Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

Fault Isolation By Manifold Learning

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan i Linköping
av

Mårten Thurén

LiTH-ISY-EX--10/4430--SE

Linköping 2010

Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

Fault Isolation By Manifold Learning

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan i Linköping
av

Mårten Thurén

LiTH-ISY-EX--10/4430--SE

Handledare: **Anna Pernestål**
ISY, Linköpings universitet

Examinator: **Erik Frisk**
ISY, Linköpings universitet

Linköping, 21 June, 2010

Avdelning, Institution
Division, Department

Division of Vehicular Systems
Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Datum
Date

2010-06-21

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English
☐ ________________

Rapporttyp
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ ________________

URL för elektronisk version

http://www.vehicular.isy.liu.se
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-281055

ISBN
—

ISRN
LiTH-ISY-EX--10/4430--SE

Serietitel och serienummer
Title of series, numbering

ISSN
—

Titel
Title

Felisolering Genom Mångfaldsinlärning
Fault Isolation By Manifold Learning

Författare
Author

Mårten Thurén

Sammanfattning
Abstract

This thesis investigates the possibility of improving black box fault diagnosis by a process called manifold learning, which simply stated is a way of finding patterns in recorded sensor data. The idea is that there is more information in the data than is exploited when using simple classification algorithms such as k-Nearest Neighbor and Support Vector Machines, and that this additional information can be found by using manifold learning methods.

To test the idea of using manifold learning, data from two different fault diagnosis scenarios is used: A Scania truck engine and an electrical system called Adapt. Two linear and one non-linear manifold learning methods are used: Principal Component Analysis and Linear Discriminant Analysis (linear) and Laplacian Eigenmaps (non-linear).

Some improvements are achieved given certain conditions on the diagnosis scenarios. The improvements for different methods correspond to the systems in which they are achieved in terms of linearity. The positive results for the relatively linear electrical system are achieved mainly by the linear methods Principal Component Analysis and Linear Discriminant Analysis and the positive results for the non-linear Scania system are achieved by the non-linear method Laplacian Eigenmaps.

The results for scenarios without these special conditions are not improved however, and it is uncertain wether the improvements in special condition scenarios are due to gained information or to the nature of the cases themselves.

Nyckelord
Keywords

Fault Diagnosis, Support Vector Machines, k-Nearest-Neighbor, Principal Component Analysis, Linear Discriminant Analysis, Laplacian Eigenmaps

Abstract

This thesis investigates the possibility of improving black box fault diagnosis by a process called manifold learning, which simply stated is a way of finding patterns in recorded sensor data. The idea is that there is more information in the data than is exploited when using simple classification algorithms such as k-Nearest Neighbor and Support Vector Machines, and that this additional information can be found by using manifold learning methods.

To test the idea of using manifold learning, data from two different fault diagnosis scenarios is used: A Scania truck engine and an electrical system called Adapt. Two linear and one non-linear manifold learning methods are used: Principal Component Analysis and Linear Discriminant Analysis (linear) and Laplacian Eigenmaps (non-linear).

Some improvements are achieved given certain conditions on the diagnosis scenarios. The improvements for different methods correspond to the systems in which they are achieved in terms of linearity. The positive results for the relatively linear electrical system are achieved mainly by the linear methods Principal Component Analysis and Linear Discriminant Analysis and the positive results for the non-linear Scania system are achieved by the non-linear method Laplacian Eigenmaps.

The results for scenarios without these special conditions are not improved however, and it is uncertain wether the improvements in special condition scenarios are due to gained information or to the nature of the cases themselves.

Sammanfattning

I det här examensarbetet undersöks huruvida black box-feldiagnos kan prestera bättre m.h.a. metoder av typen manifold learning. Manifold learning innebär kort sagt att hitta mönster i data. Idén är att det finns information i data som inte nyttjas av enkla klassificeringsalgoritmer som k-Nearest Neighbor eller Support Vector Machines, och att denna information kan hittas med manifold learning-metoder.

Den här idén testas därför på data från två olika system som man kan göra feldiagnos på: En Scania-motor och det elektriska systemet Adapt. Två lineära och en icke-lineär manifold learning-metoder används: Principal Component Analysis och Linear Discriminant Analysis (lineära) samt Laplacian Eigenmaps (icke-lineär).

Vissa förbättringar uppnås för fall med vissa antaganden. De förbättringar som uppnås för olika system överenstämmer i sin linearitet med de metoder som uppnår förbättringarna. De förbättringar som uppnås för det elektriska och relativt lineära elektriska systemet uppnås framförallt av de lineära metoderna Principal Component Analysis och Linear Discriminant Analysis, medan motsvarande förbättringar för Scaniamotorn uppnås av den icke-lineära metoden Laplacian Eigenmaps.

Resultaten för fall utan speciella antaganden är dock inte positiva, och det är osäkert huruvida de förbättringar som uppnås för fallen med antaganden beror på någon informationsförtjänst eller på fallens natur.

Acknowledgments

A number of people contributed in one way or another to this now almost year-long journey. My supervisor Anna Pernestål through her good advice, good criticism, good mentoring and positive spirit. My examiner Erik Frisk by letting me write my master thesis at the department for Vehicular Systems and by having the patience which let me finish the thesis in a satisfying way. My opponent Rickard Carlsson by his constructive opinions about my thesis. My friends and family by giving me the less academic kind of support which is just as necessary, and by contributing one small piece of code.

I am thankful to all of you.

Contents

Chapter 1

Introduction

1.1 Background

One thing can certainly be said about machines - they rarely work quite like we intend them to, and if they do, they may stop doing so any minute. That is the problem addressed by fault diagnosis: To identify the cause and location of a fault when one occurs.

A common way of doing diagnosis is Model Based Diagnosis (MBD). To do MBD is to model the machine, let the model run parallell to the machine and compare sensor readings from the machine with the equivalents in the model. If the model is good enough it is possible to infer both that a fault has occured and which fault it is [9].

The previously assumed "good enough" model might not always be available however. As machines grow in complexity it becomes harder and more time consuming to model them [10]. This makes it interesting to see what results can be achieved by using the sensor data paired with the correct fault state but without having a physical model of the machine.

This type of fault diagnosis described in the previous paragraph can be refered to as black box or dark grey box fault diagnosis, where the shade of the box symbolizes how much of the machine we understand. MBD would then be white or light grey box learning. The black box perspective of fault diagnosis transforms diagnosis into a pattern recognition problem. Each recorded state of the machine becomes an element in a pattern, resulting in patterns for the fault free and faulty cases. This gives us a wide range of methods from pattern recognition to use for fault diagnosis.

Due to the nature of black box fault diagnosis, and for reasons explained later in this thesis, manifold learning will be used as a preprocessing step for simpler classification methods such as kNN and SVM. Simply stated: Manifold learning finds low dimensional structures caused by constraints on the sensor data. Given a system and a possible fault on that system - not all data combinations are possible. The behavior of different systems with possible faults allow different data combinations. This difference will be exploited to improve classification results.

1.2 Objective

The objective of this thesis is to evaluate the performance of different manifold learning methods, such as principal component analysis and laplacian eigenmaps, on data from real machines and compare the performances of these methods using very simple classification methods. A standard of comparison is given in Section , but intuitively, a better result means fewer incorrect classifications.

The best performing manifold learning method is determined by using two tracks - SVM and kNN, each starting without any modifications. SVM and kNN are simple pattern classification algorithms and are not doing any manifold learning on their own. Increasingly advanced manifold learning methods are then applied to the data before running SVM and kNN. The methods used are:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Laplacian Eigenmaps (LapEig)

Two sets of data are used for performance evaluation; data from a Scania truck engine [3] and data from the Adapt dataset [13, 14], these consist of sensor data associated with either a certain fault or no fault. These sets of data has been chosen to allow evaluation of classification methods in different domains.

1.3 Previous Work on Fault Diagnosis Through Manifold Learning

The cross-section between fault diagnosis and manifold learning in literature is small, but existing. Two commonly mentioned papers are briefly presented here:

1.3.1 Machinery Fault Diagnosis Using Supervised Manifold Learning

In [5], a new variant of the manifold learning method Laplacian Eigenmaps is used on four different datasets. The results are then compared to those of PCA, LDA and normal Laplacian Eigenmaps, all of whom it outperforms. This paper is similar to this thesis in the nature of the data sets and methods used. The main differences are in the dimensionality of the data sets, the data sets in [5] have both much lower and much higher dimensionality than those in this thesis. The number of fault classes are also much lower than that of the Adapt dataset, but similar to that of the Scania truck engine.

1.3.2 Multiple Manifolds Analysis and its Applications to Fault Diagnosis

In [6], a method called multiple manifold analysis is applied to vibration signals from ball bearings, just like [5] it claims to outperform the standard methods. It is different from this thesis in that it applies its method to highly dynamic systems, and the subject of manifold learning is a time series instead of a data set belonging to a class.

Chapter 2

From Fault Diagnosis to Manifold Learning

This chapter describes what fault diagnosis, pattern recognition and manifold learning are, and explains the relations between them and how the relations are relevant to this thesis. The chapter is ended with an overview of how fault diagnosis is performed and evaluated in this thesis.

2.1 Fault Diagnosis

To define fault diagnosis, we must first define its two components:

Definition 2.1 *Fault Detection: To determine if faults are present in the system and usually also the time when the fault occured. [9]*

Definition 2.2 *Fault Isolation: Determination of the location of the fault, i.e. which component or components that have failed. [9]*

We can then give a definition of fault diagnosis (which is somewhat different from the one given in [9]):

Definition 2.3 *Fault Diagnosis: To do fault detection and fault isolation.*

The goal is thus to determine if a fault has occured and if so - what fault has occured, given a set of sensor data. To improve readability, “the current fault or lack thereof” will be refered to as the fault-case.

Fault diagnosis is often done with knowledge both of the diagnosed machine and of sensor data from different faults-cases [9]. In this thesis, fault diagnosis is done with only knowledge of sensor data.

2.2 Pattern Classification

In [2], Duda and Hart defines pattern classification as “the act of taking in raw data and taking an action based on the category of the pattern”. The raw data in this thesis is the sensor data and its associated fault-classes. The action is the fault diagnosis of data samples without an associated fault-case. Since a change of viewpoint from fault diagnosis to pattern classification is done, a change of terminology is also needed, this change is given in Section 2.2.1.

2.2.1 Terminology in Classification and Diagnosis

To avoid ambiguity, classification terminology will be used from this point on if not otherwise specified. A mapping from diagnosis terminology to classification terminology is given below:

Table 2.1. Classification and diagnosis terminology mapping

Diagnosis	Classification
Sensor data	Samples
Sensor reading	Sample
Fault-case	Class
Fault Diagnosis	Classification
Sensor reading associated with a fault class	Labeled Sample

2.3 Manifold Learning

It is quite obvious that labeled samples from a system can be useful when doing classification on unlabeled samples if the system providing the samples is behaving sufficiently deterministic. Two much less obvious facts are 1. That the samples can explain parts of the underlying system and 2. That the unlabeled samples can be useful as well! These facts must be explained in terms of manifolds, which are described in Section 2.3.1

2.3.1 What Are Manifolds?

There are many different types of manifolds, each with different requirements and definitions. A technical definition of the most general type is given in [11]. Such generality is not needed in this thesis however. A sufficiently descriptive but more intuitive and concrete description that is consonant with [11] is as follows: A manifold is a smooth shape residing in a high dimensional space where any point can be uniquely described by a set of parameters smaller than the set of dimension it resides in. Manifold smoothness means that around every point on the manifold there is a neighborhood that on a small enough scale is similar to the unit ball of the dimensionality of the manifold.

The smoothness property can be illustrated with the manifold that is the surface of the earth. The surface of the earth can be completely parameterized by latitude and longitude but resides in three spatial dimensions and seems flat (as a unit ball in $\mathbb{R}^2$) on a human sized scale.

Manifolds and systems are related as follows: Assume a number of vectors, each consisting of q different real scalar values are recorded from q sensors in a deterministic system with p degrees of freedom where q > p, the vectors will reside in $\mathbb{R}^q$, but also in M where $M \subsetneq \mathbb{R}^q$ is a manifold of dimensionality p. The meaning of degrees of freedom is different for different systems, but generally means the dimension of “input” to the system. The input could be the input-voltage to a electrical two-port network or the throttle and gearing applied to an engine. In any case, the degrees of freedom can be seen as parameters to the manifold and to the system.

If the system above is behaving smoothly (generating a manifold) and to the degree in which the distribution of the vectors is uniform, approximations of different quality can be done of the manifold related to the system [1]. This approximation is an example of fact 1 in Section 2.3.

