and applied it to a substantial, real-world, varied dataset of ISAR images of military vehicles. A justifiable and interpretable figure of merit, 1-NNR classification performance, was used to assess the features for their ability to generalize across certain changes in conditions between training and testing.

- Of the two types of such changes in operating conditions, namely different configurations for a given vehicle type and different vehicle types, the former was shown to

be relatively solvable whilst the latter remained to be resolved to any satisfaction.

- Despite being able to capture structure present in the training data, no method of training the RBFN was able to extract features that satisfactorily generalized the concept of 'MBT' or 'APC' etc. to new vehicle types nominally belonging to those 'classes'.

- The Box-Cox transformation was confirmed to be highly beneficial for the classification of radar images. Furthermore, it could not be left to a data-driven transformation to learn, it had to be 'hard coded' by being defined explicitly as a preprocessing stage applied to the data.

- Despite being closely related to the RBFN, the SVM is a theoretically very well motivated 'state of the art' classifier, as opposed to just performing feature extraction: it is optimized from start to finish as a classifier. As such it was shown to outperform the two-stage method of using the RBFN for feature-extraction followed by a 1-NNR classifier.

Chapter 10

Conclusions

A number of significant conclusions can be stated, which have informative implications for the design of classifiers for ATR.

10.1 Sensitivity to changes in vehicle configuration

One concern motivating this work was that radar target signatures are sensitive to small changes in target configuration [7, 39], with the implication that this will be highly detrimental to robust classification. Results presented in this thesis, sections 8.2.1 and 8.2.2 in particular, have shown that a large degree of robustness to such configuration changes is readily achievable. This is highlighted in the summary over configurations for each of the vehicles in Groups 1 and 2 for the full-dimensionality and SVM classifiers shown in table 10.1. A low overall error rate is achieved, especially by the SVM.

10.2 Robustness to localization of target in image

The improvement in the full-dimension classification results with shifting indicated that there was a lack of alignment between the vehicles in the training and testing data.

Table 10.1: *Error rates averaged over configurations for vehicles MBT1, APC1, and ADU1 using the small training set. Full dim denotes classification performed in the full-dimension measurement space, allowing a shift of* ± 1 *pixels to improve vehicle alignment. SVM denotes SVM classification. Numbers in parentheses denote combined sample sizes.*

	Error rate (%)	
Vehicle	Full dim	SVM
MBT1 (6507)	10.7	5.2
APC1 (2463)	2.4	3.1
ADU1 (5566)	13.8	0.3

Table 10.2: *Error rates averaged over configurations for vehicles MBT1, APC1, and ADU1 using the small training set. Results are given without shifting (0) and with shifting (±1). Full dim denotes classification performed in the full-dimension measurement space, LDA denotes classification performed in LDA feature space, and RBF(R) denotes classification performed in feature space derived using rescaled inter-class distances. Numbers in parentheses denote combined sample sizes.*

	Error rate (%)					
	Full dim		LDA		RBF(R)	
Vehicle	0	±1	0	±1	0	±1
MBT1 (6507)	20.0	10.7	23.0	26.1	25.6	32.0
APC1 (2463)	3.1	2.4	19.5	22.7	16.1	18.4
ADU1 (5566)	17.8	13.8	12.7	15.7	14.4	21.8

For the full-dimension experiments, the shifting improved the alignment, and hence the match, between the training and testing images. The results using extracted features, however, indicated that the feature extraction was often not robust to vehicle position within an image. This was seen for many data sets in sections 8.2.1 and 8.2.2. The effect is highlighted in table 10.2 by averaging over configurations for classification performed in full-dimensional measurement space, LDA feature space, and the RBF feature space derived using rescaled inter-class distances. The LDA and RBF(R) features were particularly adversely affected by shifting.

10.3 Feature extraction for generalization

Extracting features that sought to increase the separability between different classes, generalized most poorly across changing vehicle configurations, as seen for LDA and RBF(R) in sections 8.2.1 and 8.2.2 and summarized in table 10.2. The best RBF approach was typically RBF(S) that simply attempted to model the distances between points of different classes. This is highlighted by the summary in table 10.3, which shows error rates averaged over configuration changes for the vehicles in Groups 1 and 2 when trained using the small training set. RBF(S) feature space performed about as well as classification using full dimension measurement space. However, it performed considerably better overall on ADU1. The PCA summary results are similar to those for full-dimension with the exception of the MBT1 results, which are are considerably worse.

Table 10.3: *Error rates averaged over configurations for vehicles MBT1, APC1, and ADU1 using the small training set. Results include shift of ±1 pixels. RBF(U) denotes RBF trained using the unsupervised loss function and RBF(S) using the supervised loss function (inter-class distances). Numbers in parentheses denote combined sample sizes.*

	Error rate (%)			
Vehicle	Full dim	PCA	RBF(U)	RBF(S)
MBT1 (6507)	10.7	15.0	12.6	10.8
APC1 (2463)	2.4	3.1	4.3	3.6
ADU1 (5566)	13.8	13.7	13.1	10.7

Table 10.4: *Error rates averaged over configurations for vehicles MBT1, APC1, and ADU1 for the small and the extended training sets. Classification was performed in full-dimension space (Full dim), LDA feature space, feature space obtained by the RBF trained with the supervised loss function (RBF(S)), and feature space obtained by the RBF trained with rescaled inter-class distances (RBF(R)). Results include shift of ±1 pixels.*

	Error rate (%)							
	Full dim		LDA		RBF(S)		RBF(R)	
Vehicle	Small	Ext.	Small	Ext.	Small	Ext.	Small	Ext.
MBT1 (6507)	10.7	10.7	26.1	57.2	10.8	13.6	32.0	36.9
APC1 (2463)	2.4	2.6	22.7	46.8	3.6	5.0	18.4	16.6
ADU1 (5566)	13.8	16.0	15.7	10.3	10.7	17.2	21.8	34.3

10.4 Extending the vehicle types represented during training

Except for classification in full dimension space, the inclusion of extra (different) training vehicles often caused a rise in error rates for vehicle types that were previously the sole representatives of their class, this is demonstrated in sections 8.3.1 and 8.3.2 and summarized in table 10.4 by averaging over configuration changes. The summarized results show an increased error rate in the majority of cases. The least increase is for classification performed in the full-dimension measurement space. Thus, attempting to widen the definition of a class had negative connotations for existing members (representatives) of the class when extracting features for classification. The use of images as templates for the 1-NNR classifier was much more robust. There was, generally, a decrease in error rate on the vehicles newly represented within a class, seen in sections 8.3.3 and 8.3.4. The relevant vehicles in these groups are MBT2 and APC2. The results for these vehicles, for the small and extended training sets, are summarized by averaging over configurations in table 10.5. The linear (LDA) features performed poorly with the small training set and got worse with the addition of the extra data. The best performance was obtained using the full-dimension images, which also benefited most from the additional training

Table 10.5: *Error rates averaged over configurations for vehicles MBT2 and APC2 for the small and extended training sets. LDA denotes classification in LDA feature space, RBF(S) denotes classification in feature space derived using the supervised RBF loss, and RBF(R) denotes classification in RBF feature space derived using the rescaled inter-class distances. Results include shift of ±1 pixels.*

	Error rate (%)							
	Full dim		LDA		RBF(S)		RBF(R)	
Vehicle	Small	Ext.	Small	Ext.	Small	Ext.	Small	Ext.
MBT2 (5556)	57.6	18.6	74.2	83.4	46.0	45.4	51.1	57.8
APC2 (2499)	48.2	3.1	79.0	96.2	72.9	56.5	75.1	30.0

Table 10.6: *SVM error rates for MBT2 and APC2 averaged over configurations for the small and extended (Ext.) training sets. Numbers in parentheses are the combined sample sizes.*

	SVM error rate (%)	
Vehicle	Small	Ext.
MBT2 (5556)	70.8	34.6
APC2 (2499)	98.2	4.2

vehicles.

