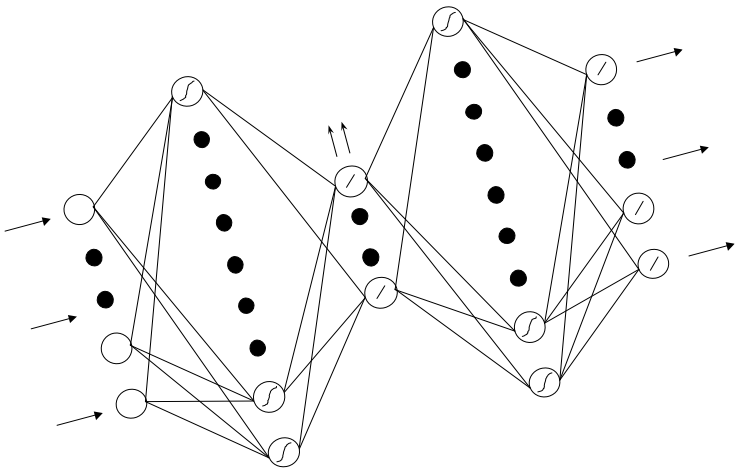
Non-linear target based feature extraction by diabolo networks

Erik van der Werf



Pattern Recognition Group
Department of Applied Physics
Faculty of Applied Sciences
Delft University of Technology

Delft, May 1999

Mentor: Dr. Ir. R.P.W. Duin

Abstract

In complex real-world pattern recognition problems, like image-recognition, many difficulties arise that are directly related to the high dimensionality of the problem. To make problems tractable, feature extraction methods are needed for reducing dimensionality whilst preserving the relevant information.

Classical linear feature extraction methods have been shown effective for many simple problems. However, several problems exist in which the linear methods fail. Furthermore, we cannot expect linear methods to function optimal for non-linear problems.

In this report a new training method for supervised and unsupervised feature extraction with diabolo networks is presented which can improve performance, for classification, compared to the classical feature extraction methods such as statistical discriminant analysis and principal component analysis.

Principal Component Analysis is an unsupervised linear feature extraction method that aims at preserving the maximum amount of variance. In literature a target based training method was found for diabolo networks which can be used as a non-linear extension to the Principal Component Analysis. In this report a method for initialising training of such a network is presented which can improve performance and training time.

In this report a new target based training method is presented, for artificial neural diabolo networks, which produces a mapping comparable to Fisher's linear discriminant mapping. Fisher's linear discriminant mapping is a well-known supervised linear feature extraction method that can be used to decrease dimensionality while preserving class separability.

It is shown in experiments, on both artificial and real data, that linear and non-linear feature extraction performed by diabolo networks can increase class separability, compared to classical linear mapping methods.

Contents

1 Introduction

Features are measurements, and can be thought of as knowledge about the world. This knowledge can be used to make decisions. In pattern recognition this is called classification. In the case that our some knowledge of the current 'state of the world', is contained in the form of n measurements, we have a feature-vector in an n-dimensional feature-space. In the ideal case, decisions are made based on the position of this point in that feature-space. Therefore a classifier is partitioning the n-dimensional feature-space into regions associated with certain decisions. In the case of simple problems with only a small number of relevant features many classifiers function well. In most cases, however, this initial feature-space will not be efficient. It is not efficient since features can be redundant, correlated or non-linearly related. Another problem for all distance-based classifiers, is the loss of meaning for distances in high dimensional space. And in practical applications the computational cost of calculations grows with the number of features.

To obtain more efficient feature-vectors it is necessary to reduce the dimensionality of the feature-space. Methods that do this are feature selection and feature extraction. In this report we focus at feature extraction. Although feature selection is just a special case of feature extraction, it can greatly decrease computational demand by discarding redundant features. An interesting approach to feature selection, based on the analysis of weights in neural networks, which might also be applied to our neural networks, can be found in [1].

In pattern recognition there is a wide range of methods for feature extraction such as principal component analysis, statistical discriminant analysis, independent component analysis [2], Kohonen's self organising mappings [3] and Sammon's mapping [4]. This report describes and analyses feature extraction methods based on non-linear mapping of original features onto a lower-dimensional subspace with feed-forward neural networks.

In this report a new supervised target based feature extraction method is presented which can extract features from a broad range of class distributions. The reason for developing this new feature extraction method is that classical linear feature extraction methods are only optimal for the limited range of linear separable class distributions. In practical applications we cannot expect classes to be linear separable.

Although the problem of extracting information is more general, we focus our research at methods suitable for image databases. However, the same pattern recognition techniques can also be applied to other kinds of data.

The rest of this report is organised as follows: first in the next section something is said about why we started looking into feature extraction methods.

In chapter 2 some classical feature extraction methods are discussed together with a new non-linear neural implementation for supervised feature extraction.

In chapter 3 a new method for initialising the diabolo network is described, which can increase classification performance and decrease training time compared to random initialisations. For completeness also a brief overview is given about the architecture and training methods for feed-

forward neural networks. Readers not familiar with neural networks should consider reading the introduction to feed-forward neural networks in chapter 3 before reading chapter 2.

In chapter 4 some classifiers are described which are later used in chapter 5 to compare the performance of feature extraction methods.

Next, in chapter 5 experiments are performed on artificial, 2-dimensional feature-spaces. Then some experiments are shown for feature extraction for the recognition of handwritten numbers. Finally some preliminary results are presented for feature extraction from real-world images.

Finally, in the last chapter, some conclusions are drawn.

1.1 Background

In today's world where fast computers become largely available and the price of digital storage falls rapidly there is an ever-increasing need for fast information retrieval. Most current database systems can only handle text and numerical data, as more databases also include information in the form of audio and visual data, new problems arise.

The current image database systems fall into one of two categories, text-based or content-based systems.

In text based systems the image is assigned keywords that describe its content. At present it is not yet possible to get a computer to assign keywords which are at the level of human perception. Therefore, text-based systems require human interaction to describe an image. This has many drawbacks, for the process is slow, subjective and it does not allow for similarity retrieval. The difficulty arises partly because it is not possible to guarantee that different people use the same expressions for the same images. There is a deeper reason: the information sought is inherently in the form of imagery, which a textual language is unable to express adequately, thus making querying inefficient.

Though high level automatic keyword assignment is not yet possible, many algorithms exist that are able to capture low-level properties of images such as texture, colour, shape and size. Several current content-based systems base their image retrieval on indices coding such low-level properties. Some well known commercial systems include IBM's QBIC [5], Query By Image Content, and Virage [6]. Although some interesting results have been obtained, using such low-level features, most systems cannot handle very selective querying. To make selective querying efficient, more advanced features will have to be devised. We hope to obtain such features by applying advanced feature extraction methods to low-level features.

1.2 Acknowledgements

I would especially like to thank the following people

- Bob Duin for all the advice.
- Dick de Ridder for the nistdigs set, and the extra 1020 images from Surrey.
- Jonathan Stoeckel for the interesting discussions and muggenzifting this report.
- Kieron Messer from the Centre for Vision Speech and Signal Processing, at the University of Surrey, for the filters that were used in the experiments on image-databases.
- Olaf Lemmers for proof-reading this report, keeping me filled with coffee and letting me beat him at soccer in places where no-one played soccer before)

And all other people at the Pattern Recognition Group for new ideas, interesting discussions and a real pleasant time.

2 Feature extraction

A feature extraction method or mapping method is a process that has as input an n-dimensional feature-vector x and as output an m-dimensional feature-vector y, $m<n$, which is a possibly non-linear function of the elements in x. The goal of projecting the original n-dimensional feature-vector onto an m-dimensional subspace is to get a more efficient combination of the original features.

Mapping methods can either be linear or non-linear and supervised or unsupervised. The difference between supervised and unsupervised methods is whether or not class information is used to find the optimal mapping. In the next sections methods for both supervised and unsupervised mapping will be discussed, for the classical linear case as well as their non-linear neural counterparts.

Readers not familiar with feed-forward neural networks should consider reading the introduction to feed-forward neural networks in chapter 3 before proceeding into chapter 2.

2.1 Unsupervised mapping methods

Unsupervised feature extraction is mapping without use of class labels. This means that the partitioning of the feature-space is unknown. Since for unsupervised mapping it is assumed that distances between feature-vectors and positions in the original feature-space have meaning, all methods try to preserve them in some way.

2.1.1 Principal Component Analysis

The most widely used linear mapping is the Principal Component Analysis (PCA), also known as the Karhunen-Loève transform. This unsupervised mapping method is a linear projection method that assumes that the best mapping preserves the maximum amount of variance.

For n-dimensional column-vectors x, linear feature extraction can be written as

$$y = F^T x, \tag{1}$$

in which y is an m-dimensional feature-vector and F is an n x m matrix build from m orthonormal n-dimensional vectors. For simplicity we assume that x is a zero-mean process so that a bias-term can be left out. If the original n-dimensional feature-space is linearly reconstructed from the extracted m-dimensional feature-vectors, we get the linear projection

$$\hat{x} = Fy = FF^T x. \tag{2}$$

A measure for the performance of the feature extraction, based on how well the original feature-space can be reconstructed from the extracted features, can be written as

$$mse = E[|x - \hat{x}|^2], \tag{3}$$

in which E is the expectation operator. Since the vectors in F are orthonormal this can be rewritten as

$$mse = E[x^T x - x^T F F^T x]. \tag{4}$$

Minimising (4) means rotating the orthonormal vectors in F to the directions of largest variance. Therefore PCA finds a projection onto a subspace spanned by the m largest eigenvectors of the covariance matrix.

It is easily shown that a 3-layer feed-forward neural network with n inputs, m neurons in the second layer and n neurons in the third layer, with all linear activation functions, is able to perform a PCA mapping between the first and the second layer. Such neural networks are often called auto-associative or diabolo networks. A schematic overview of a linear diabolo network is shown in Figure 1.

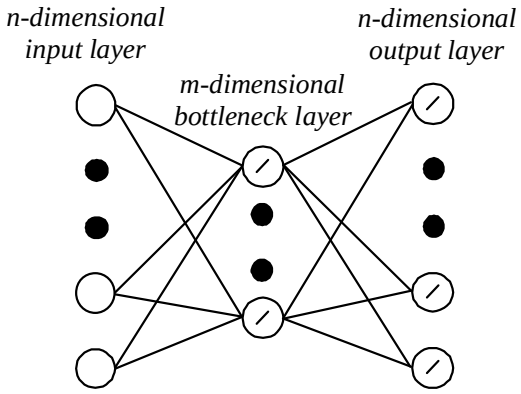


Figure 1, linear diabolo network

To learn to approximate a PCA mapping between the input and bottleneck layer, and a reconstruction between the bottleneck and output layer, the output of the network is trained to approximate the input. In other words, the network is trained to reconstruct the input as well as possible. After training, the network is split and the first half, between the input and the bottleneck layer, is used for extracting the new, m-dimensional, feature-vectors. If the targets are trained to approximate the inputs using the mean-square-error criterion the network will find a projection to the same m-dimensional subspace as found by normal PCA. The main difference with normal PCA is that, in general, the extracted features will be non-orthogonal rotated versions of those found by normal PCA. Furthermore, it turns out that for practical applications the normal PCA, which is non-iterative, is trained much faster.

2.1.2 Non-Linear Principal Component Analysis

An extension to the PCA network is the Non-Linear Principal Component Analysis (NLPCA) network. The difference with linear PCA networks is that this network has extra hidden layers between in- and output and the bottleneck-layer. The neurons in these extra hidden layers have non-linear activation functions, which allow the network to find non-linear subspaces.

The smallest NLPCA network with non-linear compression and reconstruction, shown in Figure 2, uses five layers. The first and the fifth layer are the n inputs and outputs. The third layer has m neurons, usually with linear activation functions. The neurons in the second and fourth layer perform the non-linear transformation. The number of neurons in these layers depends on the amount of non-linearity needed in compression and reconstruction of the data. After the network is trained it is split and the first half, between the input and bottleneck layer, is used for extracting the m-dimensional features.

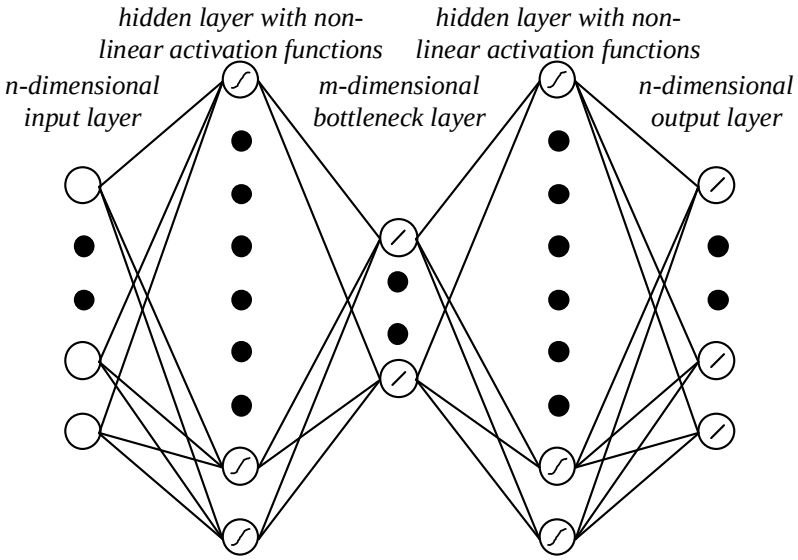


Figure 2, Non-linear diabolo network

An analysis of the NLPCA network was done by Malthouse [7]. He showed that NLPCA networks can only find continuous non-self-intersecting subspaces. The continuity of the projection comes directly from the constraint that each neuron's activation function is continuous. The reason for this is that feed-forward neural networks are usually trained with some form of back-propagation. Back-propagation requires a derivative of each activation function to propagate the error observed at the end of the network back through the network. Other training-functions such as re-enforced learning do not require back-propagation and may therefore overcome the problem with continuity. Although re-enforced learning may overcome problems with continuity we did not use it since training would be practically impossible, given the current speed of computers. Furthermore it turns out that given enough neurons in the hidden layers the network is still able to find good approximations of problematic subspaces. The second problem, with self-intersecting subspaces is more fundamental and comes directly from the fact that one point in the original feature-space cannot be projected back onto two points in the extracted feature-space.