2.3.2 Manifold Learning and Classification

The fact that unlabeled samples can be useful for classifying other unlabeled samples is explained as follows: If the samples (the labeled and unlabeled ones) reside on a manifold that can be parameterized by relatively few dimensions, there might exist a transformation from the high dimensional space to the low dimensional space. Even if an unlabeled sample can not tell which part of the manifold the location of the sample belongs to, it does tell that the location does belong to the manifold as a whole.

Observe Figure 2.3.2, where the dots represent unlabeled samples, the questionmark a sample to be classified and the x’es and +’es represent labeled samples of two different classes. Given only the labeled samples there is no way of telling which class the unlabeled sample belongs to, even if the labeled samples do reside on a manifold. When a set of unlabeled samples is added, as is done in the right part of Figure 2.3.2, the shape of the manifold on which the data resides is revealed. Knowing the shape of the manifold makes it possible to identify the questionmark as a sample belonging to the class +. If the samples in Figure 2.3.2 where transformed to a one parameter data set where the parameter is the distance within the manifold from some point, the samples would be in distinct clusters.

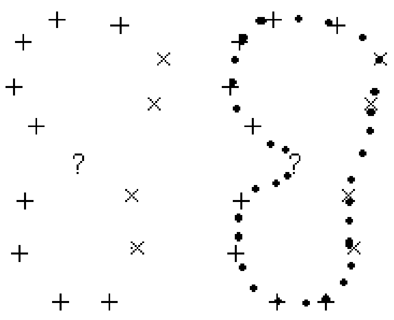


Figure 2.1. A manifold with unlabeled and labeled samples, illustrating the impact of unlabeled samples on manifold learning.

The unlabeled samples can thus help finding the manifold and the transformation, finding such a transformation has three possible benefits:

1. It is computationally cheaper to do classification on samples in low dimensional spaces.
2. Noise caused by inaccurate sensor readings that is not contained within the manifold is removed.
3. The distances within the manifold, commonly called the geodesic distances, should be more meaningful in explaining system behavior since these are related to the directions of freedom in the system.

In the data sets in this thesis, the numbers of samples are limited, the approximations of the manifolds and tranformations to them are thus limited too. Due to these limitations, the previously listed benefits may or may not be present.

2.4 Overview of Fault Diagnosis

This section seeks to provide the reader with an overview of the fault diagnosis evaluation process performed for this thesis. The process consists of four steps performed in a linear fashion with a few choices at some of the steps. The steps can be seen in Figure 2.4 and is described in some detail below:

1. Data is loaded from one of the systems and is then transformed into a normalized standard form. The data is separated into four sets for evaluation purposes. For more details, see Chapter 3.
2. The data resulting from the previous step is preprocessed by one of four different preprocessing methods, only the method LDA takes training fault data into consideration. This is described in Chapter 4.

3. The processed sample data and the training fault data is used to classify the evaluation data samples with one of two methods. This is decribed in Chapter 5.

4. The predicted evaluation fault data is compared with the real evaluation fault data resulting confusion matrix descibed in Chapter 6.

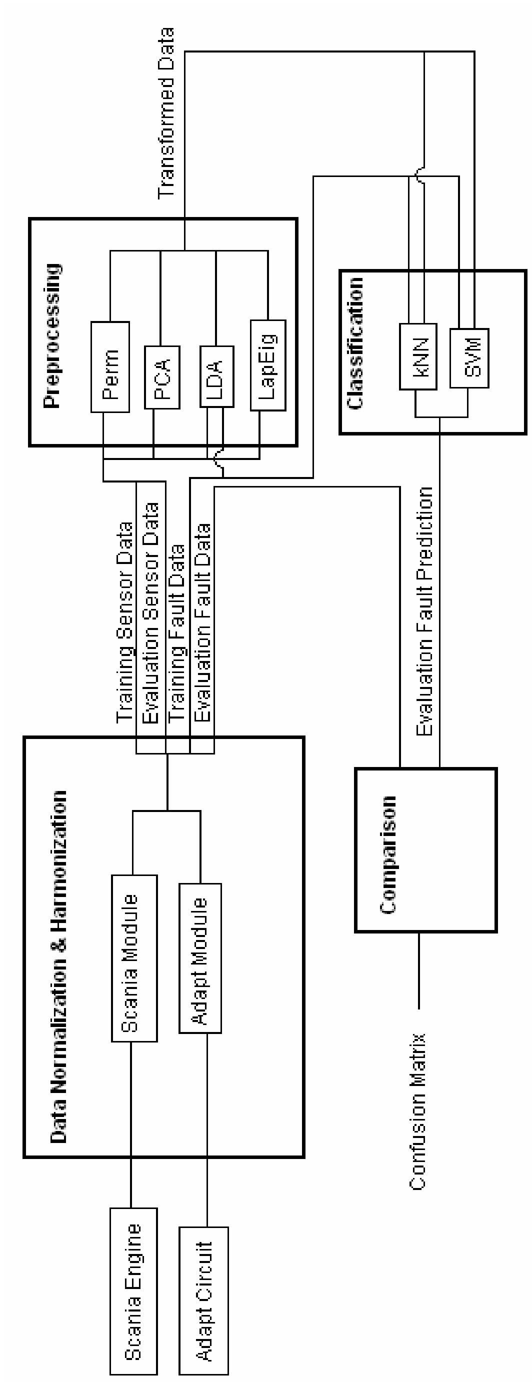


Figure 2.2. Overview of how fault diagnosis is performed and evaluated. 4 steps are performed: 1. Data preparation, where data from both sources is transformed into a standard form. 2. Data preprocessing, where one one of the methods to be compared is used. 3. Classification, where the classified data is used to classify the unclassified data. 4. Comparison, where the predicted classes are compared to the true classes.

Chapter 3

The Data Sets

Two sets of data will be used to evaluate the methods in this thesis. The sets are transformed so that both consists of a set of samples each consisting of sensor data and a fault class. The data sets are divided into training sets and evaluation sets, the former consisting of 80% of the samples. This technique is commonly refered to as cross-validation and makes prediction less reliant on noise in the training data [12]. The two sets of data are chosen to represent two different system types; an electrical system and a mechanical system.

3.1 The Scania Truck Engine

The Scania data set is the result a residual generation process performed for a Scania truck engine in [3]. 29 residuals are generated from a diesel engine model, these residual values are then recorded from real engine operation. The data used consist of 816 recordings of the residuals, each recording is associated with one or two faults, listed in Table 3.1

Table 3.1. Scania engine faults

Fault	**Fault Description**
1	exhaust gas pressure
2	intake pressure
3	intake air pressure
4	EGR vault position
5	mass flow

There are 152 samples associated with each single fault and 56 samples associated with fault 1 and fault 2. [10] The methods later used to do fault isolation through classification demand that each sample belong to one class. The two-fault samples are therefore associated with a new class in order to evaluate wether the methods find this two-fault case as distinguished from finding only either of the faults.

3.2 The Adapt

The Adapt is short for The Advanced Diagnostics and Prognostics Testbed and is a testbed developed by NASA for benchmarking of diagnostics and prognostics methods.

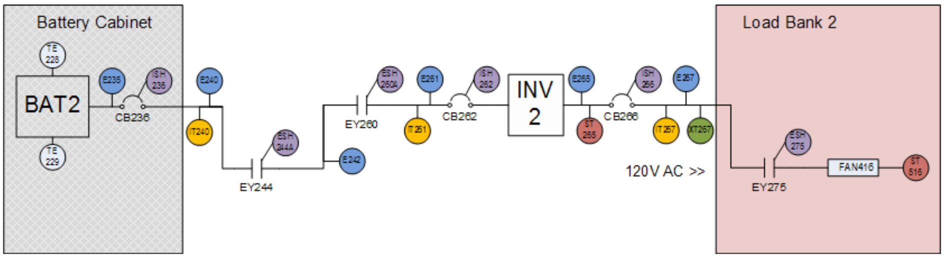
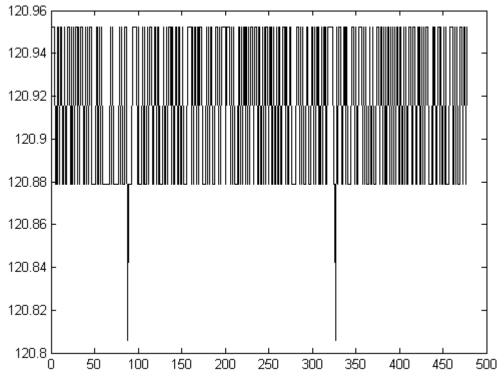


Figure 3.1. Adapt Lite circuit

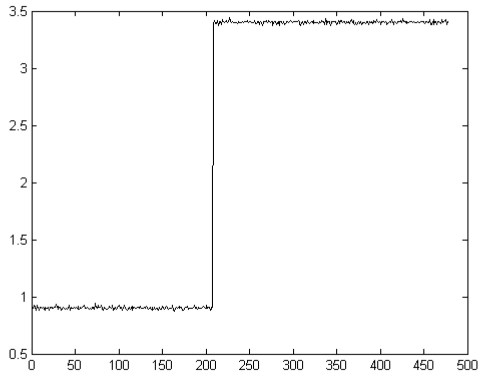
A relatively small data set called Adapt lite (Figure 3.2) which includes only a fraction of the eletrical components and possible faults included in the full version is used in this thesis [13, 14]. The sensor data available consists of 59 time series each of size approximately 478 and each representing 20 different sensors, three of these time series can be seen in Figure 3.2. There are thus 59×478 samples, each of dimensionality 20.

The time series are downsampled to 15 samples per time series in an effort to reduce noise and lower computation times. The 15 samples are then associated with a fault if one has occured. The 15 samples are treated as individual samples from this point on, i.e. their origin in the time series is ignored.

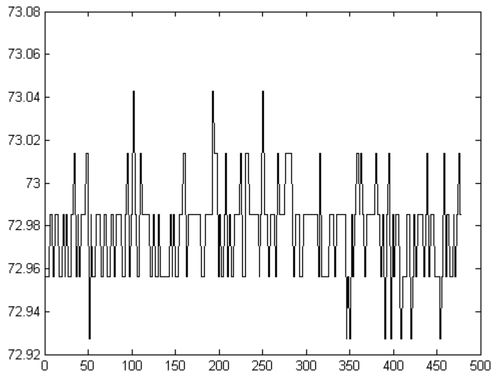
Any time series may have one out of 52 possible faults associated with it that occurs at a given time. Only 27 of the 52 possible faults are represented in the data however. Black box diagnosis can not be done on a sample whose corresponding fault is not represented in the data set, so only these 27 are considered.



(a) AC voltage sensor E267



(b) AC Current Transmitter IT267



(c) Temperature Sensor TE229

Figure 3.2. Three time series from Adapt Lite, the second one contains a fault.

Chapter 4

Preprocessing Methods

In order to find the transformations discussed in Chapter 2, three different methods were used, two of them tries to find linear manifolds i.e. subspaces, the third is capable of finding nonlinear transformations. To provide a baseline for comparison, a "Perm" preprocessing method is also used.

4.1 Perm

The Perm (permutation) preprocessing method is a baseline method for comparison with the other preprocessing methods. Unlike the other preprocessing methods, Perm does not transform the data in any way, it merely changes the order of the dimensions of the samples.

The order is changed because preprocessing methods are compared for different numbers of dimensions, i.e. only a subset of the untransformed data might be used for comparison. Some permutation must thus be chosen for every number of dimensions, and it turns out that different permutations result in different performance at fault diagnosis. The chosen solution is to select the best possible permutations that can be found in reasonable time.

If the data has n dimensions, n different permutations must be found, one for every number k of dimensions. For every number of dimensions, $\frac{n!}{(n-k)!}$ possible permutations exist, this results in a total of $\sum_{k=1}^{n} \frac{n!}{(n-k)!}$ permutations. For the data sets used in this thesis, n is 20 and 29 resulting in $6.6 \cdot 10^{18}$ and $2.4 \cdot 10^{31}$ permutations respectively.

Given the time needed to evaluate a permutation, it is unfeasible to do a complete search. A "greedy" search is instead performed, first finding the best dimension, then the one that is best together with the previously found one and so on. Best in this case means highest ratio of correct kNN-predictions. This reduces the search space size to $\sum_{k=1}^{n} k$.

4.2 Principal Component Analysis

Principal Component Analysis (PCA) is a method that finds the orthogonal directions within a space that contains most variation [4]. If the data is normalized so that its mean is 0, PCA can sometimes find “informative” directions depending on the source of the variation.

If information about the location of two clusters belonging to different classes is wanted, PCA may or may not be helpful as illustrated in Figure 4.2. In Figure 4.1(a) the variance direction and the “informative” direction coincide, in Figure 4.1(b) they do not. Figure 4.2(a) and 4.2(b) clearly illustrate the performance of PCA in these cases.