10.5 Represented versus unrepresented target types during training

When a target was represented during training, different articulations or configurations of the target were found to be a relatively easy problem for classification. Under such conditions, found in sections 8.2.1 and 8.2.2, the SVM, with its well-motivated theoretical background was very much the best performing technique (table 10.1). However, when targets were not represented during training, as found in sections 8.2.3–8.2.5, and 8.3.3–8.3.5, no technique was found to work well (e.g. table 10.5 and the first column of table 10.6). It is acknowledged that ADU2 and ADU3 were exceptions to this rule, however ADU3 is, strictly speaking, the same vehicle type as represented during training by ADU1. ADU2 remains an anomaly. Table 10.6 confirms the improvement in classification performance achieved by the SVM when MBT2 and APC2 are added during classifier training.

Table 10.7: *Error rates averaged over configuration changes for experiments using the Box-Cox transformation. Full dim denotes classification performed in measurement space after applying the Box-Cox transformation. RBF(BM) denotes classification using features obtained using the RBF trained with Box-Cox distances. RBF(BD) denotes classification performed using features calculated after the Box-Cox transformation was applied to all data. Results for 0 shift and* ± 1 *shift are given. Numbers in parentheses denote combined sample sizes.*

	Error rate (%)					
	Full dim		RBF(BM)		RBF(BD)	
Vehicle	0	± 1	0	± 1	0	± 1
MBT1 (6507)	6.5	2.4	13.3	11.8	7.4	4.9
APC1 (2463)	2.9	2.4	6.5	4.5	2.5	2.4
ADU1 (5566)	1.4	1.3	10.2	9.3	1.5	1.0

10.6 Box-Cox transformation

The Box-Cox transformation was generally beneficial to classification, especially when the test vehicle was represented during training. The attempt to get the RBFN to learn the Box-Cox transformation as part of the feature-extraction transformation did not work. It was clear that performing the Box-Cox transformation globally, on all input data, was required to realize its benefits. These results were given in section 8.4. Table 10.7 averages the error rates for MBT1, APC1, and ADU1 over configuration changes and using the small training set. The small change in error rate with shifting shows that the Box-Cox transformation renders classification much less sensitive to the localization of the vehicle in the image.

Chapter 11

Further work

The research presented herein centred on the classification of ISAR images of military vehicles. The data were all processed to the same resolution and the local background was benign. These qualities allowed the investigation to focus on the effect of changes in vehicle configuration and the effect of presenting the classifier with previously unseen vehicle types. These qualities are not, however, representative of the conditions under which operational ATR will be performed. For an operational system, the background clutter local to any targets will be much brighter and more variable than for turntable ISAR. The achievable resolution will be limited by the available dwell time. Particularly for an imaging missile-sensor, the achievable resolution will also vary with the angle subtended between the missile's velocity vector and the line-of-sight to the target.

An operational ATR system will typically have to perform the following steps:

1. Form an image;
2. Detect possible targets within the image;
3. Extract the detected objects from their local background;
4. Classify the detected objects.

Classification requires representative training data and this is difficult to obtain for operational sensors. It means the sensor must be flown past the target from many different directions. This is expensive and time-consuming. It can also be difficult to arrange in United Kingdom airspace. ISAR measurements are a much more cost-effective and rapid method of obtaining radar images of military vehicles.

Computer simulated radar images of vehicles are even easier to obtain than ISAR measurements as they do not require an example vehicle. Modelling software and detailed design information about a vehicle are all that are required to generate them. Approximations made in such software and lack of some physical detail in the models, for example

surface roughness, mean that the simulations can only model certain aspects of the radar signature of a vehicle. Thus, in order of increasing relevance to an operational sensor and descending ease of gathering, sources of training data for the classification of vehicles in an operational ATR system are:

1. Simulation;
2. ISAR;
3. Operational sensor.

A key result of this thesis has been to show that when a vehicle to be classified is present during the training of a classifier, changes in the configuration of the vehicle have only a relatively small effect on classification performance. On the other hand, the correct classification of a previously unseen vehicle type is an unrealistic goal.

Thus the major area for future study is not the generalization of classification performance across configuration or vehicle change, but generalization across sensor or sensor-parameter change. Specific goals should be to be able to train a classifier on simulated or ISAR images and successfully classify targets observed by an operational sensor, and for performance to be robust to changes in imaging resolution.

The limited availability of data from a flying platform will remain a challenge. However, there is much that can be done using simulated and ISAR data. Techniques for generalizing across different sensors should be developed, at least initially, by combining these two sources of data. ISAR data should be processed to varying resolutions to generate data sets for use in developing techniques that are robust to resolution. Specifically, the same measured data should be processed to different resolutions. There will, of course, be limits to how robust classification performance can be with respect to resolution: as the resolution degrades so will classification accuracy. Any further programme of work should include a study of the point at which classification becomes unfeasible as resolution degrades.

The images obtained by simulation and ISAR at the same resolution should ideally occupy a common subspace of measurement space. The images of differing resolutions from the same sensor can be viewed as occupying subspaces within some higher-dimension embedding space. Thus it is suggested that such subspace methods as considered in this thesis are promising avenues to pursue. This thesis highlighted the sensitivity of the feature extraction to the position of the vehicle within the image and this should be addressed in future work. This will require either better techniques for locating the vehicle within an image or a feature extraction stage that is robust to vehicle location.

Possible techniques for localizing the target within an image are Active Shape Models and Active Appearance Models [19, 20, 22, 21]. These models can only deform in ways

characteristic of the classes of objects they represent. They offer the prospect of locating an object within an image despite changes to the configuration of the object because allowable configuration changes can be built into the model. Furthermore, an interesting consequence of prescribing models for the classes is that vehicles that have not been represented during training can be excluded from subsequent classification. This would reduce spurious incorrect classifications arising from unrealistic attempts to classify previously unseen vehicle classes. Such models should be considered as part of an ATR processing chain.

Another requirement of operational ATR will be the flexible inclusion of costs. Flexibility is necessary to reflect the different aims of different missions. A mission to eliminate enemy air-defences, for example, will regard it preferable to misclassify MBTs as ADUs rather than to miss ADUs. However, the extreme case of wasting resources attacking secondary targets is to be avoided. Thus, an operational scheme for ATR will need to incorporate the cost of the resources used and the cost of misclassifications. These costs will change from mission to mission and expert military input will be vital for classifier design and training. Data fusion should also be incorporated to reflect all available information, for example heavy armoured vehicles are less likely to be found in areas of boggy ground, recent intelligence about a target's position will provide constraints on the target's current position.

The primary recommendation of this thesis for a major area of work that is necessary to achieve robust ATR is that a study should be conducted to investigate techniques for generalizing across different sensors and sensor parameters (e.g. resolution). This is because of the importance of providing training data and the difficulty of obtaining such data using flying sensors. Success will provide a key component to the formation of a robust operational ATR system.

Appendix A

Images of vehicles measured

A.1 Vehicles measured in 1997

This section contains photographs of vehicles measured during the 1997 radar trial, namely:

- 432 (APC1)
- Challenger (MBT4)
- Stormer (ADU2)
- ZSU-23-4 (ADU3).

Images of the T-72 (MBT3) are unavailable, as are images of MBT5.

An image of the 432 (APC1) vehicle is shown in figure A.1. This is a relatively simple shape. There is some detail on the upper surface, but nothing large like a gun turret.