The NLPCA network can be trained in the same way as the linear PCA-network, but this method proves to be even more time consuming. One method to improve training speed is to initialise the network with the optimal result of PCA. In the next chapter a method is described for initialising the diabolo network with the linear projection obtained from PCA. Oja presented another training method, which might also improve training speed. In [8] he suggested that the five-layer network could be trained in 3 steps. The first step would be to maximise entropy in the second layer. After that a linear PCA could be calculated between the second and the third layer. Finally, if a reconstruction would be necessary, the last layers could be trained to approximate the inputs. A third method, which can sometimes be employed to speed up the training, is to add more hidden layers. In theory a network with one hidden layer can represent any continuous

function and a network with 2 hidden layers can represent any function [9]. Although this does not hold for a small number of neurons with continuous activation functions, it suggests that 2 hidden layers for the extraction and 2 hidden layers for the reconstruction, are better equipped to approximate discontinuous jumps or other complex shapes.

2.2 Supervised mapping methods

Supervised feature extraction is mapping with use of class examples; i.e. the partitioning of the feature-space is known.

2.2.1 Criteria for class separability

When there are two or more classes, feature extraction is equivalent to the choice of the mapping, which is most effective for showing class separability. In statistical discriminant analysis, within-class, between-class and mixture scatter matrices are used to formulate criteria of class separability. The within-class scatter matrix shows the scatter of samples x_i around their class expected vector μ_i, and is expressed by

$$S_w = \sum_{i=1}^{k} P_i E[(x_i - \mu_i)(x_i - \mu_i)^T], \tag{5}$$

a between-class scatter matrix can be defined as

$$S_b = \sum_{i=1}^{k} P_i (\mu_i - \mu)(\mu_i - \mu)^T, \tag{6}$$

the mixture scatter matrix is the covariance matrix of all samples regardless of their class assignments, and is defined by

$$S_m = E[(x - \mu)(x - \mu)^T] = S_w + S_b. \tag{7}$$

P_i are the a priori probabilities of classes i, k is the number of classes and μ is the mean of all vectors regardless of class assignment.

In order to formulate criteria for class separability, we have to derive a number from the scatter-matrices. In general the number should be larger when the between-class scatter is larger or the within-class scatter is smaller. There are many ways to do this, and some typical criteria are the following:

$$J_1 = tr\left(S_2^{-1} S_1\right), \tag{8}$$

$$J_2 = \left|S_2^{-1} S_1\right| = |S_1| / |S_2|, \tag{9}$$

$$J_3 = tr(S_1) / tr(S_2), \tag{10}$$

in which S_1 and S_2 are one of S_b, S_w or S_m. In our experiments we used S_b for S_1 and S_w for S_2. Criteria J_1 and J_2 are invariant under any non-singular linear transformation, while criterion J_3 is dependent on the coordinate system.

Although the above criteria for class-separability have been applied successfully to many problems in pattern recognition, it should be pointed out that they only concern second order statistics, therefore more complex distributions will result in unreliable estimates of how well classes are separable. In theory the best measure for class-separability is the performance of a Bayes classifier. Therefore a mapping is optimal if the error made by a Bayes classifier is minimised. However, since in most cases the probability-density functions are unknown, we cannot use the Bayes classifier to calculate class-separability. Fortunately, we can use other classifiers as estimates for the performance of a Bayes classifier, to calculate the performance of our feature extraction methods. In chapter 4 some classifier are discussed that were used in our experiments.

2.2.2 Fisher's linear discriminant mapping

In [10] it is shown that the optimal linear solution with respect to (8) and (9) for representing the n-dimensional feature-vectors x in an m-dimensional linear subspace, m<n, is to project x onto the surface spanned by the m largest eigenvectors of $S_2^{-1}S_1$. When S_b is used for S_1 and S_w is used for S_2, this projection is known as Fisher's linear discriminant mapping (FLD).

Linear FLD mapping is optimal for the projection of m+1 classes which have similar covariance matrices. The reason for this is that the within-scatter is calculated by averaging the covariance matrices of all classes. This means that the estimate is optimal for linear-separable class-distributions. If more then m+1 classes are used, FLD mapping remains optimal if the class-means span an m dimensional subspace and the covariance matrices are equal.

Although linear FLD mapping has proven to be a useful tool in pattern recognition, it has some drawbacks. The most important drawback is the criterion for separability, since it is only optimal for linear separable classes and only invariant under non-singular linear transformations. Another problem with FLD mapping is that it is limited to linear projections. Sometimes a solution is found by the addition of polynomial features. However, if the number of input-features is large, the number of polynomial features tends to explode, thus requiring an even larger number of training samples which is computationally and practically unattractive. A third problem is that in the case of a small number of samples, the within-class scatter matrix becomes singular. In our research we used regularisation of the covariance matrices. Regularisation makes the covariance matrix non-singular by simulating the addition of uncorrelated noise. However, there are other methods for calculating the optimal discriminant vectors. For example in [11] a method for calculating FLD mapping using rank decomposition is presented which might also help to overcome problems with small numbers of samples.

2.2.3 Supervised feature extraction by diabolo networks

Since diabolo networks can learn linear and non-linear PCA mapping by unsupervised training of the output to approximate the input [7,8], we wonder if a similar training method, with targets presented at the output of the network, could be applied to learn FLD mapping.

The first question, in training a diabolo network to approximate FLD mapping, is what targets should be used. For the PCA network we chose the input vectors as targets. While this approach ensures that the global structure is preserved, it does not necessarily improve class separability. Since we choose to train our diabolo networks with targets presented at the output of the network, we cannot directly optimise class separability in the bottleneck layer. We can however try to optimise class separability at the output of the network. If the reconstructed feature-space is well separable the same should hold for the extracted features in the bottleneck layer, because all information at the output is also present in the bottleneck. It should, however, be noted that the job of actually separating the classes in the bottleneck layer might sometimes be harder due to the non-linear transformations.

To enhance class separability we would like a contraction of each class. Ideally each class would be projected in one unique point. Therefore, to train a diabolo network to find a good separable mapping we use one unique point per class as the target.

Since neural networks are usually trained using the mean-square-error criterion, it is instructive to see what this does to our scatter measures for class separability.

For given reconstructed feature-vectors $\hat{x}_i$ of classes i the performance criterion for training the diabolo network with targets t_i becomes

$$mse = \sum_{i=1}^{k} P_i \mathrm{E}[|t_i - \hat{x}_i|^2], \tag{11}$$

in which P_i is the a priori probability of class i and k is the number of classes. Criterion (11) can be rewritten as

$$mse = \sum_{i=1}^{k} P_i \mathrm{E}[|\hat{x}_i - \mathrm{E}[\hat{x}_i]|^2] + \sum_{i=1}^{k} P_i \mathrm{E}[|t_i - \mathrm{E}[\hat{x}_i]|^2] = tr(S_w(\hat{x})) + \sum_{i=1}^{k} P_i \mathrm{E}[|t_i - \mathrm{E}[\hat{x}_i]|^2], \tag{12}$$

showing that minimising distances to one target per class is a trade-off between minimising within-scatter and restoring the between-scatter associated with the target positions. This is a fundamental difference with normal FLD mapping, which is another trade-off between minimising within-scatter and maximising between-scatter.

Although (12) tells us that class separability is increased, it does not directly tell us what targets should be used. A criterion that could be optimised for the targets is the preservation of the global structure in the reconstructed feature-space. To do this we can define a scatter measure, similar to (5), showing the scatter of samples x_i around their target vector t_i as

$$S_{tw} = \sum_{i=1}^{k} P_i E[(x_i - t_i)(x_i - t_i)^T], \tag{13}$$

in which P_i is the a priori probability of class i and k is the number of classes.

It can easily be shown that the trace of S_{tw} (13), which is the mean-square distance to targets, is minimised by choosing the class means as targets. In most applications these targets perform well. In some cases, however, they can create a problem since the distances between targets of overlapping classes or classes having strange distributions could become small, thus resulting in reduced class-separability for the extracted features.

To overcome problems for most class distributions a second scatter measure can be devised which shows the scatter of the target vectors t_i around the expected vector μ, regardless of class assignment, as

$$S_{tb} = \sum_{i=1}^{k} P_i E[(t_i - \mu)(t_i - \mu)^T], \tag{14}$$

With these two scatter matrices targets can be calculated by maximisation of one of the measures for class-separability (8),(9) or (10) using S_{tb} for S_1 and S_{tw} for S_2. Our choice for these measures of class-separability is motivated by the fact that these criteria are also used for FLD mapping and therefore might be useful for comparing NLFLD mapping to FLD mapping. In general however there is a much wider variety of clustering techniques, optimising other criteria, that could be applied to calculate targets.

Although it is shown that training with targets is possible, we would like to point out that there are other ways to train a neural network to extract features. Recently, in [12] an interesting approach to non-linear feature extraction with feed-forward neural networks was presented, based on direct optimisation of (9) as a function of the network weights. Another interesting linear feature-extraction method, is presented in [13]. This method is based on maximising the difference in feature-vector lengths between different classes.

3 Network training

In this chapter training methods are discussed that can be applied to diabolo networks.

Since not all readers may be familiar with neural networks, first something is said about the general architecture of feed-forward neural networks and the standard algorithms used to train feed-forward neural networks. Readers familiar with neural network can skip section 3.1 and 3.2 and go directly to section 3.3, where a new initialisation method is presented which can improve training speed and performance for diabolo networks.

3.1 Feed-forward Neural networks

Although some important features of feed-forward neural networks are discussed below, this section is by no means a complete introduction to neural networks. For readers that want to know more about neural networks an introduction to neural networks can be found in [14] or on the Internet at: http://www.shef.ac.uk/psychology/gurney/notes/contents.html.

A commonly used type of neural network is the feed-forward neural network. This network consists of one input and one output layer and in between an arbitrary number of hidden layers. Each layer is build out of an arbitrary number of neurons, which are connected to all the neurons in the previous layer. There are no connections to neurons in the same layer. Furthermore, the connections are only in one direction to the output layer (no feedback). A schematic interpretation of a typical feed-forward neural network is shown in Figure 3.

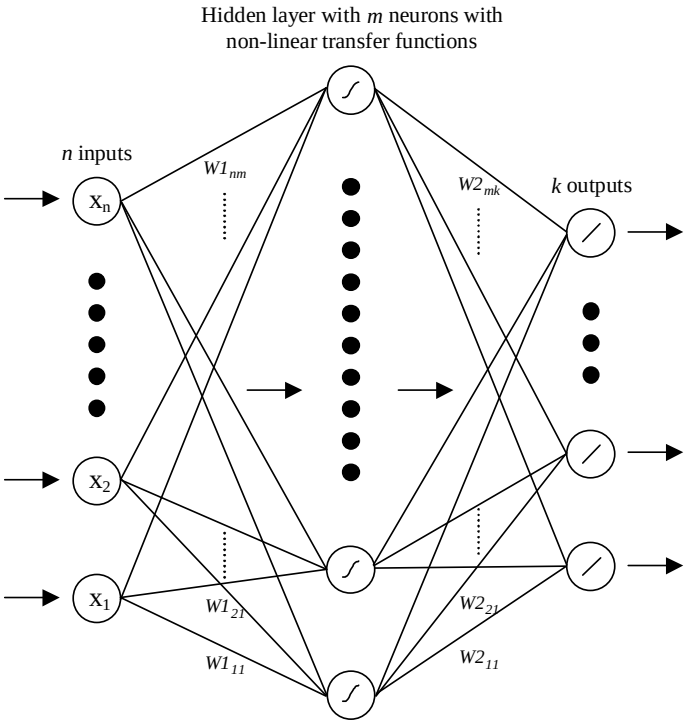


Figure 3, typical feed-forward neural network with one hidden layer

Neurons are the basic building blocks of the neural network. The neuron has a large number of input nodes, and one single output node. The output of a neuron can be described by

$$o = f(w^T i + b), \tag{15}$$

in which i is a column-vector consisting of the values of the output nodes of the neurons in the previous layer, w is a weight vector and b a bias. In our feed-forward neural networks each neuron has its own set of weights and biases, the transfer function f is the same for all neurons in one layer. The most commonly used sigmoidal transfer function is the log sigmoid (logsig)

$$f(x) = \frac{1}{1+\exp(-x)}. \tag{16}$$

Another transfer function is the linear transfer function (purelin)

$$f(x) = x, \tag{17}$$

Closely related to the sigmoid transfer function is the hyperbolic tangent sigmoid transfer function (tansig)

$$f(x) = \frac{2}{1+\exp(-2x)} - 1, \tag{18}$$

which main advantage over the logsig is that it is approximately linear around zero, therefore allowing easy conversion from linear to non-linear transfer functions, if the data is scaled around zero.