PCA can thus not be trusted to always find the directions separating classes. What PCA will always do however is to find directions with very low variance. This can be useful if the data reside on a linear manifold - i.e. a subspace. In this case the variation in directions perpendicular to the subspace will be zero or very small if there is noise.

A brief description of the theoretical foundation of PCA and how to use it is given in the following section.

4.2.1 Theory and Practice of PCA

Assume a m-dimensional zero-mean random vector **X** and a unit vector **q** of the same dimensionality on which to project X. The projection is then:

$$A = \mathbf{q}^{\mathbf{T}}\mathbf{X} \tag{4.1}$$

Since the mean of A is also 0, the variance of A is its mean-square value

$$E[A^2] = E[(\mathbf{q}^T\mathbf{X})(\mathbf{X}\mathbf{q}^T)] = \mathbf{q}^T\mathbf{R}\mathbf{q} \tag{4.2}$$

where R is the m-by-m correlation matrix of **X**.

It turns out that the projections vectors q_i containing most variance can be found by solving the eigenvalue problem:

$$\mathbf{RQ} = \mathbf{Q}\Lambda \tag{4.3}$$

Where $Q = [q_1, ..., q_m]$ and Λ is the matrix whose diagonal elements are the eigenvalues. The most important projection vector in terms of variance is the eigenvector related to the highest eigenvalue.

4.3 Linear Discriminant Analysis

In [7], Martinez and Kak gives this definition of Linear Discriminant Analysis (LDA):

“More formally, given a number of independent features relative to which the data is described, LDA creates a linear combination of these which yields the largest mean differences between the desired classes.”

LDA is thus similar to PCA, they both find a subspace of the ambient space that contain information that can be useful for classification. LDA can be seen as a supervised version of PCA in that it takes knowledge of classes into account. The possible benefit of using LDA is shown in Figure 4.3(a) where LDA manages to project the second data set so that classification is possible.

4.3.1 LDA in Practice

To find the optimal projection matrix W, two matrices called the within-class scatter matrix (S_w) and the between-class scatter matrix (S_b) are used. S_w and S_b are calculated as follows:

$$S_w = \sum_{j=1}^{c} \sum_{i=1}^{N_j} (\mathbf{x}_i^j - \mu_j)(\mathbf{x}_i^j - \mu_j)^T \tag{4.4}$$

$$S_b = \sum_{j=1}^{c} (\mu_j - \mu)(\mu_j - \mu)^T \tag{4.5}$$

Where $\mathbf{x}_i^j$ is the i:th sample of class j, μ_j is the mean of class j, c is the number of classes, N_j is the number of samples in class j and μ is the mean of all samples in all classes.

The column vectors of W are the eigenvectors of $S_w^{-1} S_b$

4.4 Laplacian Eigenmaps

Just like PCA and LDA, a Laplacian Eigenmap (LapEig) is a mapping from the original data set into a data set of (usually) lower dimensionality. Contrary to PCA and LDA, a LapEig is not a linear mapping, and contrary to LDA, it is not supervised, i.e. is does not take class information into account [1].

A LapEig is a mapping from the original samples to samples of lower dimensionality that tries to preserve local neighborhood information. To preserve local neighborhood information essentially means that if and only if two samples from the original data set are close, should they be close in the new data set too, eventhough the dimensionality is lower.

A justification for the ability of LapEigs to describe a manifold in fewer dimensions is quite theoretically intensive and thus beyond the scope of this thesis but can be found in [1]. Belkin hints at a justification as follows in [1]: "The justification for the algorithm comes from the role of the Laplace-Beltrami operator in providing an optimal embedding for the manifold. The manifold is approximated by the adjacency graph computed from the data points. The Laplace-Beltrami operator is approximated by the weighted Laplacian of the graph with weights chosen appropriately"

The LapEig algorithm in itself is simple but what it produces may not be trivial to understand. Several different but similar variants of LapEig are possible, two variations are presented in Section 4.4.1. The difference between these variations is in the construction of the adjacency graph matrix and are denoted A and B below.

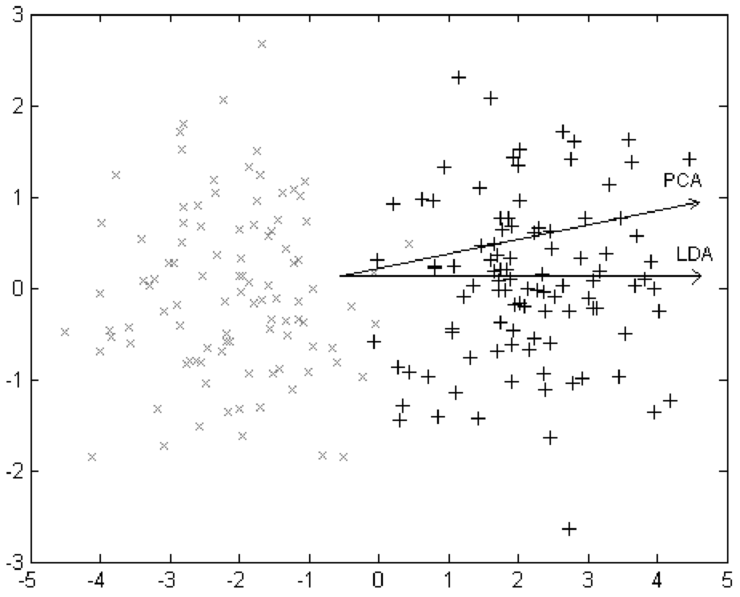
4.4.1 The Laplacian Eigenmaps Algorithm

1. Assume a set unlabeled samples x_i, both the labeled samples and the unlabeled ones are used in this thesis, but the label is irrelevant to LapEig
2. The adjacency graph matrix D for the data set is constructed as follows. $D(i,j) = 1$ if
   A. x_i is one of x_j's k closest neighbors or vice versa or
   B. if x_j is within the ε-neighborhood of x_i,
   otherwise $D(i,j) = 0$
3. A diagonal matrix W whose elements are the sums of the rows of D is calculated.
4. The Laplacian matrix L is calculated as $L = D - W$.
5. The eigenvalues and eigenvectors of L are calculated.

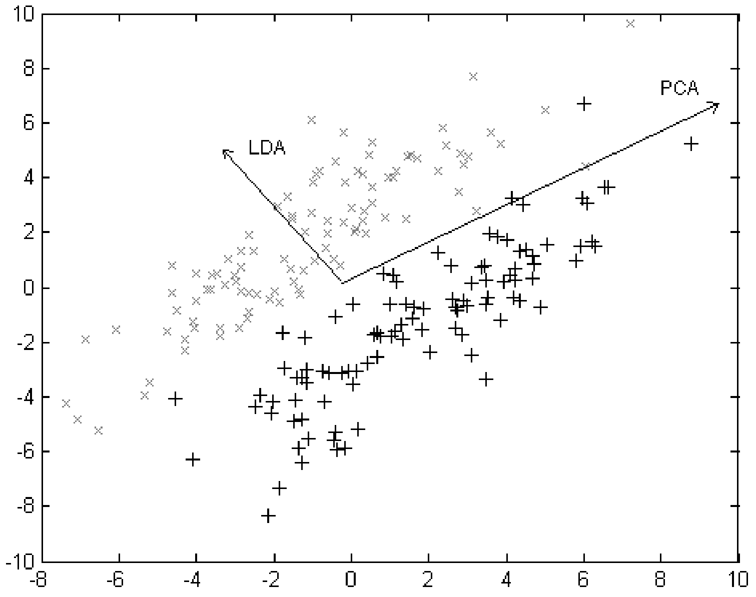
k and ε in step 1 are parameters that must be calibrated case by case.

The mapped data sample coordinates are then found in the eigenvectors, the eigenvector corresponding to the smallest nonzero eigenvalue contains the coordinates for the first dimension, the eigenvector corresponding to the second smallest nonzero eigenvector to the second dimension and so on.

The results of using PCA, LDA and LapEig variant A on a data set whose elements reside on a manifold consisting of two spiral surfaces called “the swiss roll” is shown in Figure 4.6.

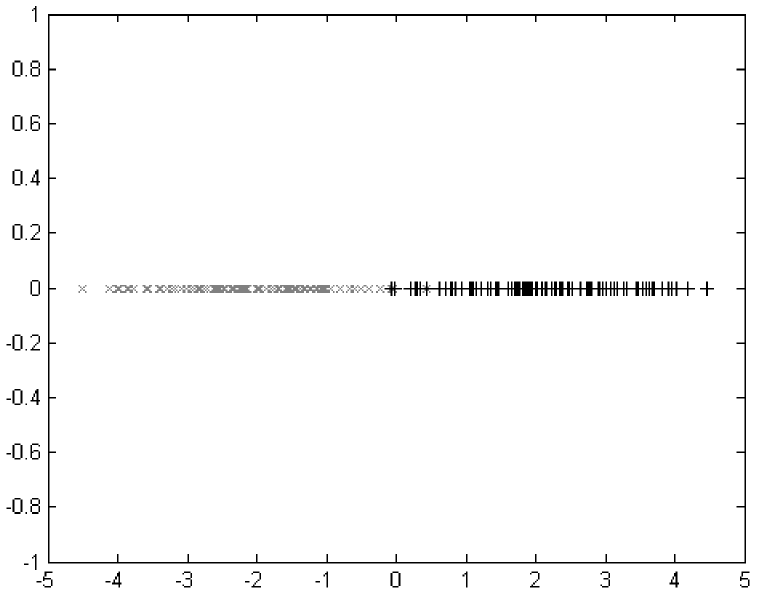


(a) Test data set 1 including first PCA -and LDA vectors, due to the data distribution, the vectors are similar

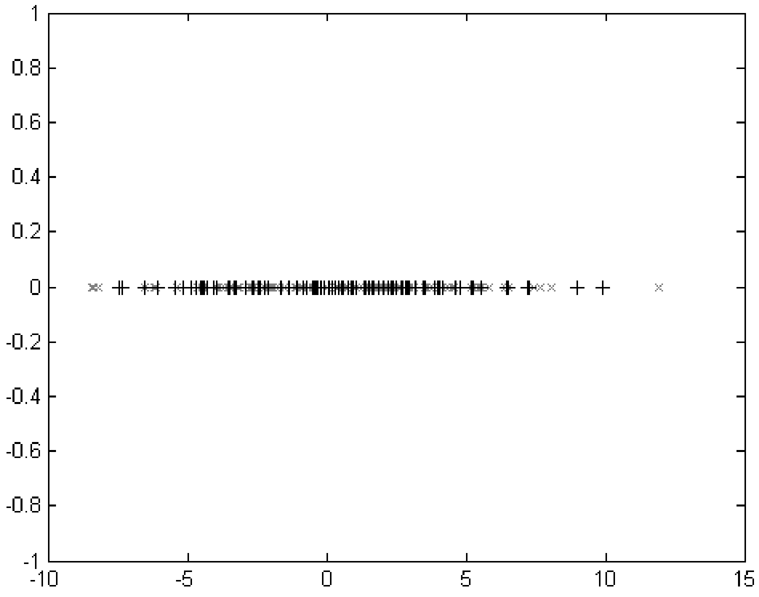


(b) Test data set 2 including first PCA -and LDA vectors, illustrating a case where the vectors are not similar

Figure 4.1. Two data sets used to illustrate PCA and LDA

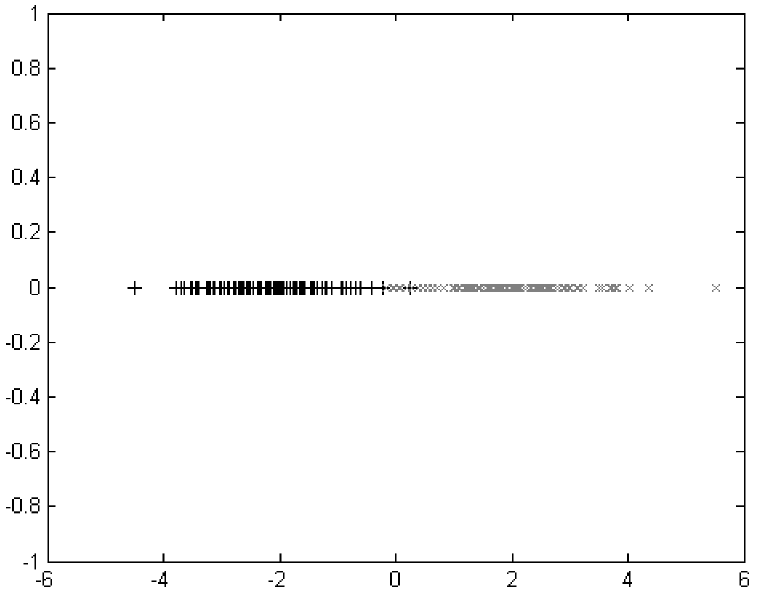


(a) Test data set 1, PCA projection, good separation between classes



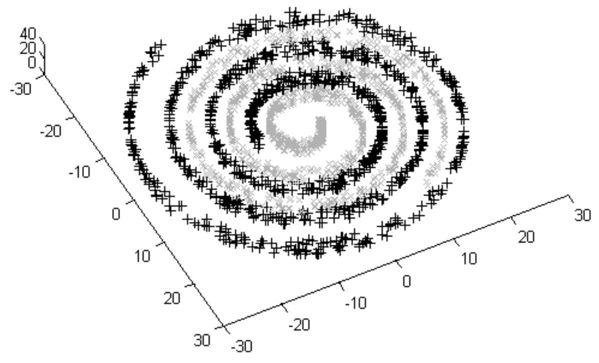
(b) Test data set 2, PCA projection, bad class separation