An image of the Challenger (MBT4) vehicle is shown in figure A.2. The photograph was taken with the vehicle on the turntable used to rotate vehicles for ISAR imaging.

Figure A.1: *The 432 (APC2) vehicle pictured standing on gravel quarry floor.*

Figure A.2: *Challenger (MBT4) vehicle pictured on the turntable used to rotate the vehicles for ISAR imaging. The front of the vehicle is visible, with the driver's hatch located centrally. The gun turret is oriented at* 90° *to the hull.*

(a) *Front view*

(b) *Rear view*

Figure A.3: *Challenger (MBT4) vehicle pictured on the turntable used to rotate the vehicles for ISAR imaging. The vehicle is shown with cover in place over the turret, gun locked into gun crutch and fuel barrels attached to the rear.*

Noticeable features of this vehicle include its smooth lines and dominating turret. The coloured material on and around the turntable is radar-absorbant material (RAM) used to attenuate unwanted reflections from the vehicle via the ground. Such multipath reflections can create ghost radar images of the vehicle. The object just to the fore is a reference reflector. Being located on a short tripod it provides a phase reference for the radar.

The Challenger (MBT4) is also shown in figure A.3 with a cover over the turret, as measured for data set MBT4-4. The vehicle is pictured slightly from the front, figure A.3a, and slightly from the rear, figure A.3b, to show the extent of the cover. The vehicle is in its transit configuration, with the gun barrel at 180° relative to the hull and locked down into the gun crutch located between the two extra-range fuel tanks at the very rear of the vehicle. The gun crutch and the fuel tanks are visible in figure A.3b.

Figure A.4: *Stormer (ADU2) located standing on the turntable with RAM beneath.*

Figure A.5: *ZSU-23-4 (ADU3) pictured standing on gravel quarry floor with radar up. The front of the vehicle is on the right in the scene, with the turret at an angle of around* 225° *relative to the hull.*

Shown in figure A.4 is the Stormer (ADU2). The surface-to-air missile launcher is clearly visible on the top rear of the vehicle. Another noticeable feature of the vehicle shape is the sloping front.

The ZSU-23-4 (ADU3) measured during the 1997 trial is shown pictured in figure A.5. The image shows the vehicle with its radar raised. The turret is very much a dominant part of the vehicle's shape.

A.2 Vehicles measured in 2000

This section contains photographs of vehicles measured during the 2000 radar trial, namely:

- BMP (APC1)
- T-55 (MBT2)

- T-62 (MBT1)
- ZSU-23-4 (ADU1).

The BMP (APC1) is shown in figure A.6. One prominent characteristic of this vehicle is the relatively smooth upper surface. Another is the smallness of the turret. An image of the rear of the vehicle is given in figure A.7. From this perspective the rear hatches can be clearly seen, with one in a partially-open position.

The T-55 (MBT2) is shown in figure A.8. Typically for an MBT, the turret forms a prominent part of the vehicle's structure. The toolboxes can be seen down the near-side of the hull, on the upper surface.

The T-62 (MBT1) is shown in figure A.9 with camouflage netting draped over. The camouflage netting is intended only to disguise the vehicle's visible features to the human eye, the netting is not designed specifically to counter a radar sensor but will still impact on the vehicle's radar signature. Increasingly, as radar becomes a greater and greater threat to vehicles, effective camouflage against that sensor can be expected to be more widely deployed. However, radar-specific netting was not available when the trials were conducted. Furthermore, if truly effective camouflage rendered the vehicle invisible to the radar, then the radar would not be measuring the vehicle and classification would be impossible: *a priori* information about the vehicle would be necessary. Indeed, just about anything can be camouflaged [72], and if the camouflage is effective then the object being hidden could be just about anything. The vehicle is pictured again in figure A.10 without the camouflage netting. This image also shows clearly the snorkel, hinged from the upper surface of the turret, laying down. The vehicle can be seen with the snorkel raised in figure A.11.

The ZSU-23-4 (ADU1) is shown in figure A.12. The figure shows the radar lowered in its 'down' position. The vehicle appears visually identical to the same type imaged for

Figure A.6: *BMP (APC1) pictured standing on gravel quarry floor. The front of the vehicle is to the fore in the scene.*

Figure A.7: *BMP (APC1) pictured from rear, standing on gravel quarry floor. One of the rear hatches is partially open.*

Figure A.8: *T-55 (MBT2) pictured standing on gravel quarry floor. The vehicle is facing to the left.*

Figure A.9: *T-62 (MBT1) with camouflage netting and pictured on the turntable with RAM in place.*

Figure A.10: *T-62 (MBT1) shown parked on gravel quarry floor, facing to the right, and with the snorkel laying down.*

(a) *Left hand side: toolboxes mounted along the side, above the tracks.*

(b) *Right hand side: fuel cans mounted along the side, above the tracks.*

Figure A.11: *T-62 (MBT1) pictured on the turntable with RAM in place. The snorkel is in its raised position.*

Figure A.12: *ZSU-23-4 (ADU1) pictured parked on gravel quarry floor. The vehicle is facing left and the radar is in its lowered position.*

the 1997 trial, shown in figure A.5.

Appendix B

Supplemental results

B.1 RBFN classification results for Group 2 data.

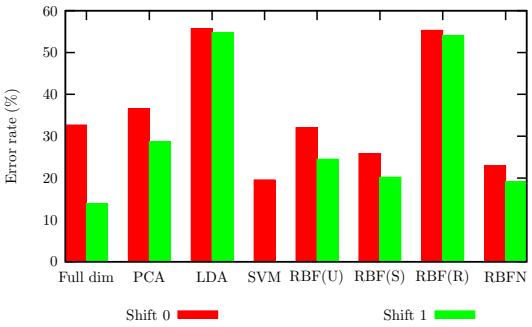
To supplement the results given for Group 1, in figure 8.7, results for Group 2 are included here in figure B.1. In agreement with the results for Group 1, there is a great deal of similarity between the SVM and RBFN results relative to the other techniques.

B.2 An alternative kernel for learning the Box-Cox transformation.

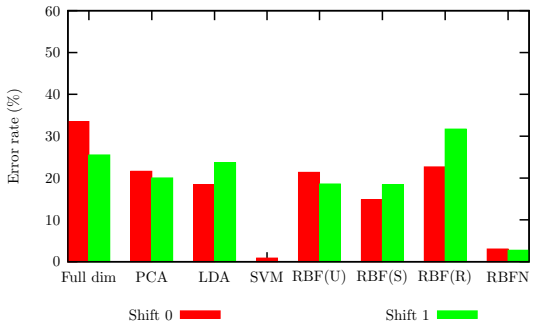
In section 3.2.3 an alternative to the Gaussian kernel was mentioned: the $z^2 \log(z)$ kernel. To see if this kernel was any better than the Gaussian for learning the Box-Cox transformation as a part of the feature extraction process, the experiment using the Box-Cox distance metric was repeated using this kernel. Model selection was performed in exactly the same way as for the Gaussian kernel as described in section 5.5.2.

The results for Group 1 are shown in figure B.2. This figure is the same as figure 8.13 but for the inclusion of the results using the $z^2 \log(z)$. The results using the $z^2 \log(z)$ kernel, denoted RBF($z^2 \log(z)$), are similar to those obtained using the Gaussian kernel, RBF(BM).

