

# Principal Component Analysis - An Introduction

Thomas B. Moeslund

Laboratory of Computer Vision and Media Technology  
Aalborg University, Niels Jernes Vej 14  
DK-9220 Aalborg East, Denmark  
Email: [tbm@cvmt.auc.dk](mailto:tbm@cvmt.auc.dk)

## Preface

This report gives an introduction to Principal Component Analysis and can be used in connection with teaching. Readers are expected to be familiar with linear algebra and stochastic signal processing.

Figures, tables, and equations are each numbered as X.Y, where X indicates the chapter and Y is a consecutive number. For example, equation 4.3 refers to the third equation in the fourth chapter. All vectors are column-vectors and indicated by a boldfaced lower case letter or symbol, e.g.  $\mathbf{e}$ . Matrices are indicated by a boldfaced upper case letter or symbol, e.g.  $\mathbf{V}$ .

Throughout the report the principal components plays a major role. These are denoted; new basis vectors, variables, components, and eigenvectors, dependent on the context.

Comments, errors, etc. should be reported to [tbm@cvmt.auc.dk](mailto:tbm@cvmt.auc.dk)

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	The Basic Principle . . . . .	4
<b>2</b>	<b>The Mathematics Behind the PCA</b>	<b>7</b>
2.1	The Eigenvalue Problem . . . . .	9
2.2	Properties of the Transformation . . . . .	10
2.2.1	The Covariance Matrix of the Input Data . . . . .	10
2.2.2	The Covariance Matrix of the Transformed Data . . . . .	11
<b>3</b>	<b>Choosing which Components to Ignore</b>	<b>13</b>
3.1	The m-method . . . . .	13
3.2	Other Techniques . . . . .	14
3.2.1	The J-measure . . . . .	16
3.2.2	The SEPCOR-algorithm . . . . .	16
3.3	Comparing the Different Methods . . . . .	18
<b>4</b>	<b>Summary</b>	<b>19</b>
4.1	How to Do a PCA . . . . .	20
4.1.1	Acquire Data . . . . .	20
4.1.2	Calculate Covariance Matrix . . . . .	20

4.1.3	Calculate Eigenvalues and Eigenvectors . . . . .	21
4.1.4	Chose Eigenvectors . . . . .	21
4.1.5	Map the Data . . . . .	21

# Chapter 1

## Introduction

The Principal Component Analysis (PCA), also known as the Hotelling transform, and the Karhunen Loeve expansion, has been known for many years. The two first to deal with PCA was Pearson (1901) and Hotelling (1933). Pearson's approach was concerned with finding lines and planes that best fitted a set of points in a  $n$  dimensional space and this research lead to the principal component (PC). Hotelling's approach was to find a smaller set of fundamental variables which expressed the original  $n$  variables. Hotelling noted that such variables were called factors in the psychological literature and since the word factor already had a meaning in the mathematical world he denoted the variables 'components'. Hotelling chose to maximise his 'components' in the sense of variance in the original variables and these 'components' he called 'principal components'. Both Pearson and Hotelling ended up with the eigenvalue problem, which is hard to solve for orders above four. Pearson claimed that his method for solving this problem was efficient also for higher than 4'th order but still a lot of calculations were involved which was hard to do without any aid from a computer. Since PCA is more efficient the higher dimension the problem is, not much work was done in the area of PCA until the sixties simply because it would take to long time to see the result of the PCA. Then, due to the new interest in statistics and the aid of computer power, the PCA became object of new interest and a lot of research was done in the interpretation and application of the PCA. Today the PCA is a well-known tool which is used in a variety of different fields.

The report at hand gives an introduction to PCA, by describing the basics of the transformation, deriving the theory, and finally summarising the presented information into a short and precise how-to section.

### 1.1 The Basic Principle

As other transformations, e.g. Fourier transformation, PCA transforms data into another representation where a new set of basis vectors (variables) are used. In PCA, however, these basis vectors are not constant as is the case in many other transformations. Instead they are calculated based on the data to be transformed.

As for other transformations PCA is designed to enhanced certain aspects of the data set. In

Fourier transformation it is the frequencies in the data set. In PCA it is the variance in the data set.