3.2 Training methods

3.2.1 Backpropagation

The most common algorithm used for training feed-forward neural networks is called error back-propagation [14]. Error back-propagation is an example of supervised learning, not to be confused with supervised feature-extraction, because at each step the network is adjusted by comparing the actual output with the desired output.

If all transfer functions have a derivative, an infinitely small change at the output of any neuron in the network results in some linear response at the output of the network. It also works the other way around. An infinitely small change at the output of the network can directly be related, through linearisation, to a change at any neuron. Loosely said this means that we know how much any neuron contributes to the output of the neural network. This property allows an error observed at the end of the network, usually the mean-square error, to be distributed over all neurons by propagating the error back through the network.

The simplest update for an arbitrary network weight w_i can be written as

$$\Delta w_i = -\alpha \frac{\partial E}{\partial w_i}, \tag{19}$$

in which α is a parameter determining the step length of each iteration. Criterion 19 is a form of steepest descent minimisation, and therefore performs quite poor in most complex applications. A better criterion for updating the network weights is

$$\Delta w_i = -\alpha\beta \frac{\partial E}{\partial w_i} + \beta \Delta w_{previous}, \tag{20}$$

in which α is called the learning rate, and β is a momentum term that helps the network to escape local minima. In most experiments gradient descent with momentum and adaptive learning was used. This training function, implemented in Matlab's neural network toolbox, minimises the error observed at the end of the network using (20), while constantly changing the learning rate α based on the change in performance due to the last network update.

Another training function is conjugate gradient back-propagation. This function calculates network updates based on line-search in the direction of the gradient calculated with normal back-propagation. Although this type of training often has faster convergence than gradient descent with momentum and adaptive learning we found that it often gets stuck in local minima, where the final performance is worse than that found by gradient descent with momentum and adaptive learning.

For networks with only a small number of weights more complex training functions such as Levenberg-Marquardt back-propagation, or Gauss-Newton algorithms can be used. For moderate networks various quasi-Newton algorithms are efficient. In general the performance of these algorithms is better than the gradient descent-like algorithms. However, due to the calculation of the Jacobian or Hessian matrices, their consumption of memory and time tends to explode for networks with large numbers of neurons, thus making training practically impossible.

In general training of a neural network can be seen as searching for the lowest error in a high dimensional error surface. Since all methods are local none of the above methods are guaranteed to find the global minimum and only iterative approaches exist, which may get stuck in local minima.

3.2.2 Validation and ending training

An important question in training a neural network is when to stop training. For small networks with large numbers of different examples training can be stopped if either the performance goal has been reached, or training hasn't significantly improved for an arbitrary number of network updates. If networks become large or the number of samples becomes small new problems arise.

It is well known that, given enough neurons, feed-forward neural networks can fit a function through any set of points. However, such a fit becomes meaningless due to loss of generalisation.

This behaviour is called over-training, and can be thought of as fitting noise. To prevent over-training we need to estimate when training is useful and when it no longer brings any generalised improvement. A way to do this is by the use of validation vectors. Validation vectors are used to calculate the network's performance without directly using them for training. Training should be ended if the performance on validation vectors starts decreasing. To get a good estimate of when training should be ended validation vectors should be as independent as possible, this however means that, especially in the case of small numbers of samples, not all information can be used for training. The choice of validation vectors therefore always is a trade-off between getting enough training data and getting good stop-criteria.

A normal training round is done in two steps. First subsets are selected for training and validation. Which are then used to train the network until some performance goal, decrease in validation performance or time-limit is reached. This type of training has the drawback that not all information is used because some samples are left out for validation. In our experiments with only limited numbers of samples, training and validation vectors were used from the same set. To be able to use all examples while still being able to use validation vectors, training was done in more rounds. For each training-round new subsets for training and validation were used, therefore allowing the information in all training vectors to be used. Although this method may not be optimal we find it to be a good solution when only a small number of training samples can be spared for validation. Another and probably the best solution in the case of independent samples is to estimate the maximum number of weight updates using a leave-one-out criterion. In practise however, this is extremely time-consuming since a network has to be trained for each sample.

3.3 Improving network initialisation for diabolo networks

Before a diabolo network can be trained initial weight values have to be set, which can serve as a starting-point in our search for a global minimum in the high dimensional error-surface. Usually there are only two constraints to these initial weights. The first being that all data is mapped in a unique way to each neuron, thus avoiding symmetry problems. The other constraint is that weights are scaled to map data in regions of the sigmoid that have useful derivatives. Large weights usually mean that the data is projected onto the flat part of the sigmoid, resulting in poor gradient information. Therefore most neural networks are initialised with small pseudo-random values scaled to project input samples around zero, at the input of neurons.

In complex problems networks tend to converge in local minima. Therefore, the initialisation and the training algorithm determine the local minimum in which the network will end. To find a close approximation of the global minimum, many training attempts may have to be made, thus making training extremely time consuming. However, if some knowledge about the solution to the problem is available it can be used to find better starting-points. In the case of feature-extraction such knowledge is available, and we have developed a method to use this knowledge in the initialisation of diabolo networks.

3.3.1 Initialization with optimal linear results

Training a non-linear diabolo network with small pseudo-random initialisations generally takes quite a long time to reach an optimal performance and may get stuck in local minima. Since good linear solutions to supervised and unsupervised feature-extraction are known (see experiments in chapter 5) these can be used as a starting point for training the diabolo networks.

This is done in five steps:

1. First a linear mapping and reconstruction is calculated which results in a linear diabolo network with one hidden bottleneck layer with m neurons.
2. For obtaining a five-layer diabolo network two extra layers are added between the hidden layer and the output layer. Initially these extra layers use m neurons, with linear activation functions. The weights and biases of the new layers are set to perform a unity mapping, thus ensuring that the output of the network remains the same. The middle of the three m-dimensional layers becomes the new bottleneck layer.
3. The m neurons in the layers around the bottleneck layer are copied a number of times and the connection weights are divided by the number of copies so that the inputs to the next layers remain the same.
4. The linear activation functions, of the neurons in the layers around the bottleneck layer, are replaced with non-linear hyperbolic tangent sigmoid activation functions. The weights and biases are adjusted so that the data is approximately in the linear part of the activation function.
5. Finally to avoid symmetry problems in training, due to identical neurons in the same layer, noise is added to all weights and biases.

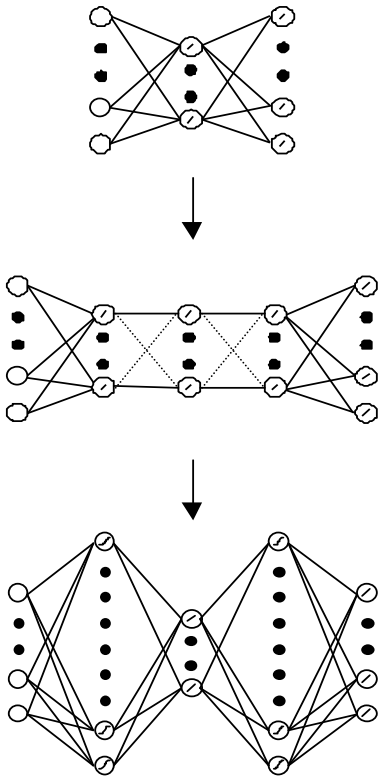


Figure 4, network initialisation

The now obtained diabolo network is trained further with standard back-propagation training algorithms, as described in section 3.2.

4 Classification

The need for classifiers arises from the need for good measures for comparing how well classes are separable. Ideally this measure is the performance of a Bayes classifier. The measures for class-separability, as defined in chapter 2, are not efficient since they are only optimal, in the Bayesian sense, for linear separable distributions.

To gain insight in the quality of the extracted features, and for comparing mappings, classical classification methods are used. Neural network classifiers are not used since they tend to suffer from random initialisations therefore reducing reproducibility. Another practical problem is that neural network classifiers suffer form long training time. The classifiers described in this section are all implemented in the PRTOOLS toolbox [18], which can be used from Matlab [19].

All methods below are listed in order of computational complexity, more information about these classifiers can be found in [15].

4.1 The nearest mean classifier

The nearest mean classifier (nmlc) is probably one of the simplest classifiers. It calculates the Euclidean distance to all class-means and selects the class which class-mean is nearest. Obviously this is not the smartest way to do classification, and most classifiers will perform better. The reason for using this classifier is its speed and simplicity.

4.2 The Mahalanobis classifier

The Mahalanobis classifier or normal densities based quadratic classifier (nqc) is a classifier, which assumes quadratic class separability; i.e. this classifier is a Bayes classifier for classes that have normal distributions.

4.3 The k nearest neighbours classifier

The k nearest neighbours classifier (knnc) is a non-parametric classifier, this means that no assumptions about the true class distributions are made. For a given sample that is to be classified knnc assigns the class-label of the most occurring class among its k nearest neighbours. The distance to neighbours is normally calculated using the Euclidean distance measure. For a large numbers of training samples, and large k, the knnc approaches the performance of the Bayes classifier.

A drawback of the knnc is that, for high dimensional feature-spaces and large numbers of samples, testing becomes extremely time consuming. However, if only the first nearest neighbour is used to classify samples (nnc), the computational burden remains acceptable, although the performance will reduce towards that of a proportional classifier.

5 Experiments

In this chapter some experiments with diabolo networks are shown.

In the first section experiments are shown with artificial feature-spaces. These experiments are purely meant to increase understanding of what diabolo networks can and cannot do, related to the several properties of the networks architecture, and compare diabolo networks with classical linear mapping methods. 2-dimensional feature-spaces are ideal for this purpose because mappings can easily be visualised, thus making the problems more understandable.

In section 5.2 we look at feature extraction for the recognition of handwritten numbers. This application is used to gain understanding of some of the problems that will arise for feature extraction from high-dimensional feature-spaces.

In section 5.3 some preliminary results on image-feature extraction are presented.

5.1 Artificial 2-dimensional feature-spaces

In this section feature-extraction from some simple 2-dimensional class-distributions is visualised for PCA, FLD, unsupervised non-linear mapping (NLPCA) and supervised non-linear mapping (NLFLD). The first 3 experiments are shown for 2-class problems, then 3 experiments are shown for the 3-class problem, finally 3 experiments are shown for classes on a circle.

Of course there is no concrete connection to specific real-world problems and therefore one should not pay too much attention to the meaning of these features.

On all feature-spaces diabolo networks were trained with various compression and reconstruction architectures. The goal of these experiments is to show how mappings are influenced by non-linearity in extraction and reconstruction, illustrate differences between supervised and unsupervised methods and show the fundamental differences between supervised target based mapping with diabolo networks and normal FLD mapping.

In the following results are discussed and some interesting phenomena are visualised. Contour-plots are used to relate the extracted 1-dimensional features to the original 2-dimensional feature-space. These plots can be interpreted as height-maps of the original feature-space with the height being the extracted feature at that position in the original feature-space. This means that points on one contour line all share the same value for the extracted feature. For non-linear mapping methods the extracted features were first ranked before using as ‘height’. This was done to prevent scale problems with visualisation due to the non-linear transformation. Some plots are shown with names like “2-8-1-4-2 mapping”. By this we mean that 1 feature is extracted from a 2-dimensional featurespace using a layer with 8 neurons for extraction and a layer with 4 neurons for reconstruction.

5.1.1 Two linear separable classes

Probably the most fundamental problem in pattern classification is the separation of two classes. In this example features are extracted from two normally distributed classes with an offset

perpendicular to the direction of largest variance. Here we expected PCA to fail because of the relatively large variance in the worst separable direction, in figure 5 a contour-plot is shown for unsupervised PCA mapping, the markers identify the members of the two classes in the original feature-space, the contours identify equal values for the extracted features. In figure 6 a similar contour-plot is shown for supervised FLD mapping.