Figure 4.2. PCA projection of the data sets to 1 dimension

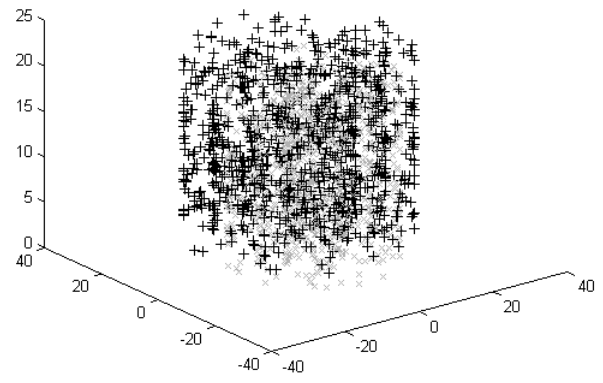


(a) Test data set 2, LDA projection, LDA manages to make a better projection than PCA in terms of separation of classes

Figure 4.3. LDA projection of data set 2 to 1 dimension

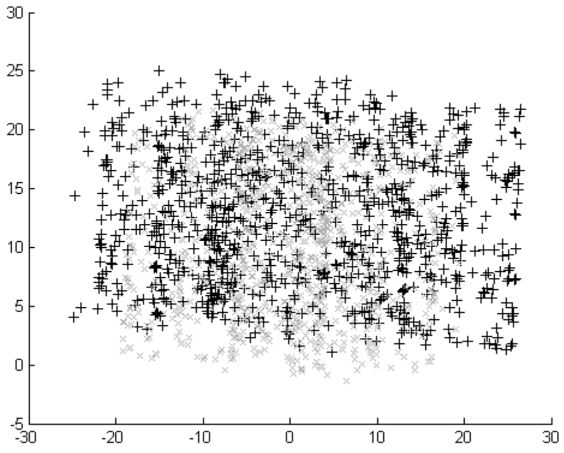


(a) Swiss roll perspective 1

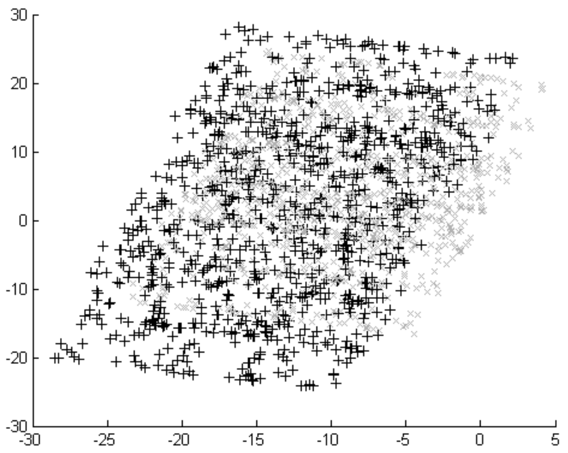


(b) Swiss roll perspective 2

Figure 4.4. The swiss roll, two perspectives. The swiss roll illustrates a 2 dimensional manifold in a 3 dimensional space

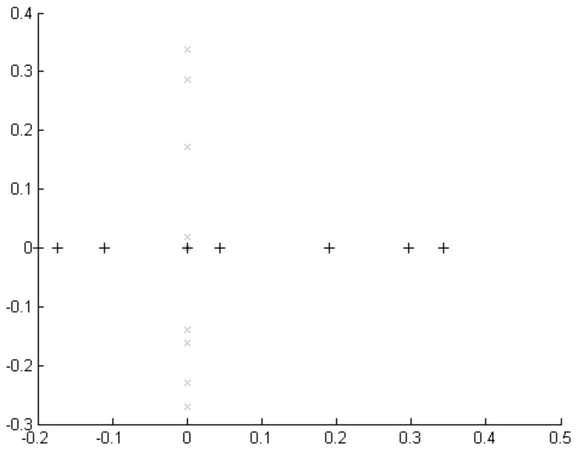


(a) Swiss roll PCA projection



(b) Swiss roll LDA projection

Figure 4.5. 2 dimensional projections of the swiss roll by PCA and LDA, neither projection results in good separation



(a) Swiss roll Laplacian Eigenmaps tranformation

Figure 4.6. 2 dimensional projection by LapEig, classes are well separated

Chapter 5

Classification Methods

As stated in Section 2.2, to do classification is to infer the class associated with a unlabled sample given a set of labeled ones. Two methods for classification are used in this thesis; SVM and kNN.

5.1 Support Vector Machines

Support Vector Machines (SVM) is a classification method that calculates an optimal plane for separation of two classes [4]. In its standard form SVM requires the classes to be linearly separable, i.e. they can be divided by a plane so that no sample is on the "wrong" side.

The standard form of SVM is described in the next section, a more general version capable of handling non-separable classes is given in Section 5.1.2. Section 5.1.3 describes how to use two-class SVM to solve classification problems with three or more classes.

5.1.1 The Linearly Separable Case

The SVM algorithm finds a plane $\mathbf{w}^T\mathbf{x} + b = k, k = 0$ so that the planes for k=1 and k=-1 are as far away from each other as possible while keeping every sample of each class on different sides of the planes [4]. The calculated planes for k=1 and k=-1 always intersect with at least one sample each since further expansion of the margin between the planes would otherwise be possible. These samples are called support vectors, giving the methods its name. The SVM algorithm thus finds the plane separating the two classes that is most distant to the closest sample, as illustrated in Figure 5.1(a).

5.1.2 The Non-Separable Case

The nature of the non-separable case implies a less trivial optimization [4]. In the linearly separable case, optimization was straight-forward - greater distance or less distance. The non-separable case demands a trade-off, the other goal being to minimize the "negative" distances of samples on the wrong side of the plane, ideally having them on the right side of the plane. The trade-off between the two optimization goals is in practice done with an algorithm parameter C. For small values, margin maximization is favoured, i.e. for C=0, we get the SVM algorithm as presented in the previous Section. For large values, minimization of misclassifications is favoured. The optimal value for C must be determined by calibration. A high value will give many correct classifications for the training set, but might give worse results for the set to be evaluated because of the inherent noise in the training data. A non-separable case with a plane corresponding to a high C value is illustrated in Figure 5.1(b)

5.1.3 The Multi-Class Case

SVM is binary in the sense that the sample to classify is always on one side or the other of the calculated plane. So for cases where a sample might belong to one of more than two classes, one has to use multiple planes. There is no one single best way to calculate these planes. Two common ways are as follows:

1. Calculate a plane for every class between that class and all the other classes, the sample to be classified is predicted to belong to the class corresponding to the plane whose normal vector to the sample is largest.

2. Calculate a plane for every pair of classes, each plane classifies the sample to belong to either of the two classes related to the plane. Each such classification is considered a vote, the class with most votes is the predicted class.

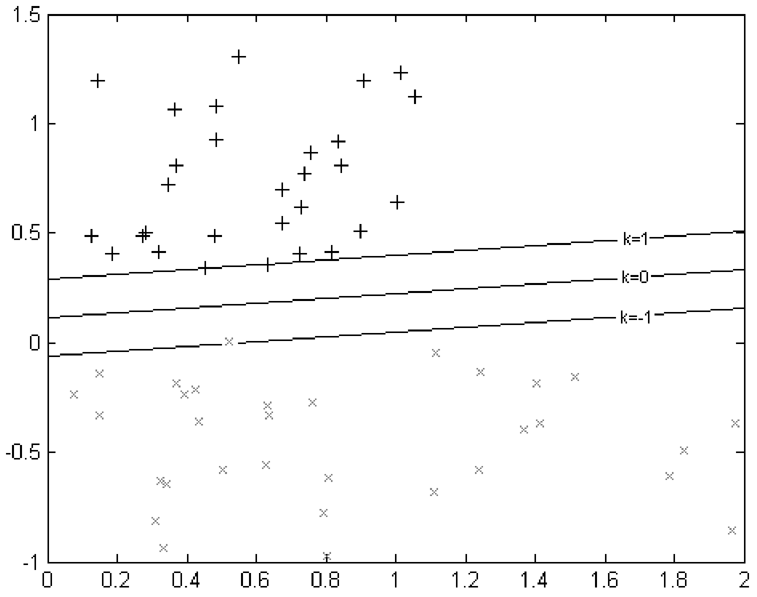
This thesis uses the second way.

5.2 k-Nearest Neighbor

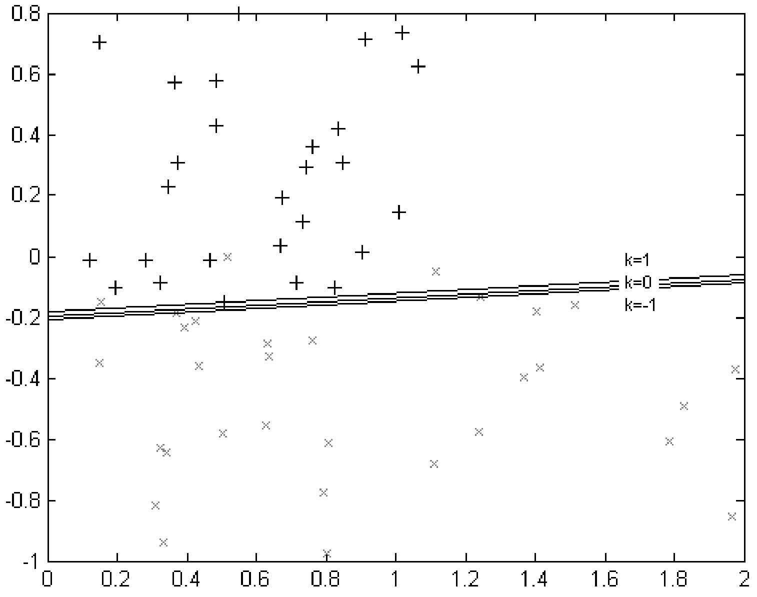
k-Nearest Neighbor is another common method for pattern classification [12]. Given a sample of unknown class, the training samples are ordered by their distance to the sample to be classified.

The sample is then predicted to belong to the most common class of its k neighbors. kNN is widely used because it's easy to use while still performing well. There are two disadvantages with kNN: First, classification is expensive both in terms of memory and time since all training samples must be stored and processed for every classification. Second, it is sensitive to differences in class size; a larger class will have an advantage over a smaller class. This advantage can be countered by various measures such as duplicating samples in the smaller class or modifying the way ones determines the predicted class. All such methods have their drawbacks however.