PCA is a linear transformation and the new basis vectors, denoted  $\mathbf{e}_i$ , are therefore subjected to an orthonormality constraint, i.e.

$$\mathbf{e}_i^T \mathbf{e}_j = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad (1.1)$$

where  $\delta_{ij}$  is Dirac's delta function.

Since PCA is a linear transformation with orthonormal basis vectors it can be expressed as a translation and rotation. Denoting the input data  $\mathbf{x}$  and the transformed data  $\mathbf{y}$ , the transformation can be expressed as

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \boldsymbol{\mu}_x) \quad (1.2)$$

where  $\mathbf{A}$  contains the new basis vectors,  $\mathbf{e}_i$  as row vectors, hence  $\mathbf{A} = [\mathbf{e}_1 \ \mathbf{e}_2 \ \dots \ \mathbf{e}_n]^T$ , and  $\boldsymbol{\mu}_x$  is the mean of the data set, hence  $\boldsymbol{\mu}_x = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$

In figure 1.1 the basic principles of PCA is illustrated for the two dimensional case.

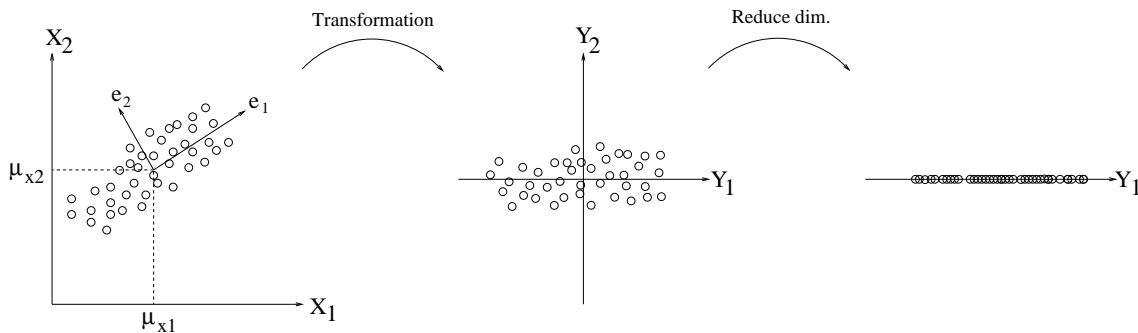


Figure 1.1: The basic principle of the PCA.

The first figure illustrates the input where the  $i$ 'th sample is denoted  $\mathbf{x}_i = [x_{1i} \ x_{2i}]^T$ . The second illustrates the transformed data where the  $i$ 'th sample is denoted  $\mathbf{y}_i = [y_{1i} \ y_{2i}]^T$  and calculated using equation 1.2.

The first two figures illustrate how the data are transformed into another representation where the main part of the variance of the data is represented in the first variable,  $\mathbf{y}_1$ . That is, if the second variable is ignored, as illustrated in the last figure, the main variance of the data is kept. In many cases variance equals information, hence a more compact representation of the data/information is obtained.

The process of figuring out which variables to ignore is referred to as the analysis part of PCA, hence the  $A$  in PCA.

In the example in figure 1.1 the compactness achieved by ignoring one variable might not seem significant. However, imagine that the dimensionality of the input data is 20 and that the main part of the variance could be represented by, say only two variables, then a truly compact representation would be obtained. For this reason PCA is mainly applied in cases where high dimensional data are present.

Besides being able to reduce the dimensionality of the data PCA also has another important property, namely a de-correlation of the data with respect to the new variables. As can be seen in figure 1.1 the data are uncorrelated in  $\mathbf{y}_1$  and  $\mathbf{y}_2$ . This property also holds for higher dimensional problem. I.e. independent of the dimensionality of the input data, the transformed data will be uncorrelated in all variables. This property is described more thoroughly in a later section.

The rest of this report is divided into three chapters. In the first the theory behind PCA is derived and the properties of the transformation is presented. In the next chapter it is described how to chose those components to be ignored. In the last chapter a summary of PCA is given together with a concise how-do description.

## Chapter 2

# The Mathematics Behind the PCA

As described above the transformation is given as

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \boldsymbol{\mu}_x) \quad (2.1)$$

where  $\mathbf{A} = [\mathbf{e}_1 \ \mathbf{e}_2 \ \cdots \ \mathbf{e}_n]^T$ , and  $\boldsymbol{\mu}_x = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$ . To utilise this transformation  $\mathbf{A}$  needs to be found. In this chapter it is described how  $\mathbf{A}$  can be found.