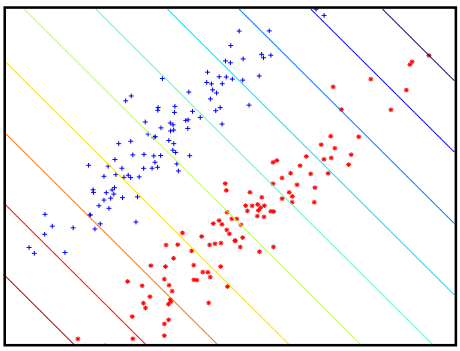


Figure 5, PCA

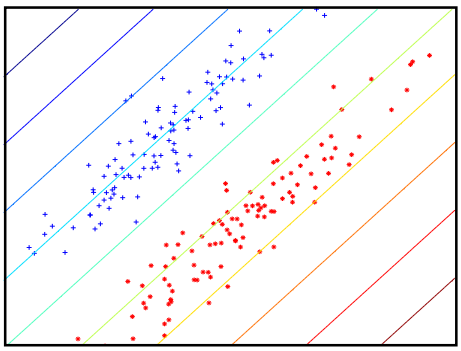


Figure 6, FLD

It is clearly shown that the class information used by FLD mapping results in optimal feature-extraction. Since the features extracted with linear FLD mapping are already optimal the result remains essentially the same for non-linear diabolo networks; it is however interesting to see what features are extracted by unsupervised NLPCA mapping. In our experiments the small networks didn't learn anything more than normal PCA, the large networks however found some more complex curves, minimising distances to the original samples. In figure 7 a contour-plot is shown of unsupervised feature-extraction with a neural network with 8 neurons for extraction and 8 neurons for reconstruction. Although the extracted features are usable for classification the result clearly remains suboptimal.

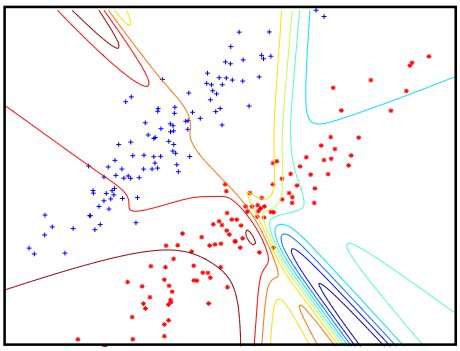


Figure 7, NLPCA with PCA initialisation

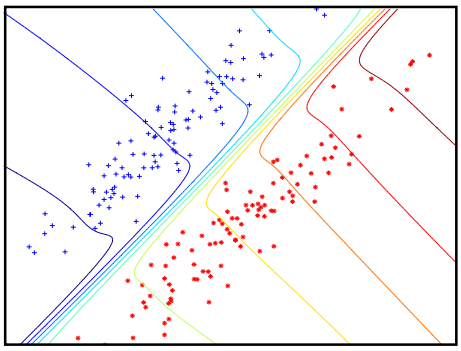


Figure 8, NLPCA with inversePCA initialisation

In this example problems arise due to the initialisation with PCA, which as shown above clearly doesn't help in identifying the different classes. To see if other solutions could be obtained by different initialisation, a second network was trained but now initialised with the worst PCA Feature, i.e. mapping in the direction of smallest variance, the result is shown in figure 8. This example clearly shows that linear mappings are not always the best starting point, especially if no class-information is used.

5.1.2 Highleyman classes

Highleyman classes are constructed of two normal distributed classes with a large difference between the direction of largest variance and the direction of smallest variance. The directions of largest variance, of the two classes, are perpendicular. Since the covariance matrices of both classes are quite different, linear methods are expected to function worse than optimal, this is shown for FLD mapping in figure 9. The result clearly is suboptimal, for classification, since a projection on any one of the axes would have performed better. In this experiment target-trained linear diabolo networks find exactly the same feature-space as the normal linear mappings.

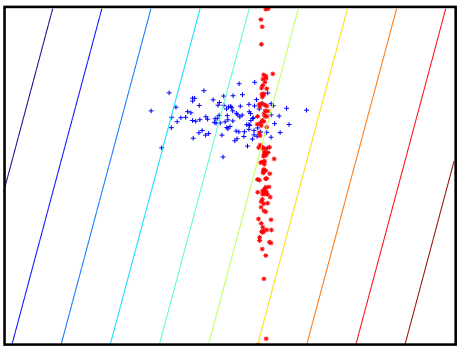


Figure 9, FLD mapping of Highleyman classes

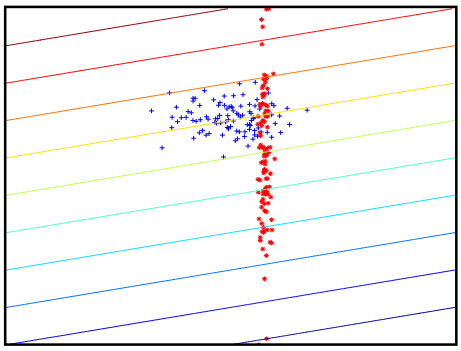


Figure 10, PCA mapping of Highleyman classes

For this example nearest neighbour classification errors (nnc) are around 24% for both the PCA and the FLD mapping.

Non-linear methods perform better. The features extracted with various NLPCA networks achieved between 11% and 20% nnc-errors, while the errors for NLFLD went down to almost 7%. An example of NLFLD-mapping with 8 neurons for extraction, and 8 neurons for reconstruction, is shown in figure 11.

Another nice example is the mapping learned by a diabolo-network with only a non-linear reconstruction layer. This mapping, shown in figure 12, is the optimal linear mapping for classification with a Bayes-classifier.

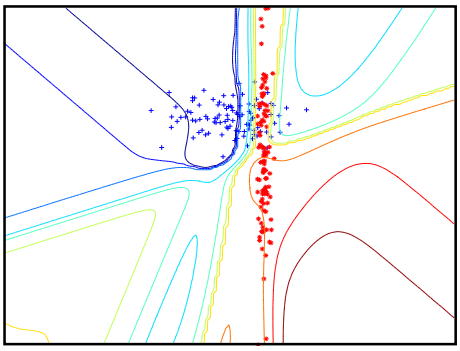


Figure 11, NLFLD, non-linear extraction and reconstruction

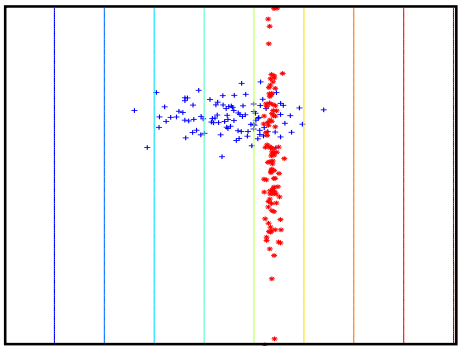


Figure 12, linear extraction with non-linear reconstruction

5.1.3 Non-overlapping banana classes

Another standard 2-class problem is the separation of 2 non-overlapping banana-shaped classes. Since the classes are non-overlapping theoretically classifiers should, when given enough examples, be able to achieve complete separability. However, since the boundary is non-linear, only non-linear feature extraction methods are able to do the job perfectly.

In all experiments the nnc-errors on linearly extracted features were around 30%. As was to be expected the non-linear classifiers performed better, however only the supervised non-linear feature extraction methods performed near optimal, which makes sense because if class information is discarded, for this example, all information about the structure is lost.

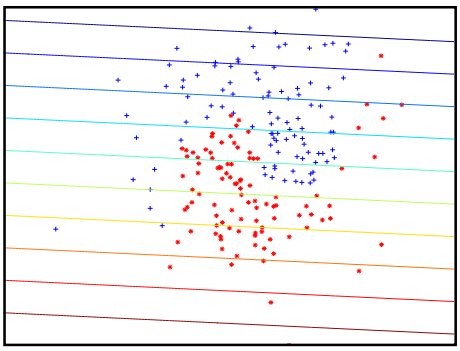


Figure 13, linear mapping

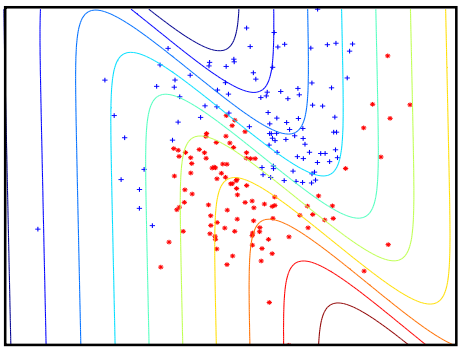


Figure 14, 2 non-linear neurons

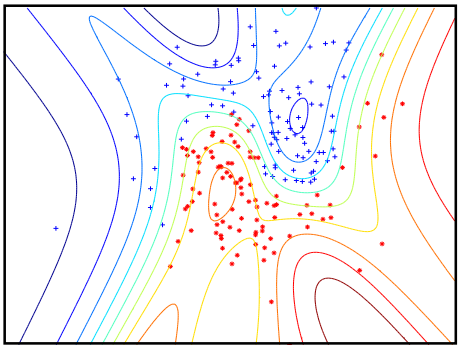


Figure 15, 4 non-linear neurons

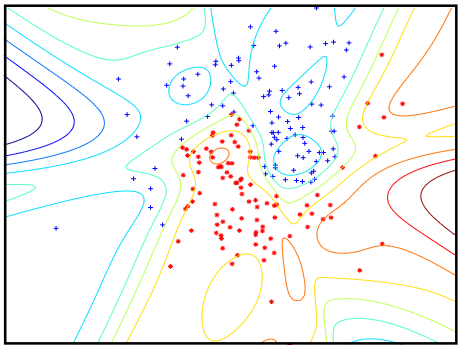


Figure 16, 8 non-linear neurons

It is interesting to see what happens to the extracted feature-space as we increase non-linearity. In figure 13 a contour-plot is shown for a pure linear mapping. In figure 14, 15 and 16 contour-plots are shown for non-linear mappings with respectively two, four and eight neurons in the hidden layer. It is shown that more neurons increase performance by more closely approximating the class boundaries.

In this example all networks were trained without non-linear reconstruction. It was experimentally shown that non-linear reconstruction does not make any significant difference for this 2-dimensional 2-class problem.

5.1.4 Linear separable 3-class problem

In the next experiment feature-extraction is presented for a typical, linear separable, three class problems. Since all three class-means are on a line and all classes have the same orientation, this feature-space allows optimal extraction for both standard FLD and linear diabolo networks.

In figure17 the mapping learned by a linear diabolo network is shown. Standard FLD finds exactly the same mapping. PCA mapping, shown in figure 18, clearly demonstrates the advantage of using class-information.

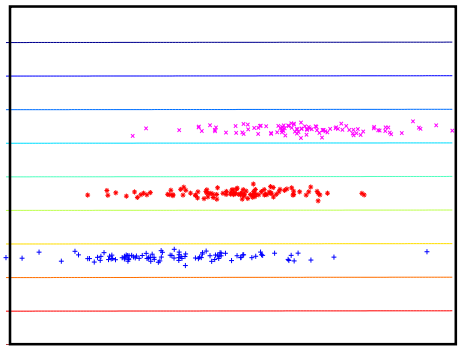


Figure 17, FLD

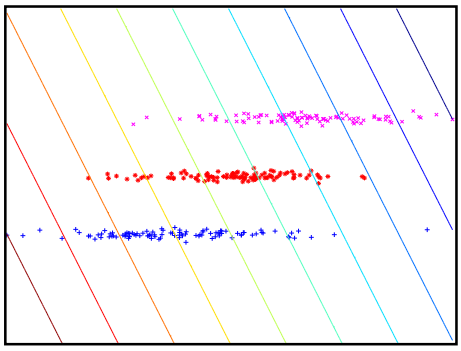


Figure 18, PCA

5.1.5 Linear separable 3-class problem with shifted class

In this experiment one of the classes from the last experiment has shifted to the left, therefore the class means are no longer on a line. In this case standard FLD is expected to provide different solutions than its target-based linear counterpart. The reason for this is the difference between restoring between-class scatter and maximising it. Due to the linear reconstruction the diabolo network cannot completely restore between scatter, therefore forcing a trade-off with minimising within scatter.

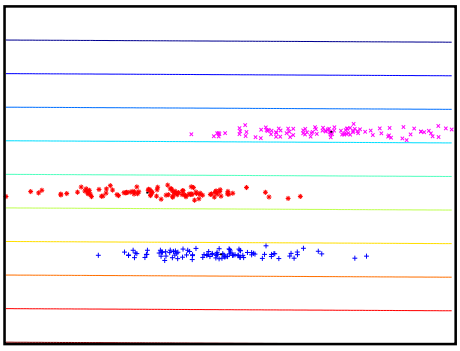


Figure 19, FLD Mapping

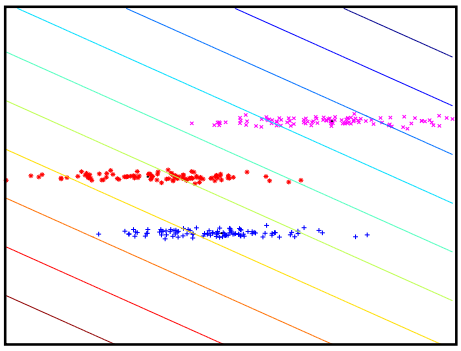


Figure 20, linear diabolo network

To illustrate the differences, normal FLD mapping is shown in figure 19. The supervised mapping learned by a linear diabolo network is shown in figure 20. Due to the impossibility to fit a straight line through the targets, which are shown by the black dots, the linear diabolo network cannot learn optimal feature extraction. Linear feature extraction with non-linear reconstruction does not suffer from this problem. This is illustrated in figure 21, where the mapping of a diabolo network with linear extraction and two non-linear neurons for reconstruction is shown.