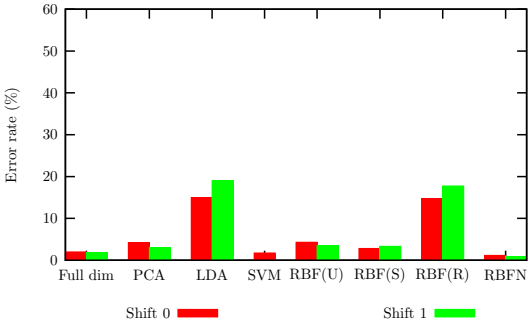
The results for Group 2 are shown in figure B.3. This figure is the same as figure 8.14 but for the inclusion of the results using the $z^2 \log(z)$. As seen with Group 1, the results using the $z^2 \log(z)$ kernel are similar to those obtained using the Gaussian kernel, RBF(BM). Thus the use of a different kernel achieved very similar results as, and certainly no better than, before.



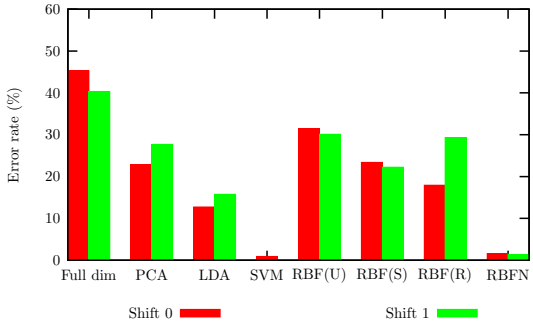
(a) *MBT1-2*



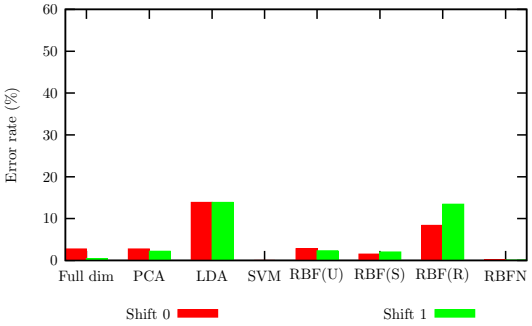
(b) *MBT1-6*



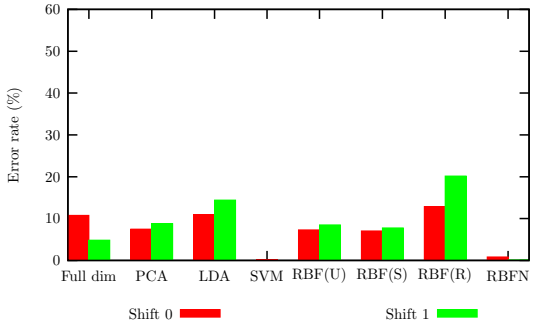
(c) *APC1-3*



(d) *ADU1-4*

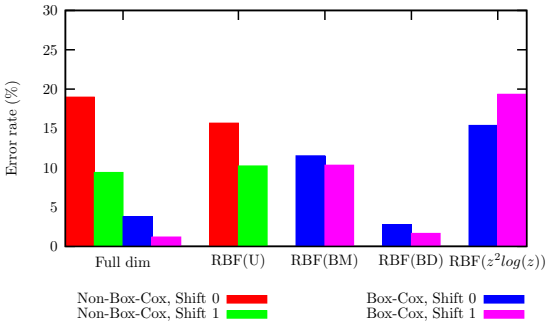


(e) *ADU1-6*

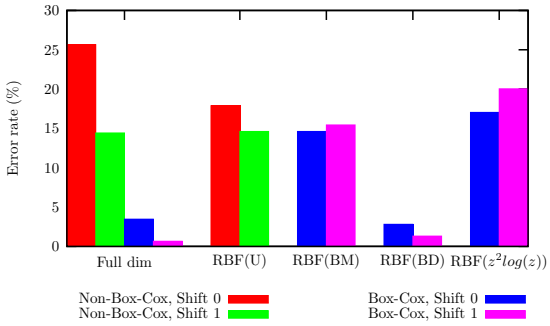


(f) *ADU1-7*

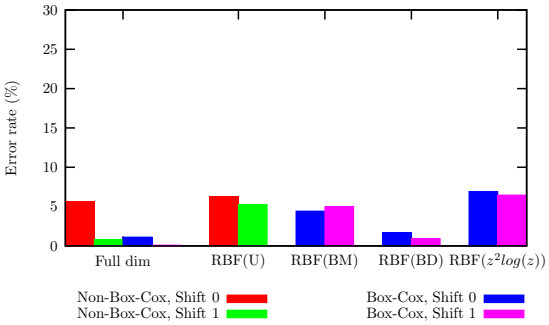
Figure B.1: *Group 2, comparison of the small-training-group results including the RBFN classifier. RBF(U) denotes RBF feature space obtained using the unsupervised loss function and projecting to 20 dimensions. RBF(S) denotes RBF feature space obtained using the supervised loss function with* $\rho = 0$ *and projecting to 20 dimensions. RBF(R) denotes RBF feature space obtained using the rescaled inter-class distances and projecting to 20 dimensions. RBFN denotes the RBFN used to perform classification.*