This advantage depends on k, for small values of k, the size differences does not have a big impact on classification but is on the other hand more sensitive to noise in the data. As k grows, kNN is less sensitive to noise but the number



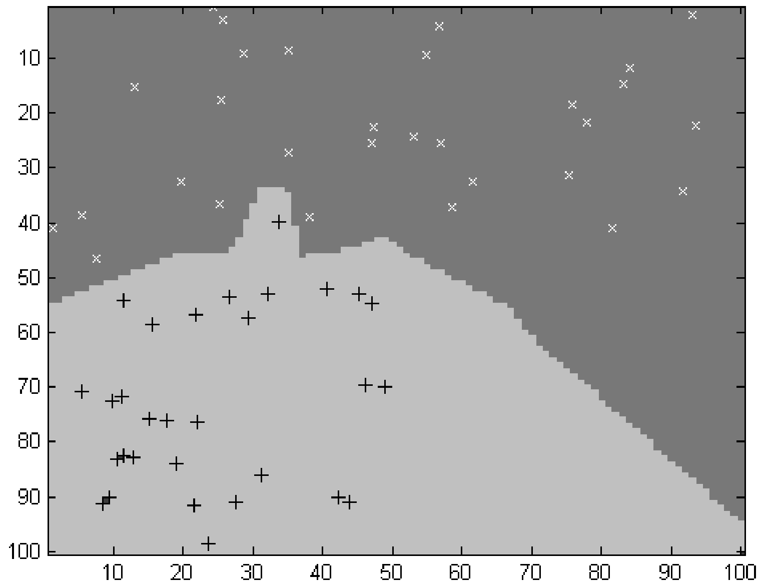
(a) SVM linearly separable case



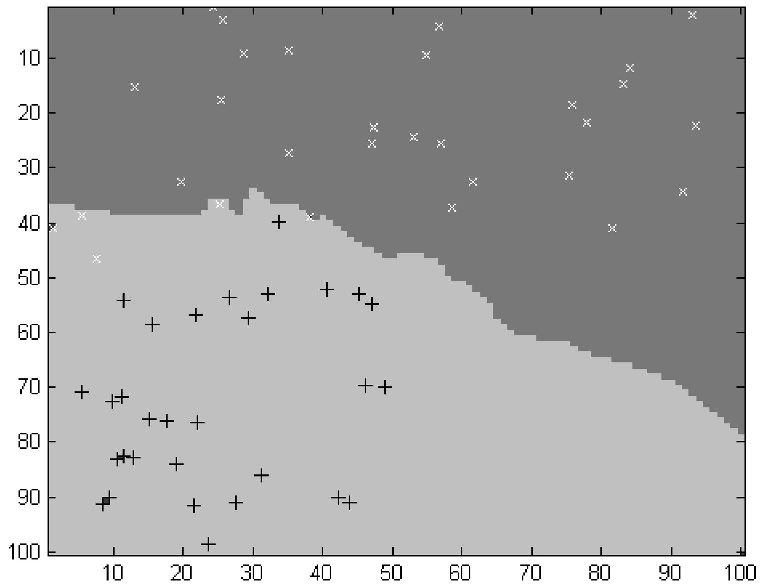
(b) SVM not linearly separable case

Figure 5.1. Two variants of Support Vector Machines

of samples classified as belonging to the smaller class steadily decreases. kNN for three different values of k is illustrated in Figure 5.2. Just like in the case with SVM and C, k for kNN must thus be chosen by calibration.



(a) kNN, n=1



(b) kNN, n=14

Figure 5.2. k Nearest Neighbor for different k's, the two colored areas indicating classification boundaries. A larger k results in a softer boundary.

Chapter 6

Measuring Performance

This chapter defines a set of standards of performance measures for evaluation of classification. A central concept used to evaluate every method in this thesis is the confusion matrix. The confusion matrix represents what the current method predicts and takes this in relation to what the actual class is.

The confusion matrix is useful since many other measures can be derived from it while it still maintains much of the information from the classification process.

Two different standards are used, in one valuation of different faults are not taken in consideration - all faults-cases are seen as equal. In the other, the ability to value of fault higher than another is considered.

6.1 The Goal is Fault Diagnosis

Even though fault diagnosis has now been "transformed" to classification, the purpose can only be defined in terms of fault diagnosis. The purpose of fault diagnosis is to identify faults when they appear [9], but this answer is not complete.

Depending on the specific machine to be diagnosed, different behaviours of a diagnosis system can be desired. A correct diagnosis is always better than an incorrect one if nothing else is known, but there may be situations that are not quite as simple.

A fault f_1 may be considered more "important" to detect than another fault f_2, meaning false diagnoses of f_2 can be accepted as long as f_1 is diagnosed correctly often enough. If diagnosis of a truck engine is not perfect, one can imagine that a dangerous fault is more important to detect than less dangerous faults or the no-fault case, if the goal is the survival of the driver. On the other hand - putting a too big emphasis on finding this dangerous fault will worsen other diagnosis for other faults and increase the number of false "faults", which could undermine the drivers trust in the diagnosis system. A technical description of measuring performance with certain performance requirements is given in Section 6.4.

6.2 The Confusion Matrix

The confusion matrix is constructed by associating every row with a predicted class and every column with an actual class. For every performed classification where predicted class is p and actualt class is a, one is then added to element in column a and row p. [8]

The result is a matrix which - as the name suggests - shows which classes are commonly confused. The confusion matrix as well as its derivates are illustrated with an example throughout this section.

For reasons descibed in Section 6.3, one confusion matrix is calculated for every number of dimensions, these are stacked into a confusion tensor.

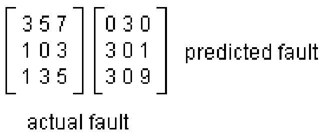


Figure 6.1. Confusion matrices for 1 and 2 dimensions

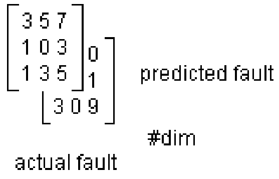


Figure 6.2. Tensor consisting of two confusion matrices

Every column is then divided element by element by the total number of samples of that column so that the column sums are 1. The resulting tensor M_n consists of columns representing relative distributions of predictions, making comparison between classes easier. M_n tells us that for one dimension, the second fault was never correctly identified and that for two dimensions, the third fault was correctly identified in 90 % of the cases. M_n will be refered to as a normalized confusion tensor.

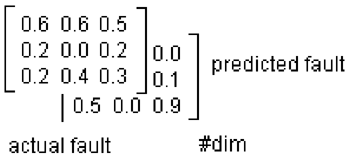


Figure 6.3. Normalized confusion tensor

6.3 Dimensionality

Another aspect that requires analysis is the dimensionality of the samples to be classified, fewer dimensions usually means less computations and thus shorter response time if a fault occurs. As stated in Chapter 2, lower dimensionality is also central to manifold learning. If a manifold is found, a smaller set of dimensions should be able to describe the manifold and the other dimensions could be discarded as noise.

For these reasons, a normalized confusion matrix is calculated for every number of dimensions. If put together, these form a 3-dimensional tensor whose indices are associated with predicted faults, actual faults and numbers of dimensions in the given order. This tensor can be reduced to a two dimensional matrix by selecting the elements where predicted fault = actual fault, i.e. the diagonal of the confusion tensor or the ratio of correct classifications.

Information about missclassifications is thus traded for visibility. The resulting matrix, henceforth refered to as correct classification matrix can then be used to evaluate how the method performs for different classes and for different dimensionalities. The correct classification matrix (M_{cc}) is then:

$$\begin{array}{cc} \begin{bmatrix} 0.6 & 0.0 \\ 0.0 & 0.0 \\ 0.3 & 0.9 \end{bmatrix} & \text{fault} \\ \text{\#dim} & \end{array}$$

Figure 6.4. Correct classifications matrix

This matrix tells us that idendification of the second fault failed for all numbers of dimensions and that the ratio of correct identifications for the third fault increased by 60 percentage points when using two dimensions instead of one.

The correct classification matrix describes classification performance for different numbers of dimensions and for diferent faults. If prediction time is a factor, it could also be used to find the optimal trade-off between time and correct predictions.

6.4 Performance on Valued Faults

The impact of externally given requirements on certain identification results will be measured using three scenarios. The scenarios are in the following form: A set of faults must be correctly identified with kNN at a certain ratio using a number of dimensions, achieve as good general identification as possible while upholding the requirements.

To achieve a high enough ratio of correct identifications for the given faults, kNN is biased towards that fault using false neighbors. A false neighbor is a ever present neighbor, regardless of what sample is to be classified. I.e. if fault 2 must be identified say at 75 % of the time, false fault-2-neighbors are added until the requirement is upheld. Since only one number of dimensions is used, M_n can be presented directly (this would be difficult for the general case where M_n is 3-dimensional), thus M_{cc} is not calculated.

Chapter 7

Results

This chapter presents the results of executions done to measure performance of preprocessing methods applied to Scania and Adapt fault diagnosis. Issues such as time consumption and parameter calibration is covered in Sections 7.1 - 7.2. The results of the executions with and without requirements on performance is presented in Section 7.3. Two abbreviations are widely used through this chapter: cc - ratio of correct classifications and #dim - the number of dimensions.

7.1 Experimental Setup

This section describes the practical aspects of how performance measuring is done in this thesis and the steps are needed to measure this performance.

Before any performance measuring can be done for the preprocessing and classification methods, a few of parameters must be set in the algorithms. The search space in which to find this optimal combination of parameters is multidimensional, and there is a unique search space to explore for every combination of methods. Another factor to consider when searching is the time consumption for evaluating a single state in this search space. Execution times are therefore measured for various method executions with parameter values and data that is likely to be used. As with all calculations in this thesis, these calculations are run in Matlab on a modern desktop computer with a dual core 1.73 GHz processor.

As can be seen in Table 7.1, the time consumptions for the classification methods are between 1 and 15 seconds, and since either kNN or SVM must be used in every evaluation, each evaluation will take some time. It turns out that this combination of large search spaces and large time consumptions makes an exhaustive search impossible.

Another effect of the time consumption is that SVM execution on the Adapt data is not run, this is due to the high number of classes in Adapt. Running SVM once on Adapt data takes three to five minutes, a complete performance evaluation would thus take up to a week.

Table 7.1. Typical execution times

Method Name	Time
Perm	0.0006-0.0009 s
PCA	0.001-0.005 s
LDA	0.01 - 0.08 s
LapEig	2.5 - 5 s
kNN	1.0 - 1.2 s
SVM	0.9 - 15 s

7.2 Parameters

To be able to do performance evaluation, there are a few parameters to set. Some methods have no parameters, some do. The parameters in the different methods are summarized in Table 7.2.

Table 7.2. Parameters, methods and their meaning

Parameter	Method	Description
List	Perm	List is not exactly a conventional parameter but provides external information that had to be calculated. The list is a best to worst list of unprocessed dimensions.
n	LapEig	The number of neighors to consider when constructing the adjacency matrix.
k	kNN	The number of neighbors to consider when classifying with kNN.
C	SVM	Tradeoff parameter between a big margin and a high number of correct classifications.

To determine optimal values of these parameters is not trivial. Any given execution have many degrees of freedom: The data used, the preprocessing used, the classification method used as well as the definition of what a “good result” is. Since an optimal parameter value for one type of execution might be suboptimal for other execution types, an exhaustive search for best parameters is unfeasible.

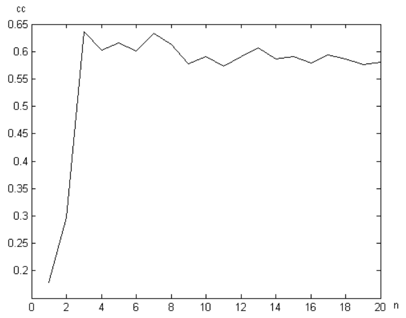
Parameter values are thus calculated on a subset of the set of possible executions, and the same parameter values are used in similar executions. Since the main goal with this thesis is to compare the preprocessing methods, non-optimal parameters for the classification methods should be less hurtful since these will penalize all preprocessing methods in the same way.

Classification parameters and preprocessing parameters are thus calibrated in different ways.

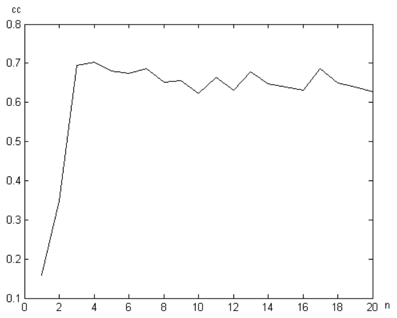
7.2.1 Preprocessing Parameters

Only two of the preprocessing methods, LapEig and Perm have parameters. PCA and LDA only depend on the labeled data. The parameters to set for LapEig and Perm are the number of neighbors parameter (n) and the best-to-worst-lists. The optimal values for these parameters depends on three factors; the data set, the classification method and the number of dimensions. On the other hand, it seems likely that e.g. a too large n should lead to universally bad preprocessing and thus classification, and that there should be an n good for both SVM and kNN and possibly for both data sets.

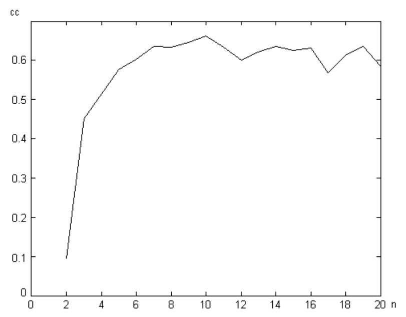
The parameter n is calibrated by evaluating LapEig with both Scania and Adapt data, using 3 and all (29 or 20) dimensions and for both SVM and kNN. As can be seen in Figure 7.1, n=3 and n=7 seems to be good choices for kNN for Scania and Adapt data respectively. This choice is based solely on general performance shown in this figure. This figure shows the performance for low values of n to illustrate the locally best choice, the performance for higher values of n is considerably worse, but as can be seen in the figures, the sensitivity to n is low for the ranges shown.