For simplicity 2.1 is written as  $\mathbf{y} = \mathbf{A}\mathbf{x}'$  where  $\mathbf{x}' = \mathbf{x} - \boldsymbol{\mu}_x$  and the inverse transformation is given as  $\mathbf{x}' = \mathbf{A}^{-1}\mathbf{y}$ . Since  $\mathbf{A}$  is an orthogonal matrix<sup>1</sup>  $\mathbf{A}^{-1} = \mathbf{A}^T$ , hence  $\mathbf{x}' = \mathbf{A}^T\mathbf{y}$ . Rewriting yields

$$\mathbf{x}' = [\mathbf{e}_1 \ \mathbf{e}_2 \ \cdots \ \mathbf{e}_n] \cdot \mathbf{y} = \sum_{i=1}^n y_i \mathbf{e}_i \quad (2.2)$$

If only  $m$  ( $m < n$ ) components are used instead of  $n$  some variance (information) is lost doing the inverse transformation and  $\mathbf{x}'$  can not be reconstructed. Instead an estimate is produced

$$\hat{\mathbf{x}}' = \sum_{i=1}^m y_i \mathbf{e}_i \quad (2.3)$$

The task at hand is to find  $\mathbf{A}$  (or rather the principal components  $\mathbf{e}_i$ ) so that the expected squared difference between  $\mathbf{x}'$  and  $\hat{\mathbf{x}}'$

$$\alpha = E \left\{ (\mathbf{x}' - \hat{\mathbf{x}}')^T (\mathbf{x}' - \hat{\mathbf{x}}') \right\} \quad (2.4)$$

---

<sup>1</sup>Note that an orthogonal matrix, despite its name, actually consist of orthonormal vectors.

is minimised, given the constraint in 1.1.

Inserting equation 2.2 and 2.3 in equation 2.4 yields

$$\alpha = E \left\{ \left( \sum_{i=m+1}^n y_i \mathbf{e}_i \right)^T \left( \sum_{i=m+1}^n y_i \mathbf{e}_i \right) \right\} \quad (2.5)$$

Due to the orthonormality constraint in equation 1.1 the product of the two sums yields

$$\alpha = E \left\{ \sum_{i=m+1}^n y_i^2 \right\} \quad (2.6)$$

From equation 2.1 it follows that  $y_i = \mathbf{e}_i^T \mathbf{x}'$ , yielding

$$\alpha = E \left\{ \sum_{i=m+1}^n (\mathbf{e}_i^T \mathbf{x}')^2 \right\} = E \left\{ \sum_{i=m+1}^n (\mathbf{e}_i^T \mathbf{x}') (\mathbf{e}_i^T \mathbf{x}') \right\} \quad (2.7)$$

Since  $\mathbf{e}_i^T \mathbf{x}' = \mathbf{x}'^T \mathbf{e}_i$  equation 2.7 can be written as

$$\alpha = E \left\{ \sum_{i=m+1}^n (\mathbf{e}_i^T \mathbf{x}') (\mathbf{x}'^T \mathbf{e}_i) \right\} = E \left\{ \sum_{i=m+1}^n \mathbf{e}_i^T \mathbf{x}' \mathbf{x}'^T \mathbf{e}_i \right\} \quad (2.8)$$

Since  $\mathbf{e}_i$  is deterministic the order of the summation and the expectation operation can be interchanged

$$\alpha = \sum_{i=m+1}^n \mathbf{e}_i^T E \left\{ \mathbf{x}' \mathbf{x}'^T \right\} \mathbf{e}_i \quad (2.9)$$

Denoting the term in the brackets  $\mathbf{C}_x$  yields

$$\alpha = \sum_{i=m+1}^n \mathbf{e}_i^T \mathbf{C}_x \mathbf{e}_i \quad (2.10)$$