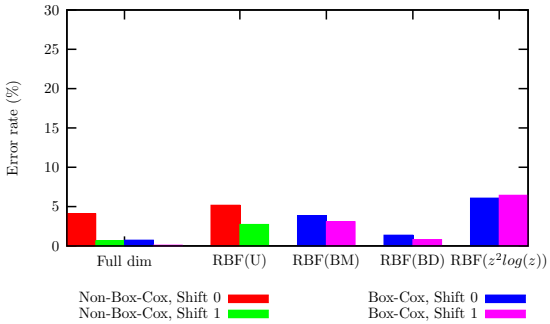
(a) *MBT1-3*



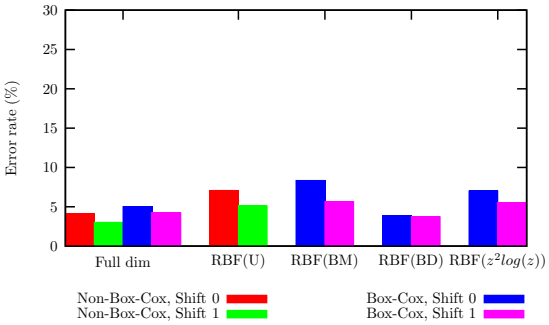
(b) *MBT1-4*



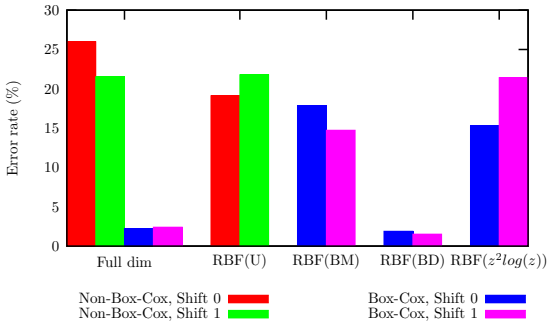
(c) *MBT1-5*



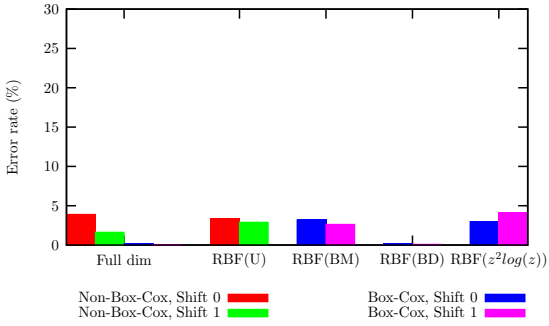
(d) *MBT1-7*



(e) *APC1-2*

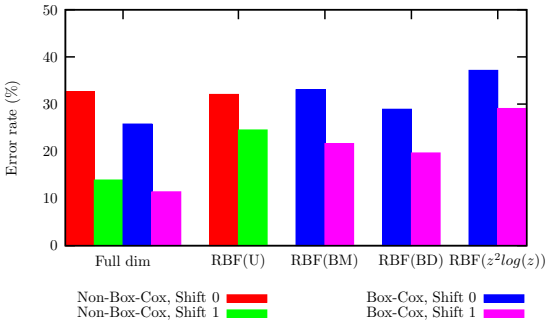


(f) *ADU1-3*

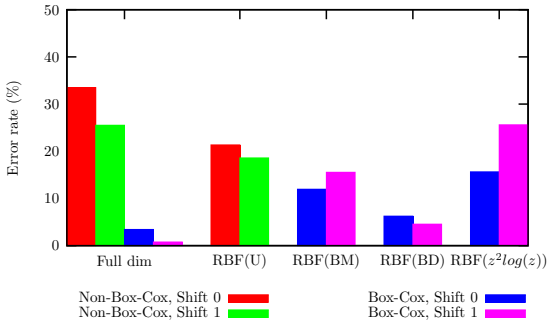


(g) *ADU1-5*

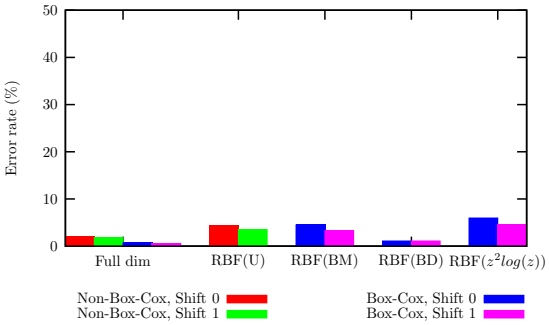
Figure B.2: *Group 1 comparison of Box-Cox results including* $z^2 \log(z)$ *kernel. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. RBF(*$z^2 \log(z)$*) is the analogue of RBF(BM) but for using the* $z^2 \log(z)$ *kernel. All projections were to 20 dimensions.*



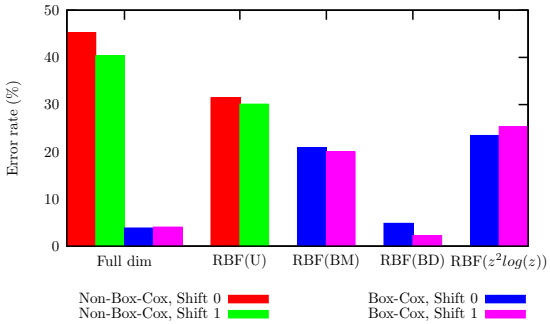
(a) *MBT1-2*



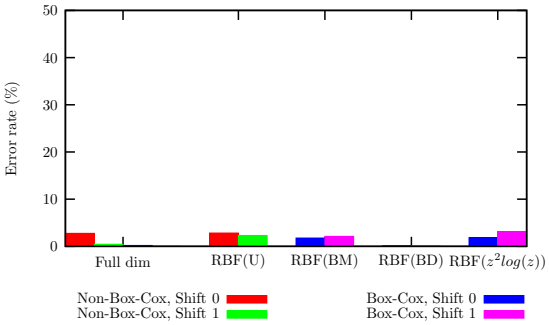
(b) *MBT1-6*



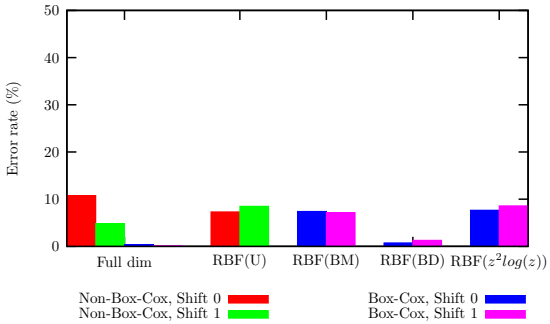
(c) *APC1-3*



(d) *ADU1-4*



(e) *ADU1-6*

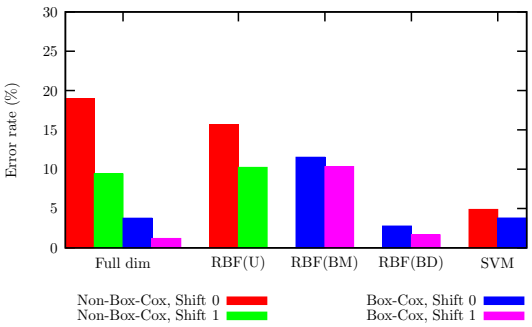


(f) *ADU1-7*

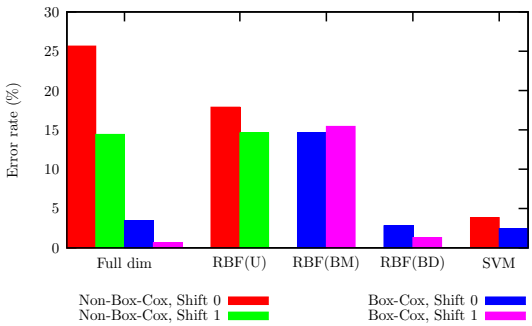
Figure B.3: *Group 2 comparison of Box-Cox results including* $z^2 \log(z)$ *kernel. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. RBF(*$z^2 \log(z)$*) is the analogue of RBF(BM) but for using the* $z^2 \log(z)$ *kernel. All projections were to 20 dimensions.*

B.3 SVM trained and tested on Box-Cox transformed data.

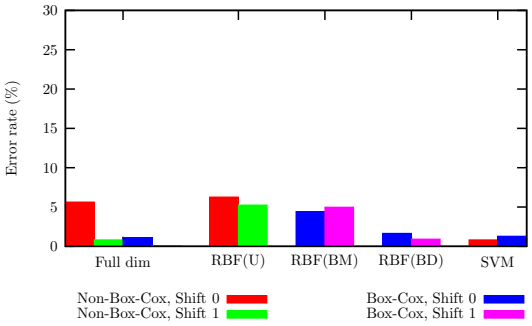
The response of the SVM to the Box-Cox transformation was not a major topic for investigation in this work. Nevertheless, for completeness, the SVM was also trained and tested on Box-Cox transformed data to see if the transformation gave rise to improved SVM performance. The results are given here for Group 1 and Group 2 in figures B.4 and B.5 respectively. The figures are the same as those in figures 8.13 and 8.14 respectively but with the addition of SVM results. Overall, the SVM using non-Box-Cox transformed data tended to perform at least as well as the other techniques when they used Box-Cox transformed data. Using Box-Cox transformed data for the SVM did not generally confer a further improvement to the SVM.