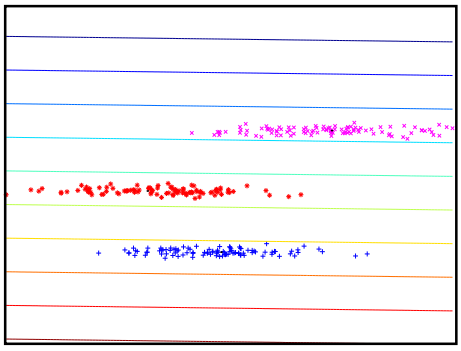


Figure 21, supervised linear mapping with non-linear reconstruction

5.1.6 High triangle

In this experiment three normally distributed classes are used that are on the corners of an imaginary triangle, which has two equal long sides and one shorter. In figure 22 the FLD mapping is visualised, which clearly demonstrates the fact that standard FLD mapping is only optimal if the number of classes is not more then one larger then the dimensionality of the extracted feature-space. Similar problems arise for linear diabolo networks, non-linear networks, however, do not directly suffer from the number of classes. A near optimal linear extraction can already be found with only 2 non-linear neurons used for reconstruction. This good linear mapping is shown in figure 23, the nnc error for this mapping is only 14% where standard FLD leaves us with 34% error.

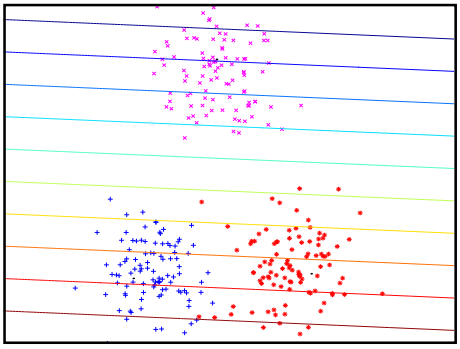


Figure 22, FLD,

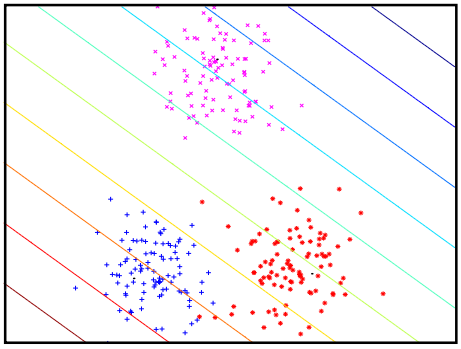


Figure 23, supervised non-linear reconstruction,

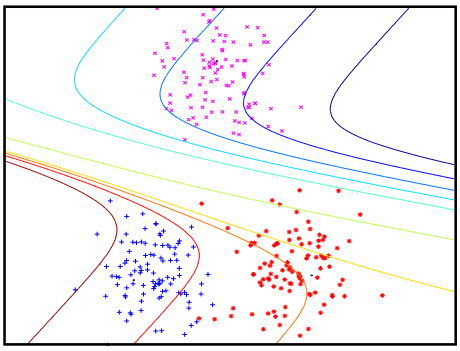


Figure 24, supervised non-linear extraction and reconstruction

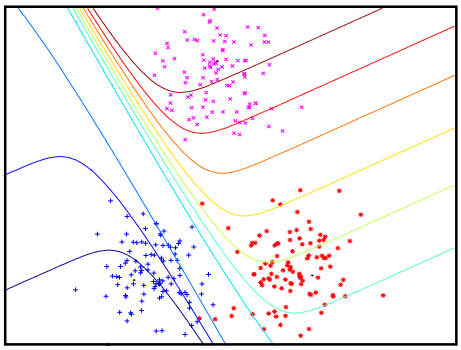


Figure 25, unsupervised non-linear extraction and reconstruction

If non-linear extraction and reconstruction are allowed, feature-extraction by a small diabolo network with 2 neurons for extraction and 2 for reconstruction, shown in figure 24, results in only 3% nnc-errors on the extracted features. If the same network is trained without class information the nnc-error drops to 5%. The associated NLPCA mapping, shown in figure 25, clearly demonstrates that class information is not always needed to achieve near-optimal performance.

5.1.7 Half circle

In the next three experiments, feature-extraction is shown for classes on a circle. This is interesting because a circle is self-intersecting, and therefore theoretically cannot completely be described by a continuous feed-forward diabolo network. The experiments are presented in order of non-linearity of both feature-spaces and mapping methods.

To start of easy, mappings were trained for classes on one half of the circle. In figure 26 and 27 the normal PCA and FLD mappings are shown which were used to initialise the networks.

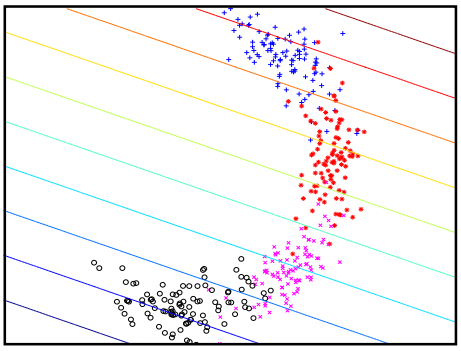


Figure 26, PCA

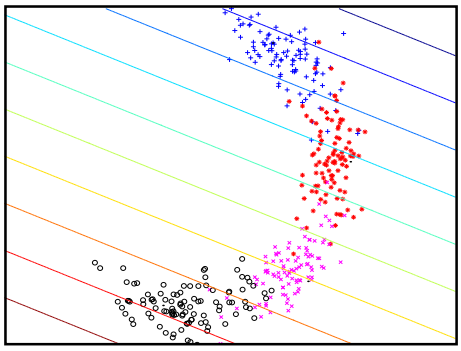


Figure 27, FLD

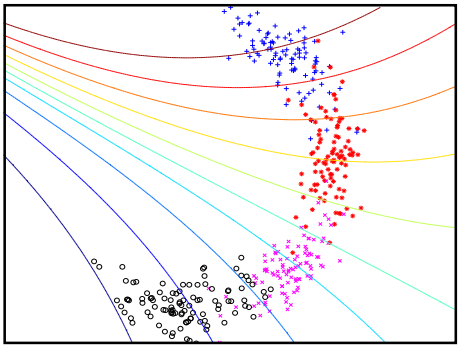


Figure 28, unsupervised 2-2-1-2-2 mapping

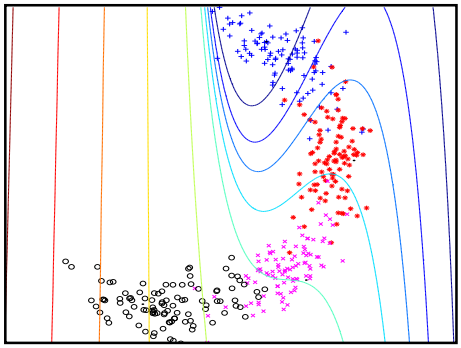


Figure 29, supervised 2-2-1-2-2 mapping

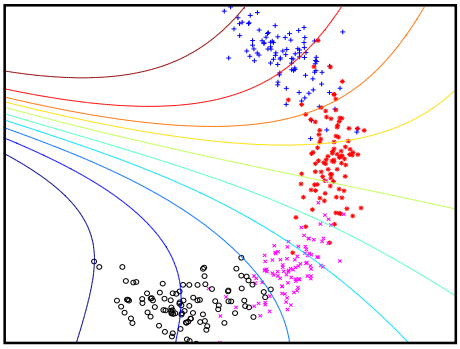


Figure 30, unsupervised 2-2-2-1-2-2-2 mapping

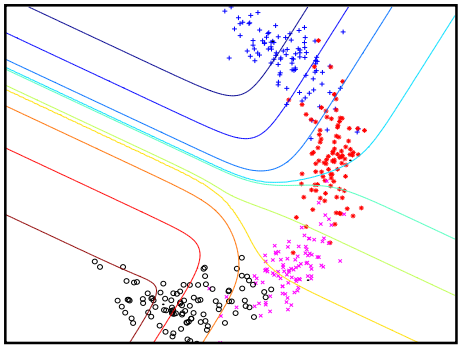


Figure 31, supervised 2-2-2-1-2-2-2 mapping

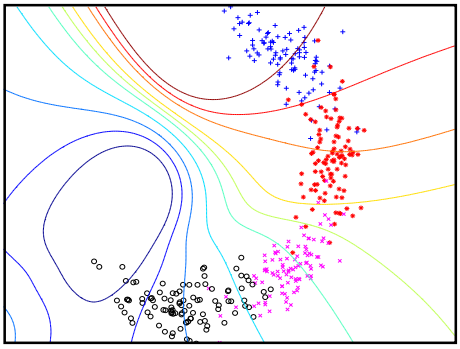


Figure 32, unsupervised 2-4-4-1-4-4-2 mapping

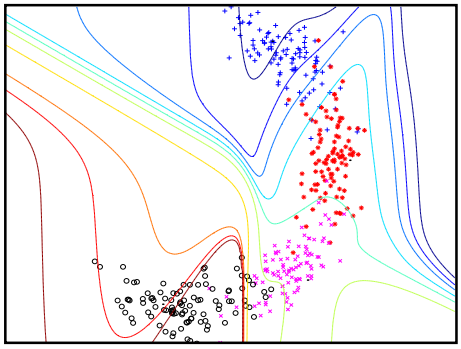


Figure 33, supervised 2-4-4-1-4-4-2 mapping

In figure 28, 30 and 32 NLPCA mappings are shown in order of network complexity. It is interesting to compare these mappings with their supervised counterparts. In figure 29, 31 and 33 the mappings of the same networks are visualised, for supervised training with one target per class. In this experiment NLPCA seems to preserve more global structure outside the classes than NLFLD.

5.1.8 3/4 circle

In this experiment we increased the need for non-linearity by adding two classes in the third quadrant of the circle, used in the last experiment. To see how the different mapping methods react to the extra classes and non-linearity 2 mappings are shown in figure 34 and 35, for a diabolo network with only two neurons. Similar diabolo networks now with 8 neurons for extraction and reconstruction were trained; the results are shown in figure 36 and 37.

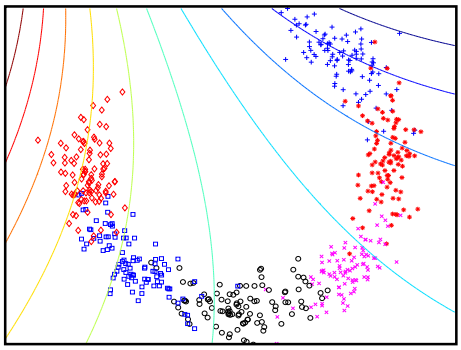


Figure 34, unsupervised 2-2-1-2-2 mapping,

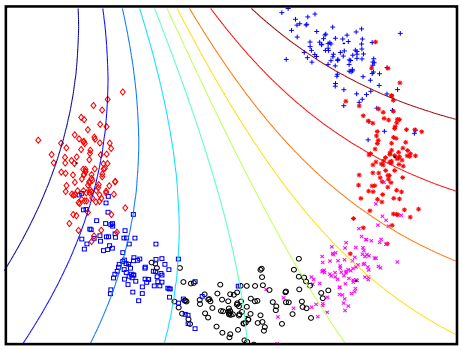


Figure 35, supervised 2-2-1-2-2 mapping

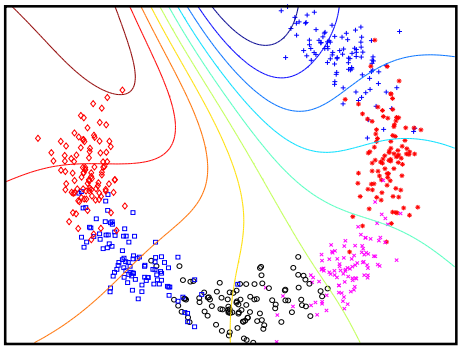


Figure 36, unsupervised 2-8-1-8-2 mapping

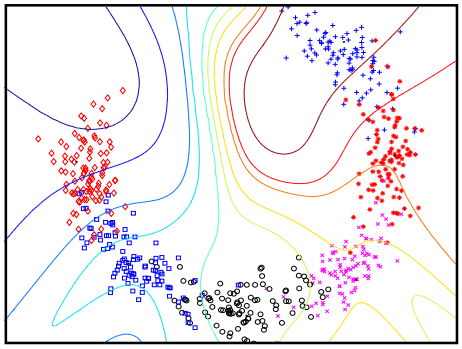


Figure 37, supervised 2-8-1-8-2 mapping

We clearly see that more neurons increase performance for the NLPCA mapping, for the supervised NLFLD the performance also increases but at the same time some loss of generalisation, outside the classes, occurs.

5.1.9 Full circle

Finally two more classes were added in the fourth quadrant of the circle. Typical NLPCA and NLFLD mappings for this feature-space are shown in figure 38 and 39. Both networks used four neurons for extraction and reconstruction.