The squared error function must now be minimised in order to find the optimal  $\mathbf{e}_i$ , hence  $\mathbf{A}$ . This is done by minimising each term in the usually way by partial differentiation and setting the result equal to zero. However the differentiation must be done in such a way that the condition in equation 1.1 still holds. This problem is solved by generating a constrained function  $g(\mathbf{e}_i)$  using the technique of Lagrange multiplier

$$g(\mathbf{e}_i) = \mathbf{e}_i^T \mathbf{C}_x \mathbf{e}_i - \lambda (\mathbf{e}_i^T \mathbf{e}_i - 1) \quad (2.11)$$

Partial differentiation of equation 2.11 and setting it equal to zero yields

$$\nabla g(\mathbf{e}_i) = \mathbf{0} \Rightarrow \quad (2.12)$$

$$\nabla g(\mathbf{e}_i) = \mathbf{C}_x \mathbf{e}_i + \mathbf{C}_x^T \mathbf{e}_i - \lambda 2\mathbf{e}_i = \mathbf{0} \quad (2.13)$$

Since  $\mathbf{C}_x = \mathbf{C}_x^T$  equation 2.13 can be written as

$$\mathbf{C}_x \mathbf{e}_i - \lambda \mathbf{e}_i = \mathbf{0} \Rightarrow \quad (2.14)$$

$$(\mathbf{C}_x - \lambda \mathbf{I}) \cdot \mathbf{e}_i = \mathbf{0} \quad (2.15)$$

where  $\mathbf{I}$  denotes the identity matrix.

## 2.1 The Eigenvalue Problem

Equation 2.15 is known as the eigenvalue problem and is a general mathematical problem which appears in many other areas besides PCA. To solve the problem the determinant of the coefficient matrix is used. Since equation 2.15 is a homogeneous system of linear equations and since it has a nontrivial solution the determinant of the coefficient matrix is equal to zero. In other words

$$\det(\mathbf{C}_x - \lambda \mathbf{I}) = 0 \quad (2.16)$$

By developing equation 2.16 the characteristic polynomial is obtained and the roots of this polynomial are the eigenvalues,  $\lambda_i$ , of  $\mathbf{C}_x$ . In the eigenvalue problem  $\lambda_i$  is the  $i$ 'th eigenvalue corresponding to the  $i$ 'th eigenvector  $\mathbf{e}_i$  and  $\lambda_i \geq \lambda_{i+1}$ . When the eigenvalues are found they are each substituted into equation 2.15 and the corresponding eigenvector is found. Finding the roots of the characteristic polynomial analytically is very hard when the order is above four and therefore numerical methods like, the power method, the Jacobi method, the Givens method, and the Householder methods are used [1].

After finding the eigenvectors,  $\mathbf{e}_i$ , they are arranged as row vectors in the  $\mathbf{A}$ -matrix. All entities of the transformation in equation 2.1 have now been found and it is ready for use.

Recall that finding the new basis vectors as the eigenvectors of the covariance matrix, as done above, is the optimal solution in terms of minimising the amount of information (variance) lost when ignoring a number of variables, given the orthonormality constraint.

## 2.2 Properties of the Transformation

### 2.2.1 The Covariance Matrix of the Input Data

When the theory behind the PCA was derived above the following substitution was made

$$E \left\{ \mathbf{x}' \mathbf{x}'^T \right\} = \mathbf{C}_x \quad (2.17)$$

Expanding  $\mathbf{x}'$  into  $\mathbf{x} - \boldsymbol{\mu}_x$  yields

$$\mathbf{C}_x = E \left\{ (\mathbf{x} - \boldsymbol{\mu}_x) (\mathbf{x} - \boldsymbol{\mu}_x)^T \right\} \quad (2.18)$$

which is the covariance matrix of  $\mathbf{x}$ , i.e.  $\text{Covar}(\mathbf{x})$ . The covariance matrix of  $\mathbf{x}$  is estimated as

$$\mathbf{C}_x = \frac{1}{n} \sum_{i=1}^n (\mathbf{x} - \boldsymbol{\mu}_x) (\mathbf{x} - \boldsymbol{\mu}_x)^T \quad (2.19)$$

Denoting  $\mathbf{V} = [\mathbf{x}'_1 \ \mathbf{x}'_2 \ \dots \ \mathbf{x}'_n]$  and observing that  $\frac{1}{n}$  only scales the eigenvalues<sup>2</sup> equation 2.19 can be written as

$$\mathbf{C}_x = \mathbf{V} \mathbf{V}^T \quad (2.20)$$

The complexity involved in solving the eigenvalue problem is strongly depended on the size of the covariance matrix. If its size is large so is the complexity of the eigenvalue problem. The following method might be applied to reduce the complexity.