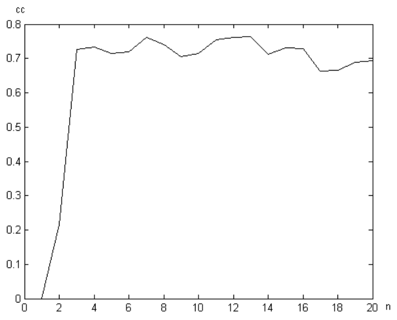
(a) LapEig kNN #dim=3 Scania



(b) LapEig kNN #dim=29 Scania



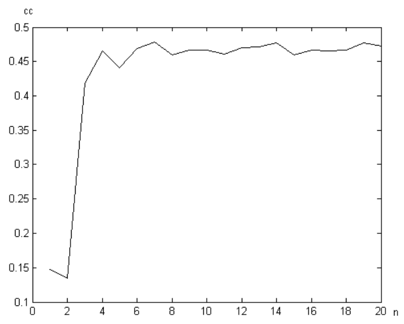
(c) LapEig kNN #dim=3 Adapt



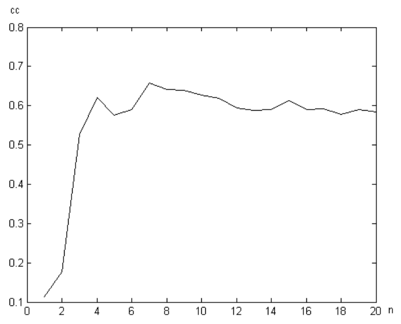
(d) LapEig kNN #dim=20 Adapt

Figure 7.1. LapEig kNN calibration, seeking an n with high cc.

n is calibrated the same way for SVM but the search space is only half of that for kNN since Adapt data is not considered. Figure 7.2 shows that n=7 seems to be a good choice for SVM. The situation in Figure 7.2 is similar to that in Figure 7.1, it shows the values for low values of n since these are the best performing ones.



(a) LapEig SVM #dim=3 Scania



(b) LapEig SVM #dim=29 Scania

Figure 7.2. LapEig SVM calibration, seeking an with high cc.

As was described in Section 4.1, an ordering of unprocessed dimensions is found for each data set so that every added dimension contributes the most to classification performance with kNN. The first dimension is thus the one that performs best on its own, the second is the one which performs best together with the first and so on.

The results of the different preprocessing methods is shown in Figures 7.3 to 7.7 and Figures 7.8 to 7.11. The Scania plot uses different markers and colors for ever fault while the Adapt plot shows only faults or fault-free samples. The two dimensions shown are the two highest ranked ones. These figures serve to illustrate the ability of various preprocessing methods to separate different classes into clusters. As can be seen, the resulting patterns are quite different but the class overlap is similar, i.e. some classes seem impossible to separate. As can be seen, both PCA, LDA and LapEig seem to be able to do some class separation, although in different ways and quality.

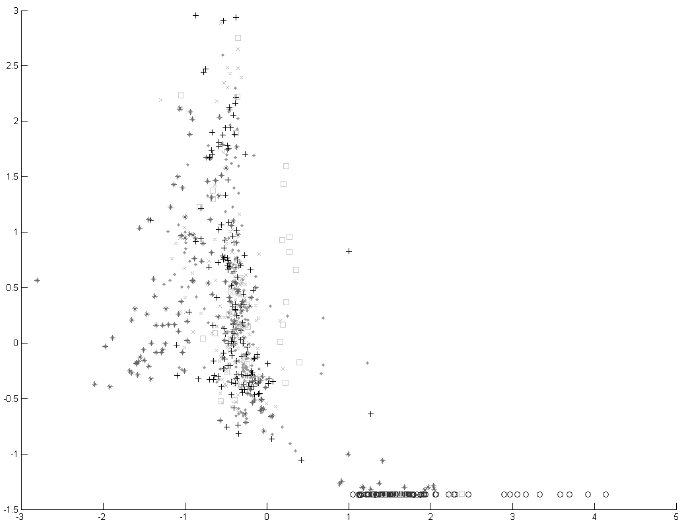


Figure 7.3. Two best dimensions from Scania data processed by Perm, there is some class separation even in the unprocessed data.

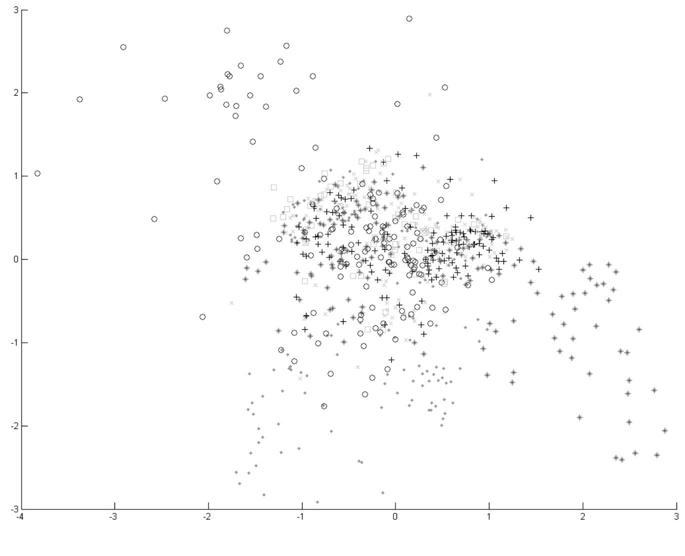


Figure 7.4. Two best dimensions from Scania data processed by PCA, three distinct clusters can be seen in the top left, bottom and bottom right.

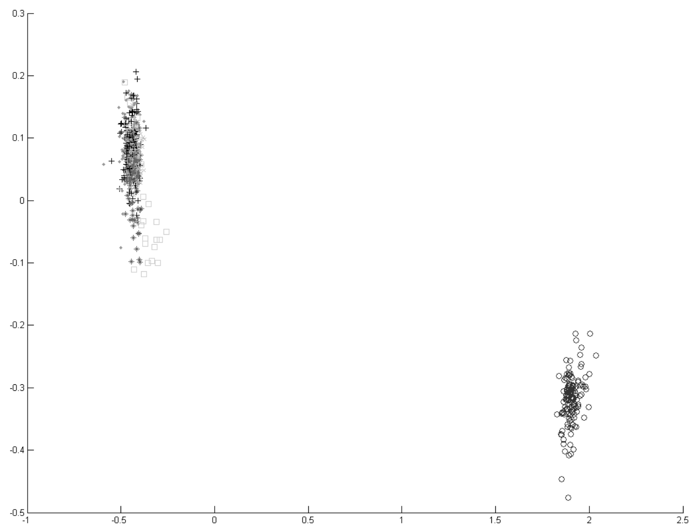


Figure 7.5. Two best dimensions from Scania data processed by LDA, LDA has projected one class into a completely separate cluster.

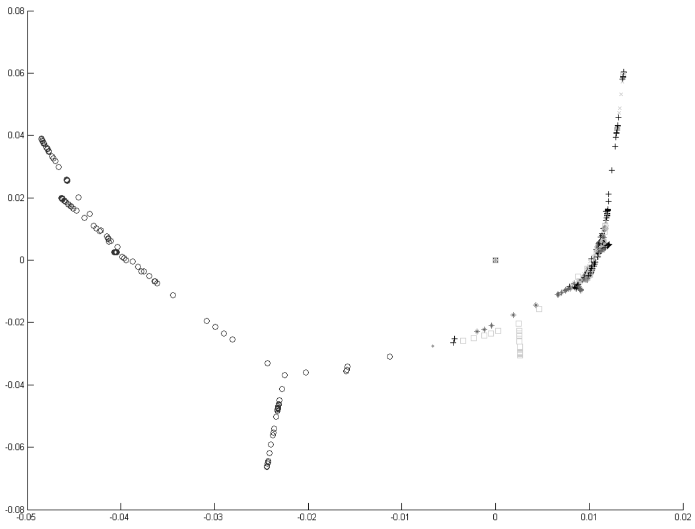


Figure 7.6. Two best dimensions from Scania data processed by LapEig with n=3, multiple distinct clusters not mixed with other classes.

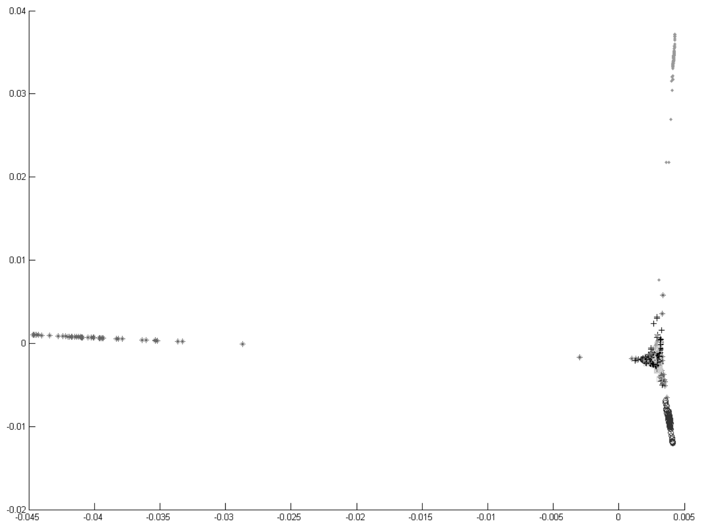


Figure 7.7. Two best dimensions from Scania data processed by LapEig with n=7, three one-class clusters and one mixed central cluster.

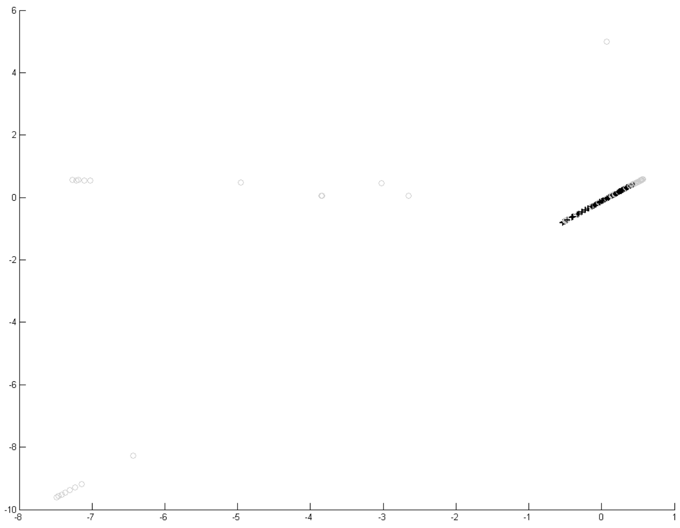


Figure 7.8. Two best dimensions from Adapt processed by Perm, the Adapt plots are not as informative as the Scania plot, but some different preprocessing behavior can be observed.