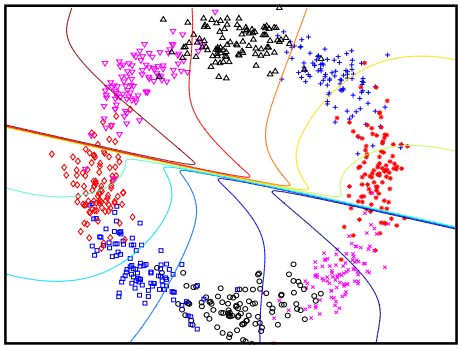


Figure 38, unsupervised non-linear mapping

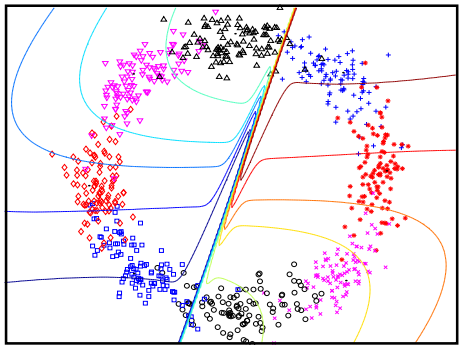


Figure 39, supervised non-linear mapping

It is interesting to see that both networks have two opposite phase-shifts along the radius of the circle. In our experiments we found that these double phase-shifts more often occur supervised non-linear mappings than in for unsupervised mappings. This may be related to the fact that reconstruction of only 8 different targets can be done in a more discontinuous way than the reconstruction of all inputs.

Another nice mapping learned by a diabolo network with 2 layers of 8 neurons for extraction and 2 layers of 8 neurons for reconstruction is shown in figure 40. Here we see that although the diabolo network will never be able to completely reconstruct the circle, it is still able to find a good approximation with only one phase jump.

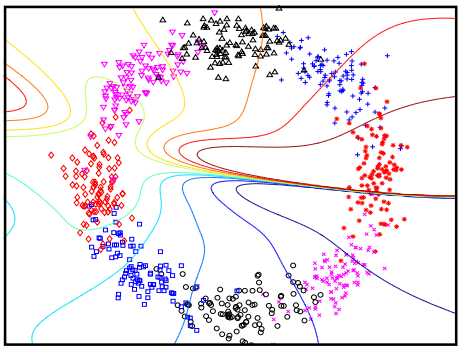


Figure 40, unsupervised, more non-linear mapping

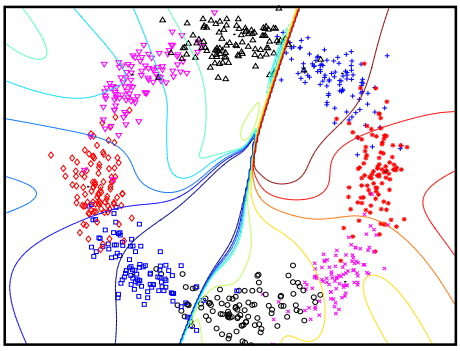


Figure 41, supervised, more non-linear mapping

5.2 Nistdigs, mapping of handwritten numbers

In this section a real-world application in the form of feature-extraction from handwritten numbers is used to investigate feature extraction by diabolo networks.

The dataset used in this section is nistdigs; Dick de Ridder constructed it from the NIST-database for his graduation [16]. Nistdigs contains 2000 16x16 grey-value-images of handwritten numbers (200 per class).

In the following experiments different feature-extraction methods are compared for extracting a small number of features from the original images. To do this we treat the 16x16 images as 256 dimensional feature-vectors. With these 256 dimensional feature-vectors diabolo networks can be trained to extract, and reconstruct the original feature-space.

5.2.1 The dataset

Although feature extraction can be performed directly on the 256 dimensional features, this proves to be extremely time-consuming. To speed up experiments a new dataset was created consisting of the first 32 KL-features of the original images; i.e. the new features are the first 32 principal components. Although our choice for 32 features is just an arbitrary number, we can assume this to be a fair choice since it linearly preserves 85% of the variance in the original images.

Before the 32 KL-features were extracted first a normalisation was done on all images. The images were normalised by first clipping the darkest and lightest 10% of the pixels. Then the mean gray values was subtracted. Finally the gray values were divided by the standard deviation.

The dataset was split to use 150 samples per class for learning and 50 samples per class for testing. Although better classification results can be obtained using all the samples we did not do this since our aim is to compare mapping methods and not directly achieve the smallest classification error. Furthermore it should be noted that the test-samples were not used to estimate the mapping for extracting the 32 KL-features.

5.2.2 Effect of dimensionality on different network architectures

In this experiment the effect of dimensionality of the extracted feature-space, on different network architectures was investigated. All networks were trained using gradient descent with momentum and adaptive learning training. The maximum number of network updates was 5000 epochs. For the supervised mappings targets were used that optimise (8). On all extracted feature-spaces classifiers were trained on the learn-set and tested on the test-set. In figure 42 plots are shown relating the nearest neighbour classification error to the dimensionality of the extracted features for different linear mapping methods. In figure 43 and figure 44 similar plots are shown for respectively unsupervised and supervised mappings. The numbers in the legend are the multiplication factors of the hidden layers of the diabolo networks, nlfld8-8-1-8-8 for instance means that the networks has 2 layers for non-linear extraction and 2 layers for non-linear reconstruction, all with 8 times more neurons than the bottleneck layer.

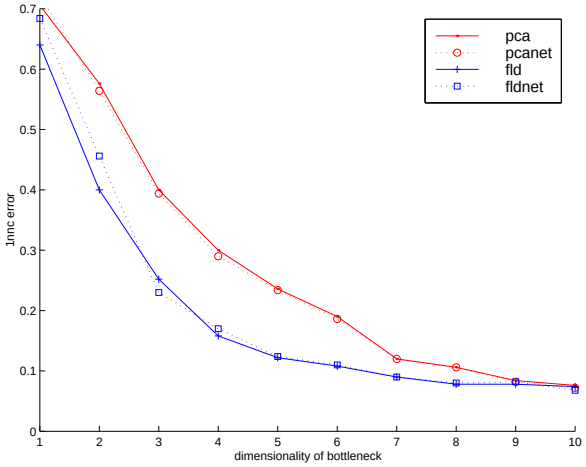


Figure 42, nnc-errors for linearly extracted features

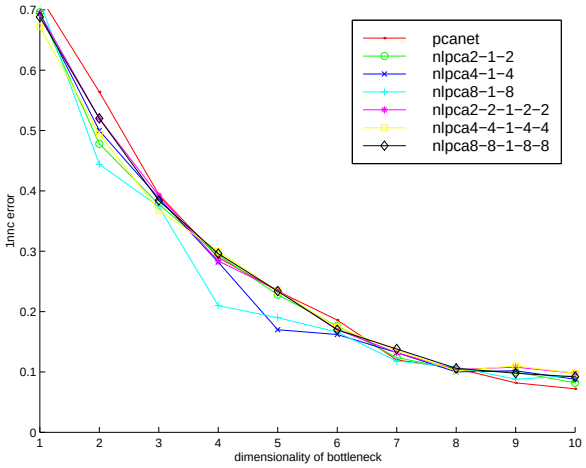


Figure 43, unsupervised feature extraction

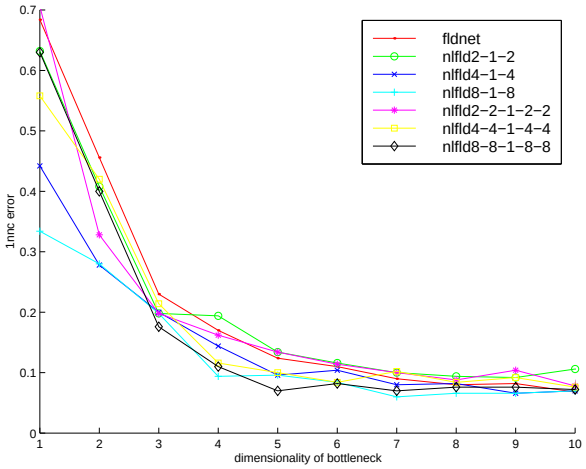


Figure 44, supervised feature extraction

For this application, using a larger bottleneck, results in a lower classification error on independent test samples. The plots in figure 42 clearly show that, for this application, there is little difference in performance between linear diabolo networks and normal FLD mapping. Furthermore, it is shown that, especially for a small bottleneck, dimensionality has larger influence on supervised mapping than unsupervised mapping. Therefore to illustrate differences between the mappings the dimensionality of the extracted feature-space should not be too high.

It should be noted that the above results are derived from one realisation per network architecture. Since the initialisation is partly random these results may change for other realisations.

5.2.3 Initialisation

An interesting question is how our initialisation influences the end-result after training the diabolo network. From various experiments we found that data scaled to the linear region of the hyperbolic tangent sigmoid scaled with a standard deviation of 1.5 combined with noise added to the weights with a standard deviation of half the average absolute weight value, gave good results. To give an idea of how these parameters effect the end result for classification, four different initialisations were tested several times. The results are shown in Table 1. These results are obtained from 23 different training rounds of a diabolo network with 75 neurons in one hidden layer for compression, a 5 dimensional bottleneck and 50 neurons for reconstruction. The networks were trained with gradient descent with momentum and adaptive learning for a maximum number of 5000 network updates. In the first column the initialisation method is shown. The first initialisation is the optimal initialisation used in most experiments, the next two are variations with respectively small added noise and small non-linear scaling. The fourth initialisation is a random initialisation implemented in Matlab's function "newff".

$\sigma_{\text{noise weights}}$, $\sigma_{\text{scaled data}}$	nnc-error (%)	$\sigma_{\text{nnc-error}}$ (%)	$\min_{\text{nnc-error}}$ (%)	$\max_{\text{nnc-error}}$ (%)
0.5 , 1.5 (FLD)	7.9	0.6	7.0	9.0
0.01 , 1.5 (FLD)	8.7	0.8	7.4	10.4
0.5 , 0.03 (FLD)	12.8	0.5	11.8	13.6
Matlab (random)	16.6	6	9.4	29.8

Table 1, effect of network initialisation with linear mapping

It is clearly shown that for this problem linear FLD initialisation provides a good start. Furthermore we see by the small standard deviation of the observed errors, shown in the third column, that the end-result is much more robust then Matlab's random initialisation.

5.2.4 Number of neurons

In this experiment we try to visualise the effect of the number of neurons used for extraction and reconstruction on the quality of the extracted feature-space.

In figure 45 a surface-plot is shown of the nearest neighbour classification error related to the number of neurons used for compression and reconstruction through a 5-dimensional bottleneck layer. The nnc-errors in this plot, represented by the colour and the heights, are an average of two realisations per network architecture. The average difference between the two realisations was less than 3% of the plotted error. Similar plots have been made for the nearest mean classifier and the Mahalanobis classifier, which show roughly the same behaviour.

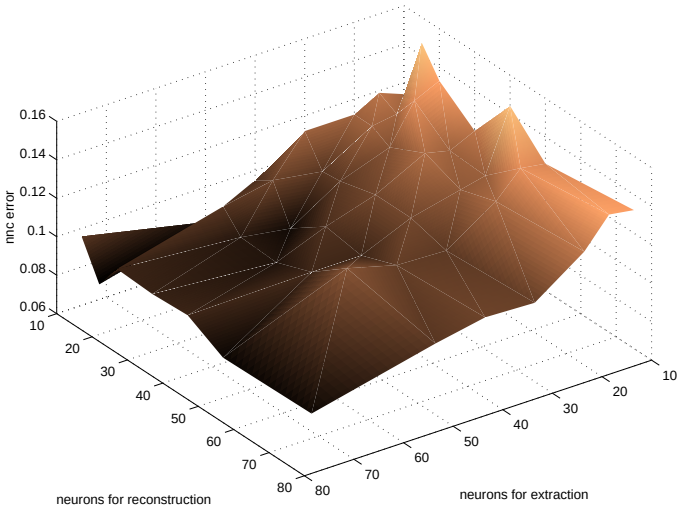


Figure 45, nnc error-surface of extracted 5d features for different numbers of neurons

It is shown that in the case of a 5 dimensional bottleneck layer the reconstruction is relatively unimportant. To see if this is also true for a smaller bottleneck a second surface plot was made for diabolo networks with a 2 dimensional bottleneck. The result, shown in figure 46. Again this plot is an average of two realisations. The average difference was 9% of the plotted errors. It is clearly shown that for smaller bottleneck layers the reconstruction becomes more important. This is explained by the fact that the reconstruction has to learn a curved 2-dimensional surface through all the class-targets, which generally requires more non-linearity for a low dimensional subspace.

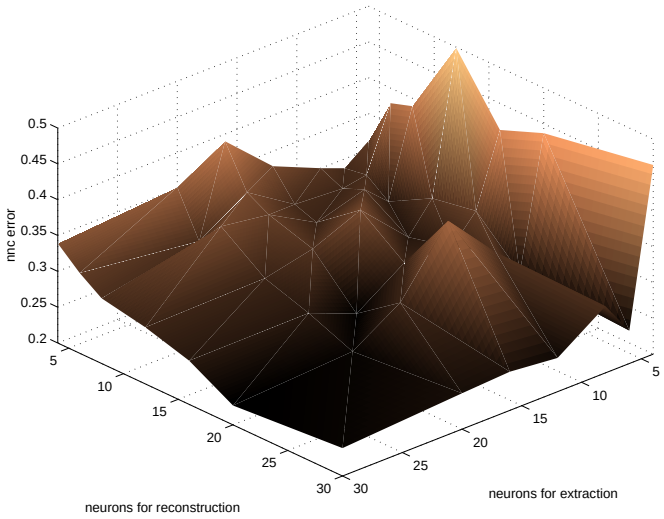


Figure 46, nnc error-surface of extracted 2d features for different numbers of neurons

5.2.5 Effect of class-size

If the number of examples per class becomes small it will generally be more difficult to extract the correct features. To see how class-size effects quality of the extracted feature-space several experiments were performed with several network architectures. In figure 47 an example is shown, for extraction of 5-dimensional features, relating the performance of a nearest neighbour classifier (nnc) to the number of samples used for training the mapping.