The eigenvalue problem can be expressed as

$$\mathbf{V} \mathbf{V}^T \mathbf{e}_i = \lambda \mathbf{e}_i \quad (2.21)$$

Instead of solving the problem in equation 2.21 consider the following problem

$$\mathbf{V}^T \mathbf{V} \mathbf{b}_i = \lambda \mathbf{b}_i \quad (2.22)$$

where  $\mathbf{b}_i$  denotes the eigenvector.

---

<sup>2</sup>This does not change anything since only the relation between the eigenvalues is applied.

Multiplying equation 2.22 with  $\mathbf{V}$  yields

$$\mathbf{V}\mathbf{V}^T\mathbf{V}\mathbf{b}_i = \mathbf{V}\lambda\mathbf{b}_i \Rightarrow \quad (2.23)$$

$$\mathbf{V}\mathbf{V}^T(\mathbf{V}\mathbf{b}_i) = \lambda(\mathbf{V}\mathbf{b}_i) \quad (2.24)$$

Comparing equation 2.21 and equation 2.24 reveals that  $\mathbf{e}_i = \mathbf{V}\mathbf{b}_i$ . This means that by solving equation 2.22 and multiplying the eigenvector  $\mathbf{b}_i$  by  $\mathbf{V}$ , the eigenvector of the original problem is found. Which of the two problems to solve depends on the dimensionality of the input data,  $n$ , and the number of input samples,  $s$ . The dimensionality of  $\mathbf{V}$  is  $n$  times  $s$  so the dimensionality of the covariance matrix and therefore the eigenvalue problem is  $n$  times  $n$  in the first problem and  $s$  times  $s$  in the second problem. The following rule can be applied

Solve problem one, hence equation 2.21, if  $n \leq s$

Solve problem two, hence equation 2.24, if  $n > s$

## 2.2.2 The Covariance Matrix of the Transformed Data

The covariance matrix of the transformed data is defined as

$$\mathbf{C}_y = E\left\{(\mathbf{y} - \boldsymbol{\mu}_y)(\mathbf{y} - \boldsymbol{\mu}_y)^T\right\} \quad (2.25)$$

Using equation 2.1 together with the fact that  $\boldsymbol{\mu}_y = 0$  and the following rule  $(\mathbf{c} \cdot \mathbf{d})^T = \mathbf{d}^T \cdot \mathbf{c}^T$  equation 2.25 can be re-written as

$$\mathbf{C}_y = E\left\{\left(\mathbf{A}(\mathbf{x} - \boldsymbol{\mu}_x)\right)\left((\mathbf{x} - \boldsymbol{\mu}_x)^T \mathbf{A}^T\right)\right\} \quad (2.26)$$

Removing the parenthesis and observing that  $\mathbf{A}$  is deterministic, the order of the multiplication and the expectation operation can be interchanged, hence

$$\mathbf{C}_y = \mathbf{A} \cdot E\left\{(\mathbf{x} - \boldsymbol{\mu}_x)(\mathbf{x} - \boldsymbol{\mu}_x)^T\right\} \cdot \mathbf{A}^T = \mathbf{A}\mathbf{C}_x\mathbf{A}^T \quad (2.27)$$

$\mathbf{A}$  is an orthogonal matrix, hence  $\mathbf{A}^{-1} = \mathbf{A}^T$ . Equation 2.27 can therefore be written as

$$\mathbf{C}_x\mathbf{A}^T = \mathbf{A}^T\mathbf{C}_y \quad (2.28)$$

From the definition of  $\mathbf{A}^T$  and equation 2.14 it follows that

$$\mathbf{C}_x \mathbf{A}^T = \begin{bmatrix} \lambda_1 \mathbf{e}_1 & \lambda_2 \mathbf{e}_2 & \cdots & \lambda_n \mathbf{e}_n \end{bmatrix} \quad (2.29)$$

Setting the right-hand side of equation 2.28 equal to the right-hand side of equation 2.29 yields

$$\mathbf{C}_y = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix} \quad (2.30)$$

This means that all the covariances of  $\mathbf{C}_y$  are zero, hence the data are uncorrelated with respect to the different variables, i.e.  $\mathbf{y}_1, \mathbf{y}_2$  etc . Furthermore, the variances of the variables are equal to the eigenvalues, i.e. the variance of the data on the  $i$ 'th variable is  $\lambda_i$ .