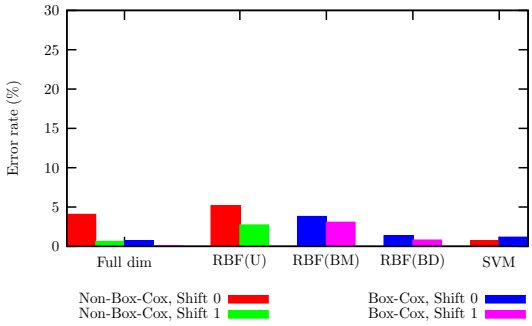
(a) *MBT1-3*



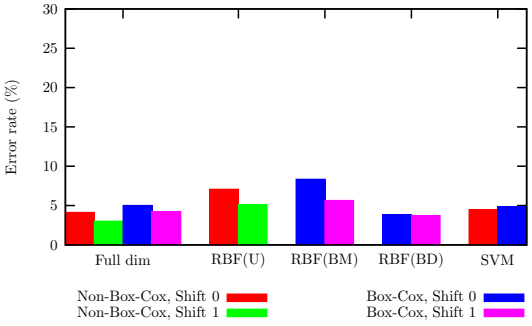
(b) *MBT1-4*



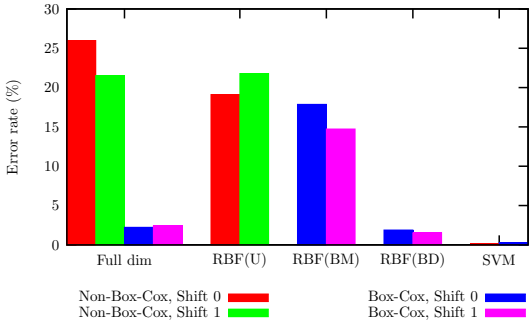
(c) *MBT1-5*



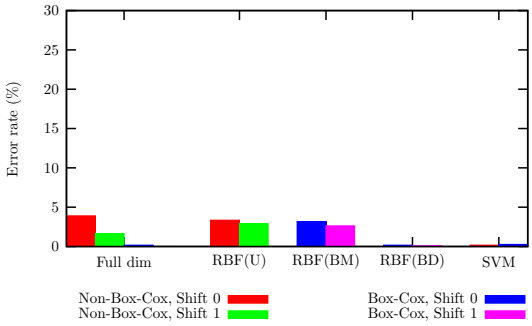
(d) *MBT1-7*



(e) *APC1-2*

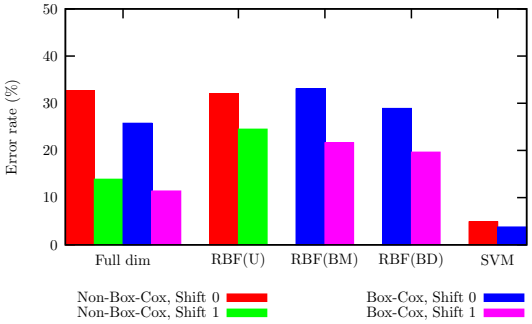


(f) *ADU1-3*

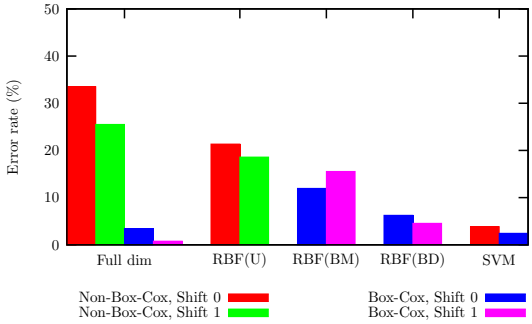


(g) *ADU1-5*

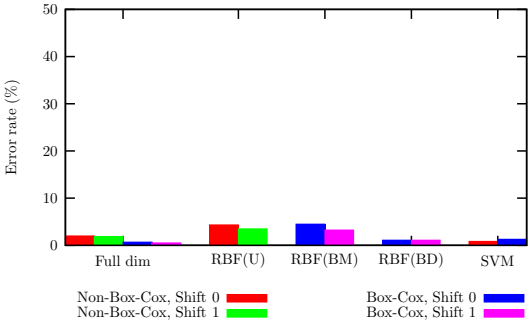
Figure B.4: *Group 1, comparison of Box-Cox results including SVM. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, and RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. All projections were to 20 dimensions.*



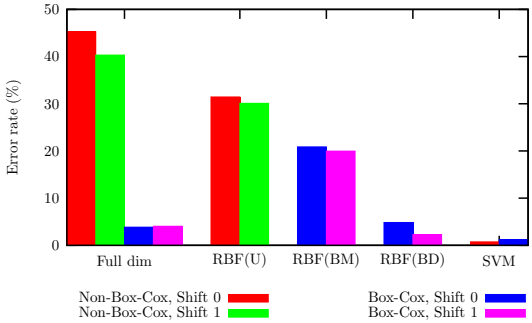
(a) *MBT1-2*



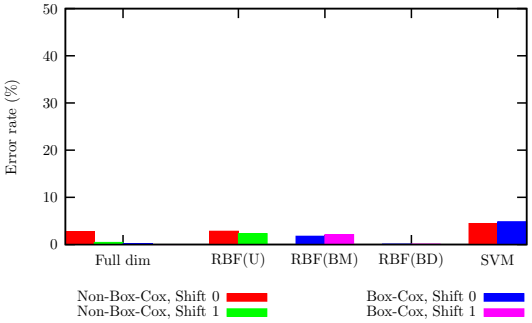
(b) *MBT1-6*



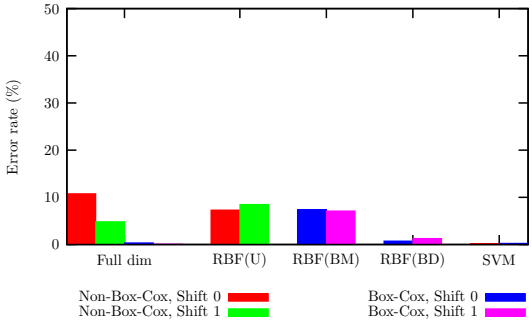
(c) *APC1-3*



(d) *ADU1-4*



(e) *ADU1-6*



(f) *ADU1-7*

Figure B.5: *Group 2, comparison of Box-Cox results including SVM. All RBF feature spaces were obtained using the unsupervised loss function. RBF(U) used the Euclidean metric to calculate distances, RBF(BM) used the Box-Cox metric* ($t = 0$) *to calculate distances, and RBF(BD) applied the Box-Cox transformation* ($t = 0$) *to all input data and then the Euclidean metric to calculate distances. All projections were to 20 dimensions.*

Appendix C

Publications

The problem being tackled was outlined, and the feature extraction approach was introduced, in a conference paper [7] for the International Society for Optical Engineering (SPIE) Aerosense 2000 conference held in Florida, USA. This paper also described a complementary Bayesian mixture model approach and preliminary results. Features extracted using the radial basis function network were compared with those from principal component analysis in a conference paper [77] for the SPIE Aerosense 2001 conference held in Florida, USA. The preceeding papers used a small data set processed to a lower resolution. Subsequent work used a new, much larger data set processed to a higher resolution. Features extracted using the supervised radial basis function network were compared with those from linear discriminant analysis in a conference paper [78] for the 2002 SPIE Aerosense conference held in Florida, USA. A comparison between classification performed on the full-dimension data and on features extracted using the supervised radial basis function network was presented in a conference paper [76] for the Institute of Electrical Engineers (IEE) Radar 2002 conference held in Scotland, UK. This paper placed emphasis on the computational saving achieved by performing classification on the extracted features.

References

[1] D. Anguita, S. Ridella, F. Rivieccio, and R. Zunino. Hyperparameter design criteria for support vector classifiers. *Neurocomputing*, 55:109–134, 2003.

[2] G.H. Bakir, L. Bottou, and J. Weston. Breaking SVM complexity with cross-training. In *Advances in Neural Information Processing Systems 17*, To be published.

[3] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.

[4] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1998.

[5] C.M. Bishop, M. Svensén, and C.K.I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[6] G.E.P. Box and D.R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 26(2):211 – 252, 1964.

[7] A. Britton, K.D. Copsey, G.T. Maskall, A.R. Webb, and K. West. Nonlinear feature extraction and bayesian mixture model approaches to target classification using MMW ISAR imagery: a preliminary study. In R. Trebits and J. Kurtz, editors, *Radar Sensor Technology V*, volume 4033, pages 145–156. SPIE, July 2000.

[8] D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2(3):321–355, 1988.

[9] D.S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Memorandum 4148, Royal Signals and Radar Establishment, March 1988.