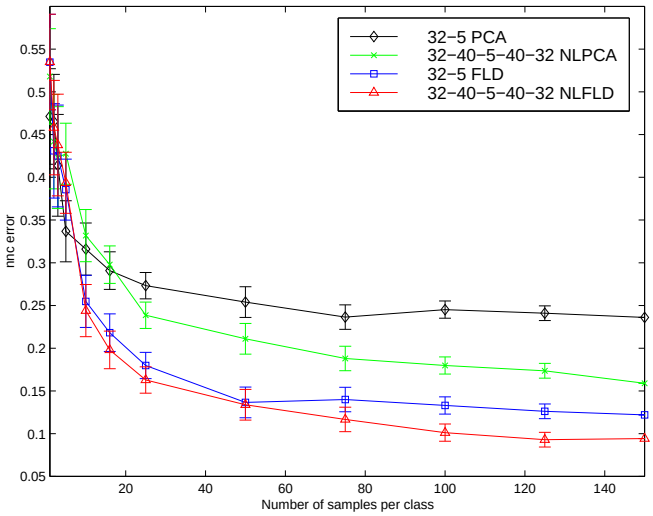


Figure 47, nnc error for 5d feature extraction with different samplesizes

It is clearly shown that feature extraction becomes difficult when the number of examples is small. The reason for this is that due to the small number of training samples there just isn't enough data to make a reliable estimate of the class distributions. Directly related to this is the problem of estimating covariance matrices for principal component analysis and Fisher's linear discriminant analysis. For the diabolo network problems occur in the form of over-training.

5.2.6 Improving performance by adding distorted copies

In the last section it was shown that small sample-sizes have a dramatic effect on the effectiveness of the extracted feature-space for classification. A well-known solution to this problem for classical linear mapping methods is regularisation (see also 2.2.2). Regularisation improves performance by simulating a larger dataset created by the addition of noise to the original samples. For classical linear mapping methods this can be done without any significant computational effort by just increasing the diagonal elements of the covariance matrix by a small amount. For diabolo networks such a nice trick does not exist, we can however make real distorted copies and use them in training. Although this has the drawback that training will become slower, it allows us to use a more advanced model of the variations that occur in natural data.

To see how adding distorted copies can improve feature extraction for the recognition of handwritten numbers, extra samples were created. Applying rotation, scaling and the addition of white noise to the original images created the distorted copies. The angles over which the images were rotated where normally distributed with a standard deviation of 15 degrees. The scaling parameter was also normally distributed with a standard deviation of 7,5% and the added noise was normally distributed with a standard deviation of 30% of that observed in the original images. It should be pointed out that these values are chosen because they looked good and are only meant as an example, other values and transformations may give better results, this, however, is beyond the scope of our research.

Several experiments were performed to compare mappings trained with distorted copies to normal feature-extraction. In figure 48, plots are shown for the performance of supervised 2-dimensional feature-extraction related to the number of original samples used in training. For this example we used a mapping to 2-dimensional because, as shown in section 5.2.2, for low

dimensionalities the differences between the different mapping methods tend to be larger. Another advantage is that the 2-dimensional features allow for easy visual inspection of the extracted feature-space. In figures 49 to 52 plots are shown for the 2-dimensional features extracted from all test samples by the supervised mapping methods trained from 100 original samples per class.

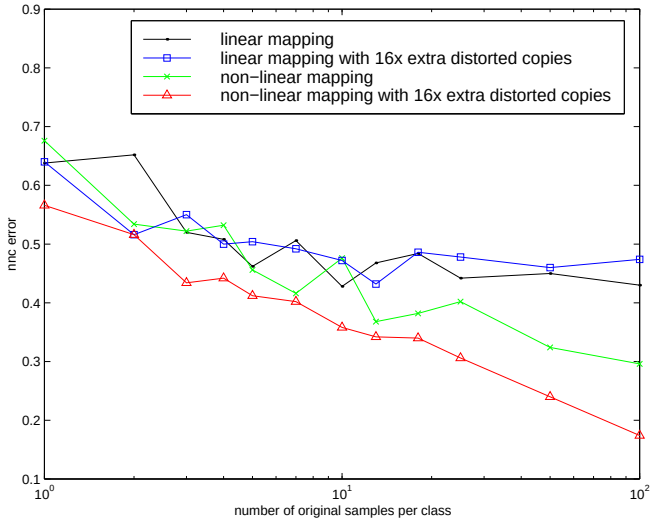


Figure 48, nnc errors for 2d feature-extraction with distorted copies

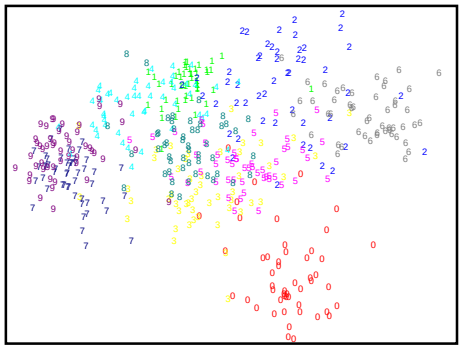


Figure 49, normal FLD trained with 100 samples per class

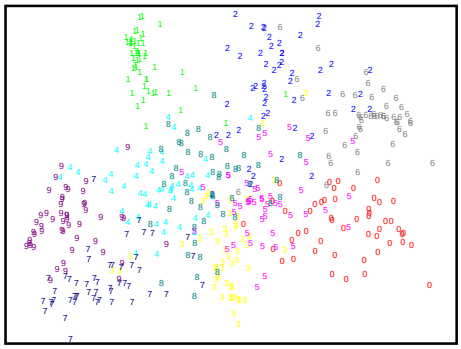


Figure 50, non-linear mapping trained with 100 samples per class

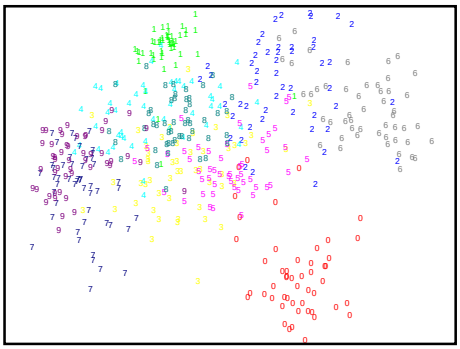


Figure 51, FLD trained with distorted copies

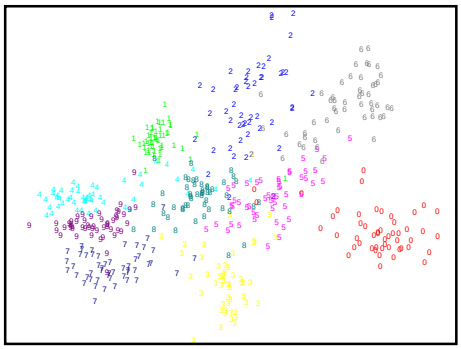


Figure 52, non-linear mapping trained with distorted copies

In general, well chosen distorted copies can increase performance for all mappings, especially when only a small number of samples is available. In this experiment the performance increased more for the non-linear mappings then for the linear mappings. An explanation for this may be that the linear mappings cannot adapt to rotation and scaling.

5.2.7 Effect of non-linear reconstruction

In section 5.1 some examples were shown where non-linear reconstruction improved linear feature extraction. To see if the linearly extracted FLD features could be improved by applying diabolo networks, with non-linear reconstruction, several different networks were trained. In figure 53 the classification results are shown related to the number of non-linear reconstruction neurons, for different dimensionalities of the extracted feature-space. The dotted lines are the average errors made by classical Fisher's Linear Discriminant analysis.

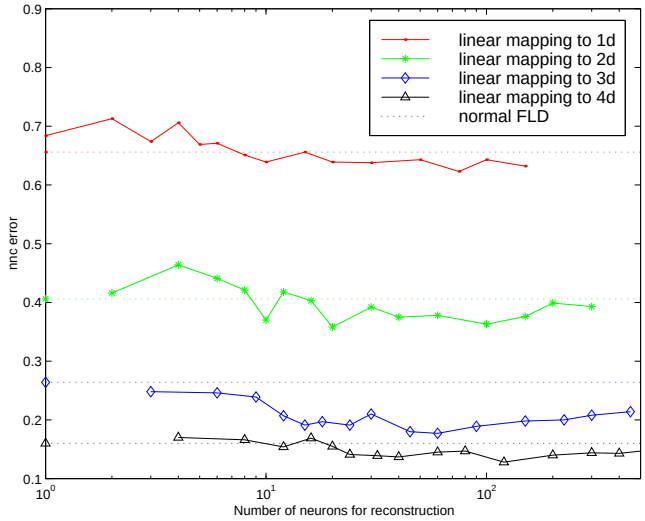


Figure 53, supervised linear extraction with non-linear reconstruction

It is shown that diabolo networks can improve linear feature extraction, for the recognition of handwritten numbers. However, if the number of non-linear neurons is too small the resulting mapping often performs worse than classical FLD. This is partly explained by the fact that addition of random noise to the copied neurons may cause the network to end up in worse local minima. However, the most important reason is the fundamental difference between optimising class-separability through minimisation of distances to targets and direct maximisation of criteria for class-separability.

5.2.8 Targets

Directly related to reconstruction and the architecture that is needed to do it, are the targets themselves. In some early experiments much time was spent trying to find the best targets for feature extraction. Several algorithms were designed for finding targets optimising several criteria of class-separability. In our experiments, however, we found no fundamental differences, with respect to classification, between criteria (8), (9) and (10), applied to the calculation of targets.

An early approach, for the selection of targets, was to randomly select another sample from the same class as target for each input sample. Although this method did increase class separability compared to PCA it performed much worse then the methods using one target per class.

More advanced clustering methods like k-means and k-centres were also tested, but results found little or no change in performance for classification of handwritten numbers. Only for image segmentation of complex classes we found some examples where clustering into more targets per class might be useful.

5.3 Some preliminary results on image-feature extraction

To handle a selective image database search, low-level properties in images can be calculated in a local neighbourhood. Algorithms that calculate such low-level properties are called filters. We can think of a filter as an operation that performs a measurement on the local neighbourhood around each pixel. If more filters are applied to an image each pixel has its own feature-vector, built from all measurements, which can be used to classify the pixels, finally resulting in image segmentation.

Many special purpose applications have been developed which apply problem specific filters. For well-defined tasks a small number of these filters often is sufficient to characterise different image regions. However if the important features are not previously known, either because there is no expert on the subject, because the problem is too complex, or because examples of the objects are not given in advance, all one can do is apply more filters and search for a useful combination.

In this section some preliminary results are discussed on image-feature extraction for application in image database search. It should be emphasize that these experiments are only a first exercise meant to illustrate where possible further research may be going. Although some results are promising, much work still remains to be done.

5.3.1 Segmenting Lena

In this experiment a 256x256 image of Lena was used. The image is shown in figure 54. From this image 4 Intensity, 9 DCT, 8 Gabor, 4 Wavelet and 12 Colour features were calculated in a local 9x9 window around each pixel. From this initially 37 dimensional feature-space new features were extracted using several mapping methods. In [17] some experiments were described showing that diabolo networks can improve feature extraction, compared to standard PCA and FLD mapping, for classification of manually selected regions of the classes skin, the hat, the boa, hair and the background. However, due to deadlines, the result was only derived from a small number of realisations. Another problem was the selection of learn- and test-regions. Since both regions spanned roughly the same amount of surface the results did not reflect the performance that could be obtained from training on the whole image. Furthermore, no pixels near class-boundaries were used, therefore possibly oversimplifying the problem.

To obtain more reliable results new experiments were done. The main idea of these experiments was to calculate a leave-one-out-like estimate for different test regions.

For each realisation the user-defined regions of the classes skin, the hat, the boa, hair and the background, shown in figure 55, were clustered into learn, validation and test regions. The clustering was done using the k-centres algorithm in PRTOOLS [18] for each class. For each realisation 30 centres were used for the learn region, 3 centres were used for validation and one was used for testing. After clustering, two 8-connected erosions were performed on learn, validation and test regions, thus ensuring that the local 9x9 windows did not overlap. From the learn-regions 200 samples per class were selected for learning 20 samples per class were selected for validation. Training was ended after a maximum number of 10000 epochs or if the performance on the validation set decreased over 100 epochs. If training had not reached 10000 epochs

and the performance had not decreased for three consecutive training rounds it was re-started with other samples from learn and validation sets.