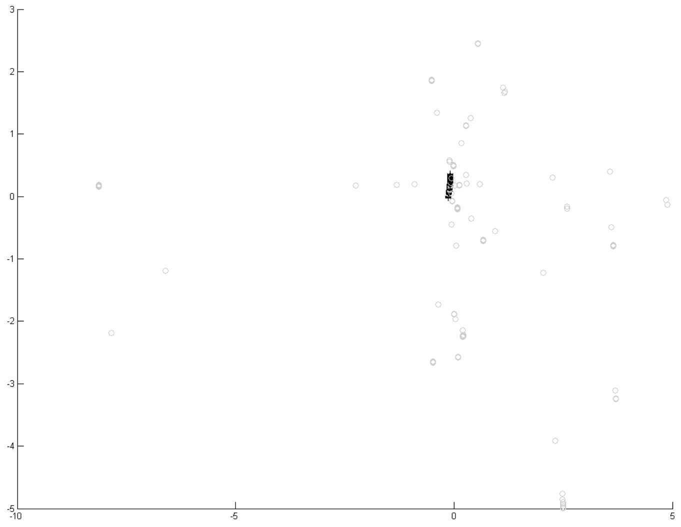


Figure 7.9. Two best dimensions from Adapt processed by PCA

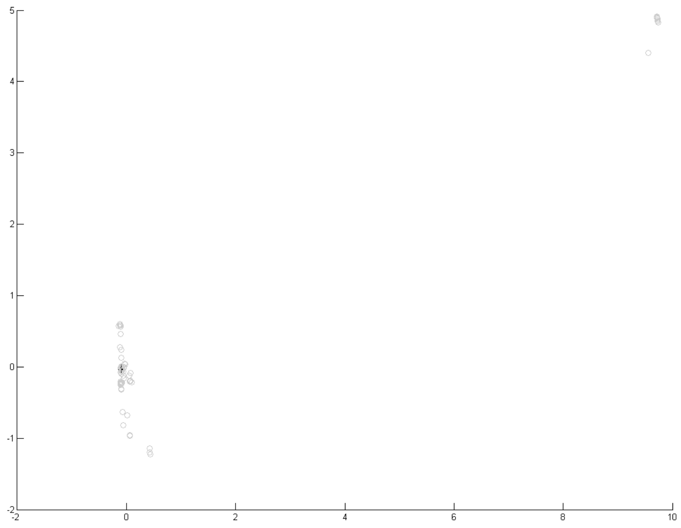


Figure 7.10. Two best dimensions from Adapt processed by LDA

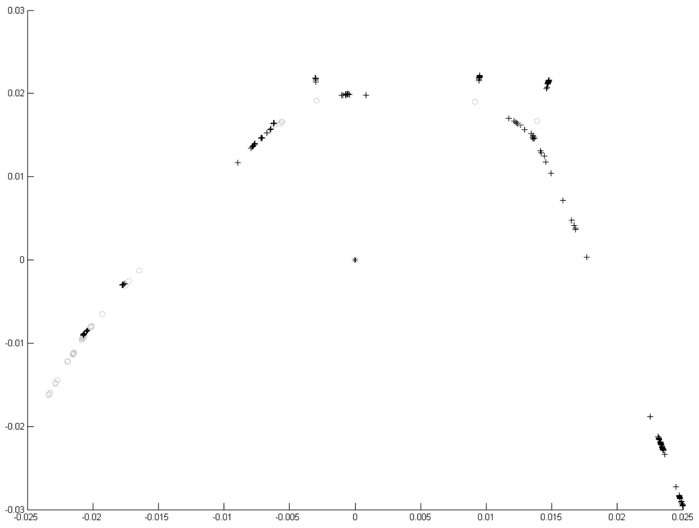


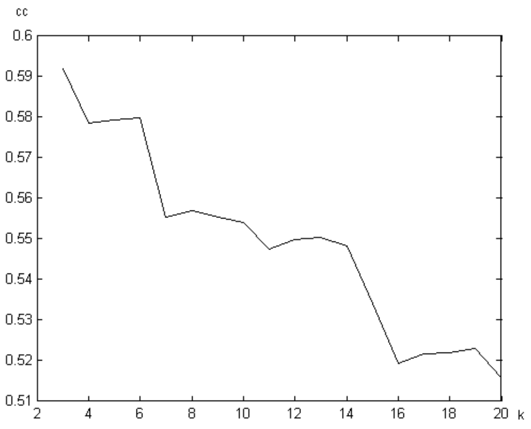
Figure 7.11. Two best dimensions from Adapt processed by LapEig with n=7

7.2.2 Classification Parameters

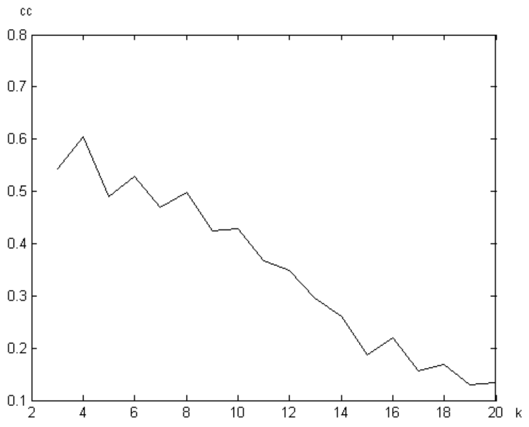
The parameters k and C in the classification methods kNN and SVM are both related to over-fitting, i.e. to reliance on noise (as opposed to reliance on signal.) A given finite set of data from a larger distribution will generally be noisy i.e. not perfectly representing the larger distribution. In the case of kNN, the law of large number suggests that high values of k will cancel out the effects of this noise. On the other hand, a too large value of k will confuse the different classes with each other as was shown in Figure 5.2. A similar noise-based argument can be made about C and SVM.

In both these cases, the best way to calibrate the parameter without getting exposed to noise is to measure the performance for different parameter values. kNN is run with Perm preprocessing with 3 dimensions on both the Adapt data set and on the Scania data set with k ranging from 3 to 20. The average number of correct classifications for the two data sets, the Perm preprocessor and different values of k is shown in Figure 7.12, as can be seen in this figure, kNN is not very sensitive to the noise in the data sets in this case since the optimal value is very low. To avoid ambigious classifications that would be more likely using a even value of k, the odd value k=3 is chosen.

A similar set of executions is done for SVM for both datasets. It turns out that a linear search for C is less suitable than an exponential search, so values from 10^{-5} to 10^{7} are used, taking steps of powers of 10, the results are seen in Figure 7.13 where the performance is growing for higher values of C until it reaches



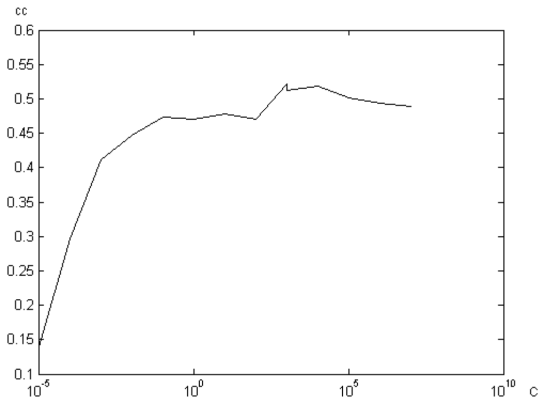
(a) kNN, Scania



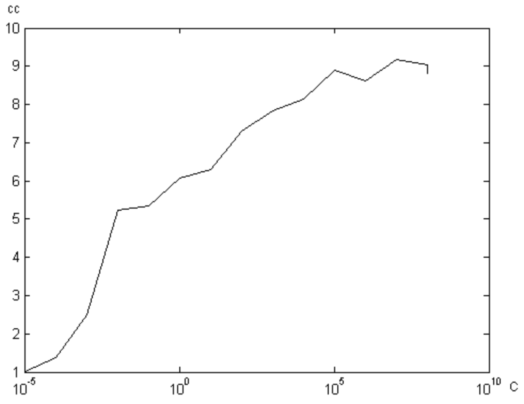
(b) kNN, Adapt

Figure 7.12. k Nearest Neighbor calibration, seeking an odd k with high cc.

a few powers of 10, the chosen value for C is 10^4.



(a) SVM, Scania



(b) SVM, Adapt

Figure 7.13. Support Vector Machines calibration, seeking a C with high cc.

7.3 Performance Evaluation Setup

Four main combinations of executions are done: Scania-kNN, Scania-SVM, Adapt-kNN, and Scania-kNN for differently valued faults, the first three of these combinations consist of a set of execution for ever preprocessor. Every execution is done 25 times for every number of dimensions, executing 25 times is necessary since each execution has a unique randomization of training/evaluation data. A matrix M is calculated for every dimension (i.e. is the results of 25 summations.) From these matrices the correct classifications matrix M_{cc} is calculated.

The fourth group is somewhat different in that it consists of three different scenarios, these scenarios and their results are presented in Section 7.5.

7.4 Performance Without Requirements

These are the results of the executions without requirements on performance. The results are presented as surface diagrams of correct classifications matrices where the height is the ratio of correct classifications, the depth is the number of dimensions and width is the fault. Every combination is described by a surface diagram for the baseline (Perm) executions as well as difference surface diagrams for the performance increases of different preprocessing methods.

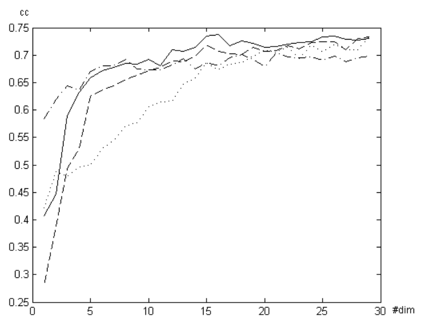
A simple graph of averages of correct classifications for different preprocessing methods and numbers of dimensions is provided to give an overview of the group.

The results of each group is presented in Figures 7.14, 7.15 and 7.16.

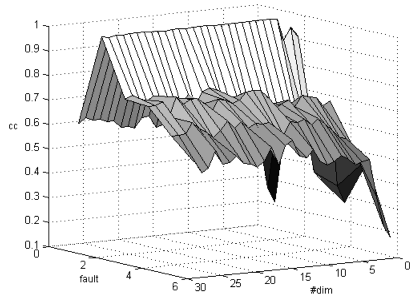
Generally speaking the results seem to depend on both the data set and the preprocessing method. In the Scania-kNN-case, LapEig performs very well for few dimensions but is surpassed by all other methods which converge at a common general performance level as the number of dimensions grow towards 29.

In the Scania-SVM-case LDA and LapEig is very slightly superior to PCA and Perm for 2-3 dimension but as in the Scania-kNN-case the performance converges towards the same level eventually. As opposed to the Scania-kNN-case, LapEig performs as well as the other preprocessing methods for 29 dimensions.

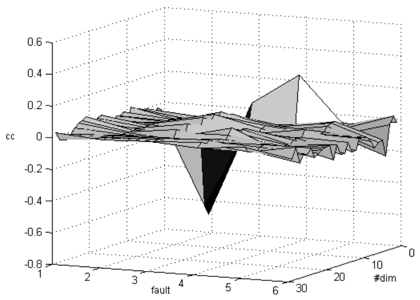
Adapt-kNN shows the highest performance increase for both PCA, LDA and LapEig both for low and medium numbers of dimensions. Not until the last numbers of dimension does Perm catch up with the other preprocessing methods.



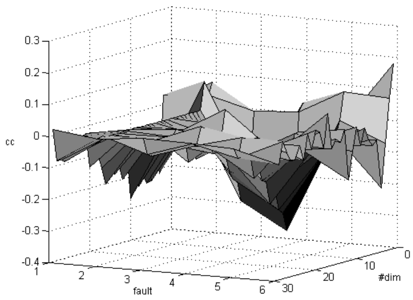
(a) Averages, solid: Perm, dashed: PCA, dotted: LDA, dash-dotted: LapEig



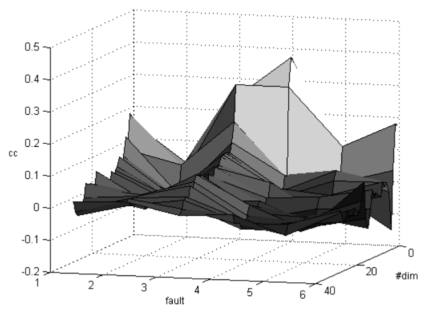
(b) Perm



(c) PCA-Perm-difference

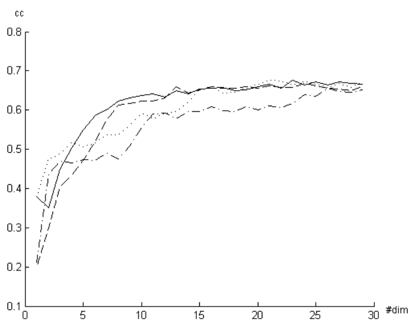


(d) LDA-Perm-difference

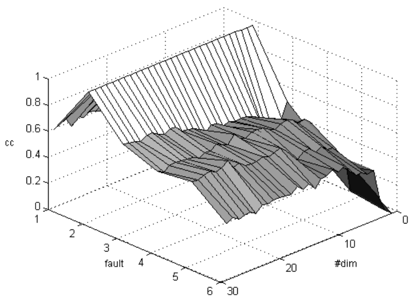


(e) LapEig-Perm-difference

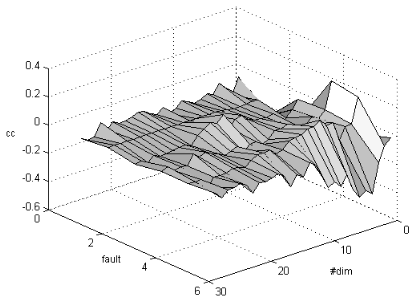
Figure 7.14. Results for Scania using kNN, good results for LapEig for few dimensions



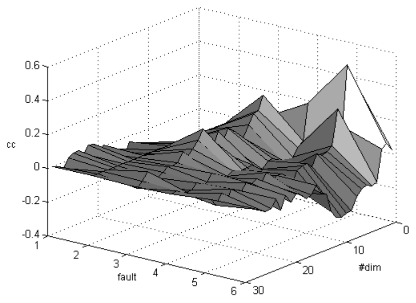
(a) Averages, solid: Perm, dashed: PCA, dotted: LDA, dash-dotted: LapEig