[10] A. Buhot and M.B. Gordon. Robust learning and generalization with support vector machines. *Journal of Physics A: Mathematical and General*, 34:4377–4388, 2001.

[11] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. Also at `http://www.kernel-machines.org/papers/Burges98.ps.gz`.

[12] C. Campbell. An introduction to kernel methods. In Robert J. Howlett and Lakhmi C. Jain, editors, *Radial Basis Function Networks 1: Recent Developments in Theory and Applications*, volume 66 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag, 2001.

[13] M. Carozza and S. Rampone. Function approximation from noisy data by an incremental RBF network. *Pattern Recognition*, 32:2081–2083, 1999.

[14] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.

[15] C.-C. Chang and C.-J. Lin. Training ν-support vector classifiers: Theory and algorithms. *Neural Computation*, 13(9):2119–2147, 2001.

[16] C.-C. Chang and C.-J. Lin. Training ν-support vector regression: Theory and algorithms. *Neural Computation*, 14(8):1959–1977, 2002.

[17] P.-H. Chen, C.-J. Lin, and B. Schölkopf. A tutorial on ν-support vector machines. `http://www.csie.ntu.edu.tw/~cjlin/papers/nusvmtutorial.pdf`.

[18] V. Cherkassky and F. Mulier. Guest editorial: Vapnik-Chervonenkis (VC) learning theory and its applications. *IEEE Transactions on Neural Networks*, 10(5):985–987, September 1999.

[19] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. In H. Burkhardt and B. Neumann, editors, *Proceedings of the European Conference on Computer Vision 1998*, volume 2, pages 484–498. Springer, 1998.

[20] T.F. Cootes, G.J. Edwards, and C.J Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, June 2001.

[21] T.F. Cootes and C.J. Taylor. Statistical models of appearance for medical image analysis and computer vision. In M. Sonka and K.M. Hanson, editors, *Medical Imaging 2001: Image Processing*, volume 4322, pages 236–248, July 2001.

[22] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models — their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, January 1995.

[23] T.M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14:326–334, 1965.

[24] T.F. Cox and G. Ferry. Discriminant analysis using non-metric multidimensional scaling. *Pattern Recognition*, 26(1):145–153, 1993.

[25] A.P.M. Coxon. *The User's Guide to Multidimensional Scaling*. Heinemann Educational Books Ltd, 1982.

[26] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based Learning Methods*. Cambridge University Press, 2002.

[27] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[28] J. de Leeuw. Applications of convex analysis to multidimensional scaling. In J.R. Barra, F. Brodeau, G. Romier, and B. van Cutsem, editors, *Recent Developments in Statistics*, pages 133–145. North-Holland Publishing Company, Amsterdam, 1977.

[29] J. de Leeuw and W.J. Heiser. Convergence of correction-matrix algorithms for multidimensional scaling. In J.C. Lingoes, editor, *Geometric Representations of Relational Data*, pages 735–752. Mathesis Press, Ann Arbor, 1977.

[30] J. de Leeuw and W.J. Heiser. Multidimensional scaling with restrictions of the configuration. In P.R. Krishnaiah, editor, *Multivariate Analysis-V*, pages 501–522. North-Holland Publishing Company, Amsterdam, 1980.

[31] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall International, 1982.

[32] W.R. Dillon and M. Goldstein. *Multivariate Analysis: Methods and Applications*. John Wiley & Sons, 1984.

[33] D. Dong and T.J. McAvoy. Nonlinear principal component analysis — based on principal curves and neural networks. In *Proceedings of the American Control Conference*, pages 1284–1288, Baltimore, Maryland, June 1994.

[34] D. Dong and T.J. McAvoy. Nonlinear principal component analysis — based on principal curves and neural networks. *Computers and Chemical Engineering*, 20(1):65–78, 1996.

[35] T. Downs, K.E. Gates, and A. Masters. Exact simplification of support vector solutions. *Journal of Machine Learning Research*, 2:293–297, 2001.

[36] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, second edition, 2001.

[37] `www.eunite.org`.

[38] J.H. Friedman. Multivariate adaptive regression splines (with following discussion). *The Annals of Statistics*, 19(1):1–141, 1991.

[39] R. Gil-Pita, P. Jarabo-Amores, M. Rosa-Zurera, and F. Lopez-Ferreras. Improving neural classifiers for atr using a kernel method for generating synthetic training sets. In *Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing*, pages 425–434, 2002.

[40] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63:169–176, 1990.

[41] R.C. Gonzales and R.E. Woods. *Digital Image Processing*. Addison-Wesley, 1993.

[42] D.J. Hand. *Construction and Assessment of Classification Rules*. John Wiley & Sons, 1997.

[43] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, June 1989.

[44] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[45] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing, 1994.

[46] W.J. Heiser. A generalized majorization method for least squares multidimensional scaling of pseudodistances that may be negative. *Psychometrika*, 56(1):7–27, March 1991.

[47] W.J. Heiser. Convergent computation by iterative majorization: Theory and applications in multidimensional data analysis. In W.J. Krzanowski, editor, *Recent Advances in Descriptive Multivariate Analysis*, pages 157–189. Clarendon Press, Oxford, 1994.

[48] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[49] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3:551–560, 1990.

[50] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

[51] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 3(2):415–425, March 2002.

[52] S.-C. Huang and Y.-F. Huang. Bounds on the number of hidden neurons in multilayer perceptrons. *IEEE Transactions on Neural Networks*, 2(1):47–55, January 1991.

[53] `www.ijcnn.net`.

[54] A.P. Jackson. The radial basis function network: A preliminary study adapting the centres and nonlinearities. Student project report, R.S.R.E., January 1989. Supervised by D. Lowe.

[55] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, January 2000.

[56] N. Kambhatla and T.K. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9:1493–1516, 1997.

[57] R.E. Karlsen, D.J. Gorsich, and G.R. Gerhart. Target classification via support vector machines. *Optical Engineering*, 39(3):704–711, March 2000.

[58] `www.kernel-machines.org`.

[59] W.G. Koontz and K. Fukunaga. A nonlinear feature extraction algorithm using distance transformation. *IEEE Transactions on Computers*, C-21(1):56–63, January 1972.

[60] M.A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, February 1991.

[61] M.A. Kramer. Autoassociative neural networks. *Computers and Chemical Engineering*, 16(4):313–328, 1992.

[62] J.B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, March 1964.

[63] J.B. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, June 1964.

[64] J.B. Kruskal. Comments on "A nonlinear mapping for data structure analysis". *IEEE Transactions on Computers*, 20:1614, December 1971.

[65] J.B. Kruskal and M. Wish. *Multidimensional Scaling.* Quantitative Applications in the Social Sciences. Sage Publications, 1978.

[66] C. Lawlor and R. Jackson. Classification of targets in ISAR data. Unpublished DERA report.

[67] M. LeBlanc and R. Tibshirani. Adaptive principal surfaces. *Journal of the American Statistical Association*, 89(425):53–64, March 1994.

[68] J. Leonard and M.A. Kramer. Improvement of the backpropagation algorithm for training neural networks. *Computers and Chemical Engineering*, 14(3):337–341, 1990.

[69] D. Lowe. Adaptive radial basis function nonlinearities, and the problem of generalisation. In *First IEE International Conference on Artificial Neural Networks*, pages 171–175, October 1989.

[70] D. Lowe. On the use of nonlocal and non positive definite basis functions in radial basis function networks. In *Fourth IEE International Conference on Artificial Neural Networks*, pages 206–211, June 1995.

[71] D. Lowe. Radial basis function networks. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 779–782. MIT Press, Cambridge, MA, USA, 1998.