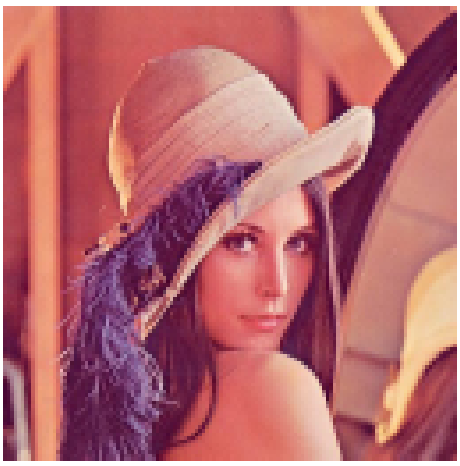


Figure 54, Lena

Figure 55, Labelled classes

In Table 2 the average classification errors are shown for different network architectures. All networks used one non-linear layer for extraction and reconstruction with 6 times the number of neurons in the bottleneck layer.

method	nmlc error (%)	nqc error (%)	nnc error (%)
PCA, 2d	54	46	49
NLPCA, 2d	43	41	37
FLD, 2d	26	19	25
NLFLD, 2d	13	13	19
PCA, 3d	46	30	31
NLPCA, 3d	38	26	26
FLD, 3d	18	15	18
NLFLD, 3d	14	11	12

Table 2, Average percentage of errors on test-regions for different realisations

The errors in Table 2 are an equal-weighted average for all classes. It is interesting to see how these errors are divided among the five classes. In Table 3 the errors made by the Mahalanobis classifier on the extracted 2-dimensional feature-vectors are shown for the individual classes. To get good estimates the experiment was repeated several times, using different learn and test regions, resulting in 95% certainty regions of less then 2% errors. Similar results were obtained for other classifiers and mappings.

	face/body	hat	boa	hair	background
PCA, 2d	71	32	34	44	48
NLPCA, 2d	63	26	22	48	48
FLD, 2d	34	11	4	5	43
NLFLD, 2d	19	11	5	6	23

Table 3, Average percentage of errors per class using the Mahalanobis classifier

To give an idea of how the extracted features perform for image segmentation some typical realisations are shown, for the different mapping methods, in figure 56 to 59. All images were segmented using the Mahalanobis classifier. The reader should keep in mind that these are just four realisations and the results may change for other initialisations and training samples.

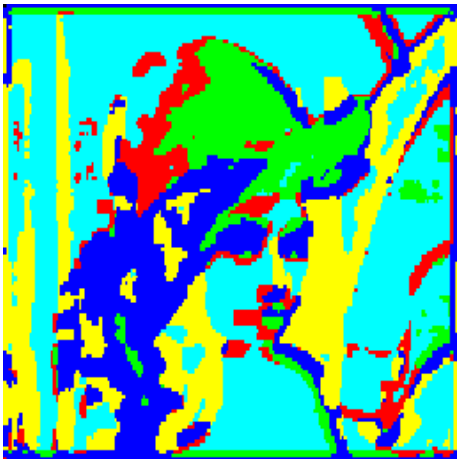


Figure 56, unsupervised linear mapping

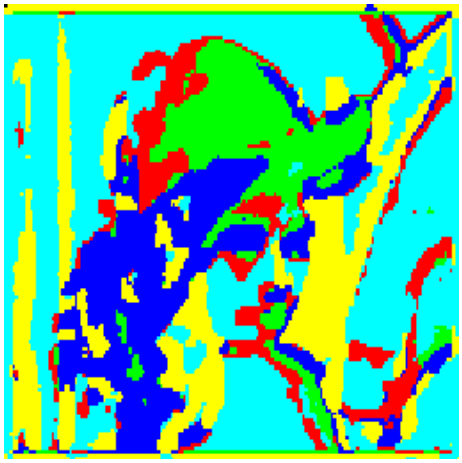


Figure 57, unsupervised non-linear mapping

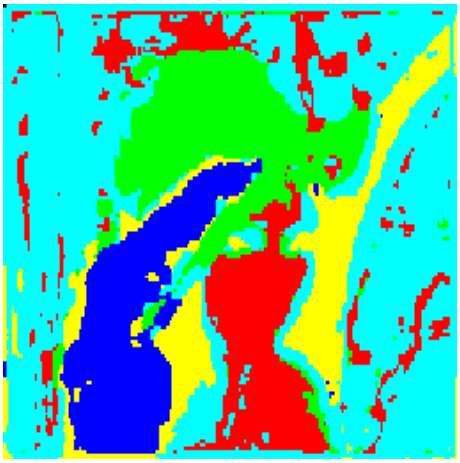


Figure 58, supervised linear mapping

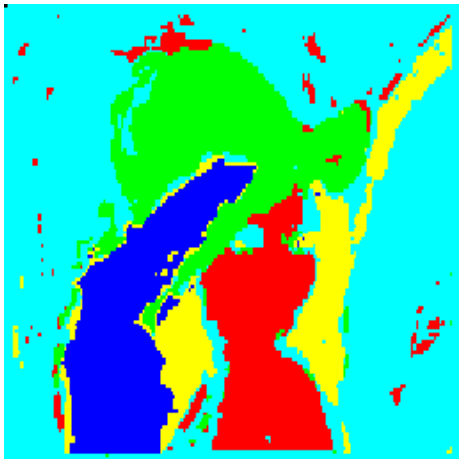


Figure 59, supervised non-linear mapping

It is shown that non-linear feature-extraction improves classification results mainly by increasing separability for the body and the background. These classes are probably more difficult for linear mappings because respectively the body has little textural features that are not also in the background and the background actually is a combination of more subclasses, thus making these classes less likely to be linearly separable.

5.3.2 Delft-images

In the last section, an approach to feature extraction for image segmentation was presented. To see if this type of feature extraction could be applied for a large number of images a database was created containing over 600 'real-world' images of Delft.

In some images classes, such as grass, air, water, trees & bushes, stone and wood, were manually selected. Again using the same filters as applied to the segmentation of the Lena image, new features were extracted using several mapping methods.

Initially our extracted features showed little generalisation to images that where not used in training. To get a better generalisation without having to go through the trouble of labelling more images we came up with the idea of combining supervised and unsupervised training. Training a diabolo network using both samples that should be mapped onto their class-assigned targets and samples that are to be mapped onto themselves can do this. To see if performance could be increased by combining methods, several networks were trained with different numbers of labelled and unlabelled samples.

In figure 60 plots are shown for the performance of the nearest neighbour classifier on independent test samples related to the number of supervised and unsupervised learn samples used in training. All diabolo networks were trained to map the original 37 dimensional feature-vectors onto a 7 dimensional subspace using 42 neurons in the hidden layers for extraction and reconstruction. The bottleneck was 7 dimensional because this was the average of the two estimates of the intrinsic dimensionality, calculated for a large number of samples from all images.

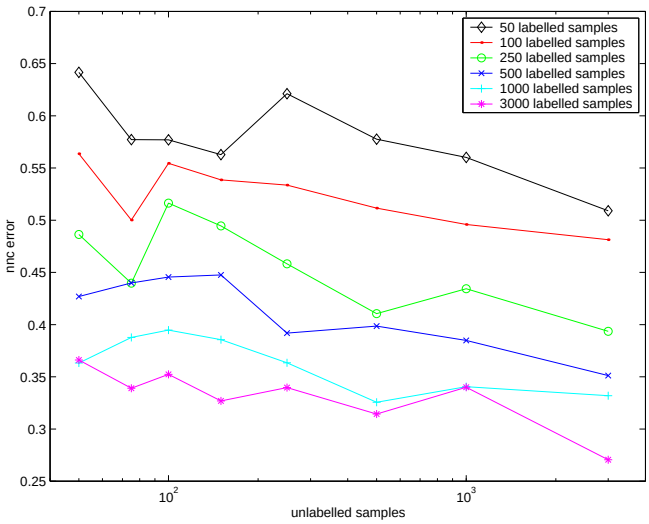


Figure 60, effect of combining supervised and unsupervised samples on non-linear mapping

It is shown that adding unlabelled samples can increase classifier performance on the extracted features. However, adding labelled samples is much more effective for increasing classification performance then adding unlabelled samples. Therefore increasing the number of labelled samples seems the appropriate choice. However, since manual labelling of images is extremely time-consuming we may not want to do this. A possible alternative may be to use distorted copies, like in section 5.2.6. This however has not yet been tested for large image databases.

6 Conclusions and discussion

6.1 Conclusions

A new supervised feature extraction method has been designed which can effectively extract features from a much broader range of class distributions than classical Fisher's linear discriminant mapping. In general the reduction of dimensionality performed by non-linear mapping can aid in improved speed and performance for all distance based classifiers.

A new method for network-weights initialisation has been developed which can be applied to supervised and unsupervised diabolo networks. Experimentally we've shown network training with this initialisation to be faster, more robust and less sensitive to local minima than standard random initialisations.

In several experiments non-linear feature extraction methods were shown to outperform both classical principal component analysis and statistical discriminant mapping. However, not only non-linear feature extraction can be improved. Experimentally it was shown that linear feature extraction could also be improved, compared to classical linear methods, by allowing non-linear reconstruction of the original feature-space. The reason for this is that non-linear reconstruction causes higher-order statistics of the extracted feature-space to be taken into account, therefor preserving more information then classical linear reconstruction methods.

6.2 What remains to be done

Until now only a good comparison was made between classical feature extraction and feature extraction performed by diabolo networks. There is, however, a much wider range of feature-extraction methods such as self-organising mappings, Sammon's mapping and independent component analysis. It would for instance be interesting to see how self-organising mapping scales to high dimensional problems compared to diabolo networks. Another interesting experiment would be to compare subspaces found by linear and non-linear independent component analysis to those found by diabolo networks.

In our experiments we found that if the network was given enough freedom the choice of targets became relatively unimportant. If we do not care for preserving original spatial relations in the reconstructed feature-space, the diabolo networks reconstruction architecture could be altered to perform direct classification. It would be interesting to compare such a network's generalising capabilities, with respect to unknown classes, to a normal diabolo network that tries to preserve spatial relationships.

Another remaining question is the choice for the number of neurons and hidden layers that should be used. In general the degrees of freedom should be as low as possible, since each extra neuron increases the computational complexity and the chance of over training. However aside from some rough estimates for the intrinsic dimensionality most parameters still have to be tuned on a trial and error basis.

Non-linear diabolo networks can extract good image features. However, if we want to apply our feature extraction methods to large 'real-world' image-databases more work needs to be done on increasing the number of learn samples. This can be done in an extremely time-consuming way by more manual labelling. However, if we can model class-invariant transformations, this can be used to improve feature extraction by artificially increasing the number of training samples. Furthermore, to decrease computational demands, feature-selection methods should be applied to remove redundant initial features before extracting more complex features.

7 References

Literature

[1] K. Messer, J.Kittler and M. Kraaijveld : Selecting features for neural networks to aid an Iconic search through an image database, IEE 6th International Conference in Image Processing and its Applications pages 428-432, University of Surrey, 1997.

[2] J. Karhunen et al. : A class of neural networks for Independent Component Analysis, IEEE transactions On Neural Networks, Vol.8 nr.3, May 1997.

[3] T. Kohonen : Self-Organising Maps, Springer-Verlag Berlin Heidelberg, 1995.

[4] J.W. Sammon Jr. : A nonlinear mapping for data structure analysis, IEEE Transactions on Computers, C-18:401-409, 1969.

[5] M.Flickner et al.: Query by image and video content: The QBIC system. Computer, 28(9):23-31, September 1995.

[6] J.Bach et al. : The virage image search engine: An open framework for image management. In SPIE Electronic Imaging: Storage and Retrieval for Image and Video Databases IV, volume 2670, pages 76-87, San Jose, California, February 1996.

[7] E.C. Malthouse : Limitations of nonlinear PCA as performed with generic neural networks, IEEE transactions On Neural Networks, Vol.9 nr.1, January 1998.

[8] E. Oja: Data Compression, Feature Extraction, and Autoassociation in Feedforward Neural Networks, Elsevier Science Publishers B.V., 1991.

[9] S. Russel & P. Norvig: Artificial intelligence: A modern approach, pag.595, Prentice-Hall, 1995.

[10] K.Fukunaga : Introduction to statistical pattern recognition, Academic Press, New York, 1990.

[11] Yong-Qing Cheng et al. : Optimal fisher discriminant analysis using the rank decomposition, Pattern Recognition, Vol.25 nr.1, pag.101-111, 1992

[12] C.S. Cruz, J.R. Dorronsoro : A nonlinear discriminant algorithm for feature extraction and data classification, IEEE transactions On Neural Networks, Vol.9 nr.6, November 1998.

[13] D. Torrieri : The eigenspace separation transform for neural-network classifiers, Neural Networks Vol.12 nr.3 pag. 419-427, April 1999.

[14] B. Müller, J.Reinhardt : Neural Networks, An Introduction, Springer-Verlag Berlin Heidelberg, 1990.

[15] E.Backer, R.P.W.Duin : Statistische patroonherkenning , Delftse Uitgevers Maatschappij, Delft, 1989.

[16] D. de Ridder, M.Sc.Thesis, Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, February 1996.

[17] E.C.D. van der Werf, R.P.W. Duin : Improved image features by training non-linear diabolo networks, To appear in proceedings of Asci'99 5th annual Conf of the Advanced School for Computing and Imaging (Heijen, NL, June 15-17) , ASCI, Delft, 1999.

Software

[18] R.P.W. Duin : Pattern Recognition tools (Prtools 2.1), Pattern Recognition Group, Department of Applied Physics, Faculty of Applied Sciences, TU Delft, Netherlands

[19] The Mathworks Inc : Matlab 5.2, 24 Prime Park Way, Natick, MA 01760-1500