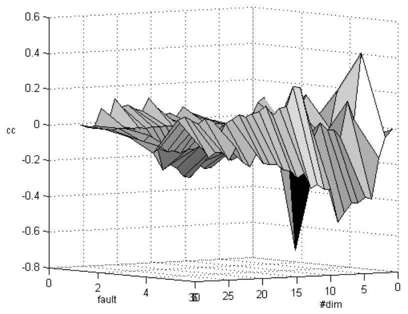
(b) Perm



(c) PCA-Perm-difference

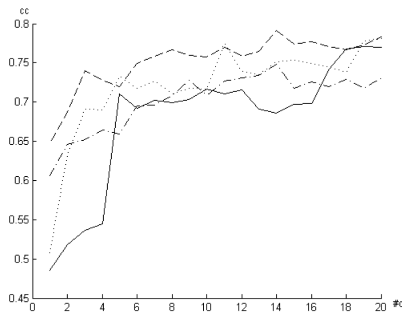


(d) LDA-Perm-difference

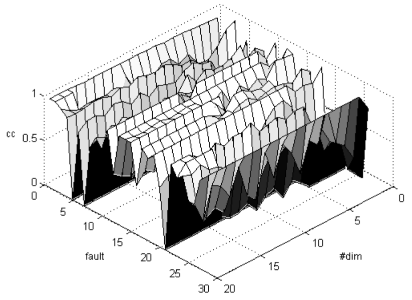


(e) LapEig-Perm-difference

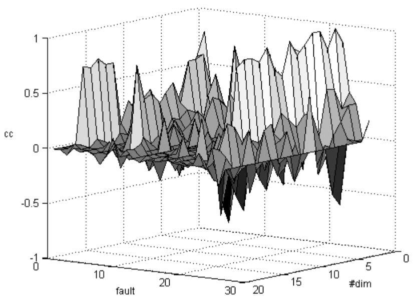
Figure 7.15. Results for Scania using SVM, somewhat good results for LDA for few dimensions



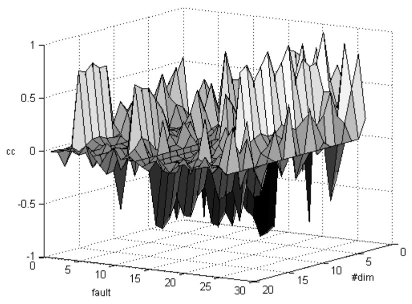
(a) Averages, solid: Perm, dashed: PCA, dotted: LDA, dash-dotted: LapEig



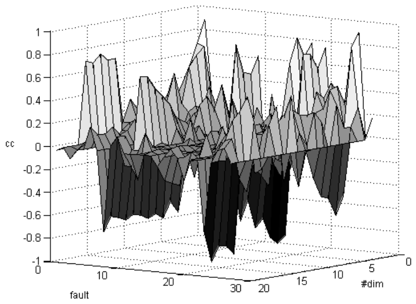
(b) Perm



(c) PCA-Perm-difference



(d) LDA-Perm-difference



(e) LapEig-Perm-difference

Figure 7.16. Results for Adapt using kNN, all three methods outperforms Perm for few dimensions

7.5 Performance With Requirements

To simulate a situation where there are requirements on the diagnosis performance on various faults, three different scenarios are created. These scenarios are varied in the faults on which the requirements are put, in the number of dimensions that are allowed to be used but not in the classification algorithm used, only kNN and Scania data is used.

These are the three scenarios:

1. It is required that the fault-case 3 faults are identified correctly at least 90 % of the time, all dimension are available. Adjust the classifier for optimal performance while upholdning the requirements.

2. It is required that both the fault case 4 and 5 faults are identified correctly at least 50 % of the time, only two dimension are available for classification. Adjust the classifier for optimal performance while upholdning the requirements.

3. It is required that the fault case 6 faults are identified correctly at least 90 % of the time using only ten dimensions for classification. Adjust the classifier for optimal performance while upholdning the requirements.

To achieve these results kNN is biased towards the specified faults by inserting "false neighbors" into the data. These false neighbors are coordinate-less but are assumed to be neighbors of every sample. Since kNN uses an integer number of neighbors, k is increased from 3 to 11 to allow a more gradual impact of the false neighbors. The number of false neighbors is increased until the specified goal is fulfilled.

A good result for these scenarios is a result where the impact of the false neighbor bias is small on the faults not specified. The results are listed in Table 7.3 to 7.5. The best result for every fault as well as best average result is written in bold style. Just as when measuring performance without requirements, every execution is run 25 times.

LapEig performs best for two out of six faults and in total in Scenario 1, the difference between the different preprocessing methods is very small however, and can be considered to be within the margin of error considering the relatively low number of executions.

In Scenario 2, LapEig and to some extent LDA outperforms Perm, but LDA was not able to uphold the requirement completely. LapEig outperforms the other methods for all faults except the second one, which seems consistant with the high results for low dimensions achieved in Figure 7.14.

The results of Scenario 3 are also similar to those in Figure 7.14, but for high dimensions. Some preprocessing methods perform better for certain faults but in total Perm performs best.

Table 7.3. Results for scenario 1 as ratios of correct diagnoses, LapEig achieves the highest average score by a small marginal

	Perm	**PCA**	**LDA**	**LapEig**
Fault 1	0.4355	**0.4585**	0.4264	0.4412
Fault 2	**0.9854**	0.9844	0.9820	0.9352
Fault 3	0.9170	**0.9276**	0.9094	0.9051
Fault 4	0.5659	0.5987	0.5598	**0.5993**
Fault 5	0.5556	0.5673	**0.5935**	0.5844
Fault 6	0.3488	0.3299	0.3745	**0.4130**
Average	0.6347	0.6444	0.6409	**0.6464**

Table 7.4. Results for scenario 2 as ratios of correct diagnoses, LapEig achieves the highest average score by a large margin

	Perm	**PCA**	**LDA**	**LapEig**
Fault 1	0.1054	0.1899	0.1280	**0.5575**
Fault 2	**1.0000**	0.3441	**1.0000**	0.9247
Fault 3	0.2978	0.3489	0.4126	**0.5765**
Fault 4	0.5518	0.6447	0.6273	**0.6727**
Fault 5	0.5105	0.5962	0.4578	**0.6162**
Fault 6	0.0336	0.0279	0.1879	**0.3822**
Average	0.4165	0.3586	0.4689	**0.6216**

Table 7.5. Results for scenario 3 as ratios of correct diagnoses, Perm achieves the highest average score by a small margin

	Perm	**PCA**	**LDA**	**LapEig**
Fault 1	**0.1440**	0.0688	0.1081	0.1371
Fault 2	**1.0000**	0.9117	**1.0000**	0.8657
Fault 3	0.4142	0.3722	**0.4577**	0.3936
Fault 4	0.4613	0.3893	0.3760	**0.5122**
Fault 5	**0.1673**	0.0538	0.1521	0.1048
Fault 6	0.9077	**0.9683**	0.9302	0.9541
Average	**0.5158**	0.4607	0.5040	0.4946

Chapter 8

Evaluation And Conclusions

The purpose of this thesis was to measure the perfomance of three different manifold learning methods, principal component analysis (PCA), linear discriminant analysis (LDA) and laplacian eigenmaps (LapEig), at improving “black box” fault diagnosis.

To fulfill this purpose, a set of practical steps were undertaken:

- Standards of performance was defined.
- A modular system for performance measuring was developed.
- Modules for Scania/Adapt data loading -and normalization, Perm, PCA, LapEig and kNN were developed.
- Modules for LDA and SVM were imported and adapted to the system.
- Parameters for Perm, LapEig, kNN and SVM were calibrated.
- Performance was measured using four different preprocessing methods and two different classification methods on the two data sets Scania and Adapt according to various performance standards.

8.1 Absolute Results

Neither of the manifold learning methods managed to achieve an improvement in absolute results, i.e. the best results were achieved when using all dimensions and the baseline results were as good or better than those for PCA, LDA or LapEig, for both data sets. The following conclusions are infered from these fact:

- Neither PCA or LDA managed to “project away” any noise that could worsen results when using all dimensions. This could have been possible for data sets residing on linear manifolds, resulting in better results for lower dimensions.

- Benefits 2 and 3 in Section 2.3.2, that noise would be removed and that the new distances would have more meaning, were probably not achieved, at least not in a large enough scale. Given that the data actually resided on some manifold, this is attributed to either le's inability to learn the manifold or the inaccurrate representation of the manifold.

8.2 Low Dimension Results

For few dimensions, LapEig managed to achieve far better results when used together with kNN on the Scania data. On the Adapt data, all three preprocessing methods produced better results then the baseline for few dimensions. SVM, which was only run on the Scania data was hardly improved at all by either preprocessing method. The following conclusions can be drawn from these facts:

- Since all three preprocessing methods had good results on Adapt, it seems likely that this data relies on a manifold that is at least in part linear. This is natural since many electronical systems (like that from which Adapt is recorded) are linear.
- The poor results of PCA and LDA on Scania data compared to those of the baseline or of LapEig suggests that the the Scania data relies on a non-linear manifold, which seems likely since it's recorded from a complex system.
- Some data sets are more suitable for manifold learning than others. Linear manifolds seem to be easier to learn, but the interesting ones to non-linear methods are of course the non-linear manifolds.

An important observation is that the relevance of the results for few dimensions was due to hopes of lower computations times. Since the preprocessing time for all metods was far greater than any possible classification time reduction due to lower dimensionality, the time argument does not hold. Even if a greater portion of the information is compressed into fewer dimension as shown in Figures 7.3 to 7.11, this does not seem to have any practical uses.

8.3 Performance With Requirements Results

As with low dimensional results, there were some improvements at the 2-dimension scenario. There were no major difference in results when using all dimensions and small decreases in results for 10 dimensions. The conclusions for these scenarios are the same as for the results mentioned in Section 8.2. Because of the actual time it takes to preprocess data, the scenarios where only a few dimension are used do not seem very realistic.

8.4 Improvements and Possible Future Work

First of all, if manifold learning will ever be useful for any fault diagnosis, it must clearly be a viable choice for black box classification so that it actually improves results either in terms of time or in diagnosis results. To improve results beyond those of not using manifold learning, i.e. so that it uses the manifold information, two things are needed: More and better data and better manifold learning methods. Ideally, the method should be able to compute a manifold coordinate for a sample independently of the other samples, to make preprocessing faster.

Second, manifold learning competes with system knowledge, so model based diagnosis where the model is well known will always be the better choice. As long as the known model of the system is better than the manifold approximation, it is believed that model based diagnosis will perform better than black box diagnosis.

Bibliography

[1] Belkin. *Problems of Learning on Manifolds*. PhD thesis, The University of Chicago, 2003.

[2] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition, November 2000.

[3] Henrik Einarsson and Gustav Ahrrenius. Automatic Design of Diagnosis Systems Using Consistency Based Residuals. Master's thesis, Uppsala University, 2004.

[4] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[5] Quansheng Jiang, Minping Jia, Jianzhong Hu, and Feiyun Xu. Machinery fault diagnosis using supervised manifold learning. *Mechanical Systems and Signal Processing*, 23(7):2301 – 2311, 2009.

[6] Min Li, Jinwu Xu, Jianhong Yang, Debin Yang, and Dadong Wang. Multiple manifolds analysis and its application to fault diagnosis. *Mechanical Systems and Signal Processing*, 23(8):2500 – 2509, 2009.

[7] Aleix M. Martinez and Avinash C. Kak. Pca versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):228–233, 2001.

[8] Editors O. Mcgraw-Hill. *The McGraw-Hill Dictionary of Scientific and Technical Terms, Seventh Edition (Mcgraw Hill Dictionary of Scientific and Technical Terms)*. McGraw-Hill Professional, 7 edition, October 2009.

[9] M. Nyberg and E. Frisk. *Model Based Diagnosis of Technical Processes*. Linköping University Electronic Press, 2008.

[10] Anna Pernestål. *Probabilistic Fault Diagnosis with Automotive Applications*. PhD thesis, Linköping UniversityLinköping University, Vehicular Systems, The Institute of Technology, 2009.

[11] Rowland. Manifold. `http://mathworld.wolfram.com/Manifold.html`.

[12] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, 2 edition, December 2002.

[13] Ashok Srivastava. Adapt dataset. `https://dashlink.arc.nasa.gov/data/adapt-an-electrical-power-system-testbed/`.

[14] Ashok Srivastava. Diagnostic competition. `https://dashlink.arc.nasa.gov/topic/diagnostic-challenge-competition\`.