[72] R. Maines. Socially camouflaged technologies: The case of the electromechanical vibrator. *IEEE Technology and Society Magazine*, pages 3–23, June 1989.

[73] E.C. Malthouse. Limitations of nonlinear PCA as performed with generic neural networks. *IEEE Transactions on Neural Networks*, 9(1):165–173, January 1998.

[74] E.C. Malthouse, R.S.H. Mah, and A.C. Tamhane. Some theoretical results on nonlinear principal components analysis. In *Proceedings of the American Control Conference*, pages 744–748, Seattle, Washington, June 1995.

[75] A.M. Martínez. Recognizing imprecisely localized, partially occluded, and expression variant faces from a single sample per class. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):748–763, June 2002.

[76] G.T. Maskall. An application of nonlinear feature extraction to the classification of ISAR images. In *2002 International Radar Conference*, volume CP490, pages 405–408, October 2002.

[77] G.T. Maskall and A.R. Webb. Nonlinear feature extraction applied to ISAR images of targets for classification. In F.A. Sadjadi, editor, *Automatic Target Recognition XI*, volume 4379, pages 255–265. SPIE, October 2001.

[78] G.T. Maskall and A.R. Webb. Nonlinear feature extraction for MMW image classification: a supervised approach. In F.A. Sadjadi, editor, *Automatic Target Recognition XII*, volume 4726, pages 353–363. SPIE, July 2002.

[79] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909.

[80] D. Meyer, F. Leisch, and K. Hornik. The support vector machine under test. *Neurocomputing*, 55:169–186, 2003.

[81] C.A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

[82] D. Michie, D. Spiegelhalter, and C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood Limited, 1994.

[83] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.

[84] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, March 2001. Also at `http://mlg.anu.edu.au/~raetsch/ps/review.pdf`.

[85] H. Niemann and J. Weiss. A fast-converging algorithm for nonlinear mapping of high-dimensional data to a plane. *IEEE Transactions on Computers*, C-28(2):142–147, February 1979.

[86] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

[87] M. Paluš and I. Dvořák. Singular-value decomposition in attractor reconstruction: pitfalls and precautions. *Physica D*, 55:221–234, 1992.

[88] J. Park and I.W. Sandberg. Universal approximation using radial basis function networks. *Neural Computation*, 3(2):246–257, 1991.

[89] J. Park and I.W. Sandberg. Approximation and radial-basis-function networks. *Neural Computation*, 5:305–316, 1993.

[90] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, September 1990.

[91] M. Pontil and A. Verri. Properties of support vector machines. *Neural Computation*, 10(4):955–974, 1998.

[92] M.J.D. Powell. Radial basis functions for multivariate interpolation: A review. In J.C. Mason and M.G. Cox, editors, *Algorithms for Approximation*, pages 143–167. Clarendon Press, Oxford, 1987.

[93] M.J.D. Powell. The theory of radial basis function approximation in 1990. In Will Light, editor, *Wavelets, Subdivision Algorithms and Radial Basis Functions*, volume II of *Advances in Numerical Analysis*, chapter 3, pages 105–210. Clarendon Press, Oxford, 1992.

[94] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge, second edition, 1994.

[95] V. Roth. The generalized LASSO. *IEEE Transactions on Neural Networks*, 15(1), January 2004.

[96] R.M. Sakia. The Box-Cox transformation technique: A review. *The Statistician*, 41(2):169–178, 1992.

[97] J.W. Sammon, Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.

[98] T.D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2:459–473, 1989.

[99] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A.J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, September 1999.

[100] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[101] B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[102] R.N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika*, 27(2):125–140, June 1962.

[103] R.N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. II. *Psychometrika*, 27(3):219–246, September 1962.

[104] A.J. Smola. *Learning With Kernels*. PhD thesis, Technische Universität Berlin, 1998.

[105] A.J. Smola and B. Schölkopf. A tutorial on support vector regression. NeuroCOLT Technical Report NC2-TR-1998-030, Royal Holloway College, University of London, 1998.

[106] I. Steinwart. On the optimal parameter choice for ν-support vector machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1274–1284, October 2003.

[107] I. Steinwart. Sparseness of support vector machines. *Journal of Machine Learning Research*, 4:1071–1105, 2003.

[108] I. Steinwart. Sparseness of support vector machines — some asymptotically sharp bounds. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16, Cambridge, MA., 2004. MIT Press.

[109] M. Stinchcombe and H. White. Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. In *International Joint Conference on Neural Networks*, volume 1, pages 613–617, June 1989.

[110] G. Strang. *Linear Algebra and its Applications*. Harcourt Brace & Company, 1988.

[111] `www.support-vector.net`.

[112] R. Tibshirani. Principal curves revisited. *Statistics and Computing*, 2(4):183–190, 1992.

[113] M.E. Tipping. Sparse kernel principal component analysis. In *Advances in Neural Information Processing Systems 13*, pages 633–639, 2000.

[114] M.E. Tipping and C.M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11:443–482, 1999.

[115] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[116] R. van der Heiden. *Aircraft Recognition with Radar Range Profiles*. PhD thesis, Universiteit van Amsterdam, 1998.

[117] R. van der Heiden and F.C.A. Groen. A metric for the radial basis function network — application on real radar data. In *Proceedings of the 1996 World Congress on Neural Networks*, pages 363–367, 1996.

[118] R. van der Heiden and F.C.A Groen. The box-cox metric for nearest neighbour classification improvement. *Pattern Recognition*, 30(2):273–279, 1997.

[119] V.N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[120] V.N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, September 1999.

[121] A.R. Webb. Nonmetric multidimensional scaling using feed-forward networks. CSE1 Research Note 192, Defence Research Agency, June 1992.

[122] A.R. Webb. Multidimensional scaling by iterative majorization using radial basis functions. *Pattern Recognition*, 28(5):753–759, 1995.

[123] A.R. Webb. An approach to non-linear principal components analysis using radially symmetric kernel functions. *Statistics and Computing*, 6:159–168, 1996.

[124] A.R. Webb. Nonlinear feature extraction with radial basis functions using a weighted multidimensional scaling stress measure. In *Proceedings of the 13th International Conference on Pattern Recognition (ICPR)*, volume 4, pages 635–639, 1996.

[125] A.R. Webb. Radial basis functions for exploratory data analysis: An iterative majorisation approach for Minkowski distances based on multidimensional scaling. *Journal of Classification*, 14:249–267, 1997.

[126] A.R. Webb. A loss function approach to model selection in nonlinear principal components. *Neural Networks*, 12:339–345, 1999.

[127] A.R. Webb. *Statistical Pattern Classification*. Arnold, 1999.

[128] A.R. Webb. via e-mail, 2006. Personal communication.

[129] A.R. Webb and S. Shannon. Shape-adaptive radial basis functions. *IEEE Transactions on Neural Networks*, 9(6):1155–1166, November 1998.

[130] L. Xu, E. Oja, and C.Y. Suen. Modified Hebbian learning for curve and surface fitting. *Neural Networks*, 5:441–457, 1992.

[131] M-H. Yang, D.J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, January 2002.

[132] L. Ying, R. Yong, and S. Xiuming. Radar HRRP classification with support vector machines. In Y.X. Zhong, S. Cui, and Y. Wang, editors, *ICII: 2001 International Conferences on Info-tech and Info-net*, volume 1, pages 218–222, Beijing, China, 2001.

[133] Q. Zhao and J.C. Principe. Support vector machines for SAR automatic target recognition. *IEEE Transactions on Aerospace and Electronic Systems*, 37(2):643–654, April 2001.

[134] W. Zhao, R. Chellapa, P.J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys (CSUR)*, 35(4):399–458, December 2003.