## Chapter 3

# Choosing which Components to Ignore

Since the eigenvectors have been found the input data can be transformed. The dimension of the transformed data is the same as the dimension of the input data so nothing is gained so far in terms of data reduction. The problem at hand is now to reduce the number of required components as much as possible without losing too much information.

An initial reduction is given if the number of samples is less than the number of dimensions in the input, e.g., if only two samples exist then they can be represented with only one eigenvector since any two points in space lay on a common line in space. If three points are present then they, at the most, span a plane which can be represented by two eigenvectors etc. This means that only a certain number of eigenvectors having corresponding non-zero eigenvalues exist if the number of samples is less than the number of dimension. All the eigenvectors having zero-eigenvalues can be ignored without losing any information. The number of zero-eigenvalues,  $z$ , is calculated as

$$z = n - (s - 1) = n - s + 1 \quad (3.1)$$

where  $n$  denotes the dimension of the input data and  $s$  denotes the number of samples. Obviously, equation 3.1 only apply if  $n \geq s - 1$ .

In the following three methods which can help to decide which components to ignore are described.

### 3.1 The m-method

Normally it is desired to reduce the dimensionality as much as possible and therefore the initial reduction is not enough. Generally speaking the task of determining how much the dimensionality can be reduced is a matter of representing as much information as possible in as small a space as possible. In other words, to determinate how many eigenvectors to ignore is a tradeoff between the

wanted low dimension and the unwanted loss of information. Since the  $i$ 'th eigenvalue is equal to the variance of the  $i$ 'th variable and since  $\lambda_i \geq \lambda_{i+1}$  the tradeoff can be defined as

$$I_k = \frac{\sum_{i=1}^m \lambda_i}{\sum_{i=1}^n \lambda_i} \cdot 100\% \quad (3.2)$$

where  $\lambda_i$  denotes the  $i$ 'th eigenvalue,  $m$  denotes the number of eigenvectors that is used,  $n$  denotes the dimension of the input data, and  $I_k$  denotes the percentage of information (variance) that is kept in the transformation.

The equation can also be illustrated as shown in figure 3.1 which has the successive eigenvalues on the first axis and the associated eigenvectors on the second axis. On the graph  $m$  is the threshold between ignored and selected eigenvectors, the area  $I_1$  represents the information kept in the transformation and  $I_2$  represent the information lost in the transformation. The larger  $m$  is the more information is kept in the transformation. This technique is called the  $m$ -method.

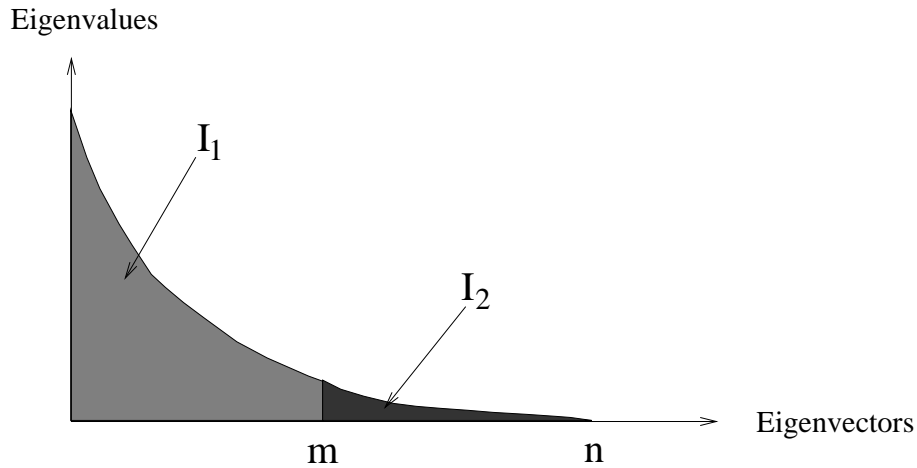


Figure 3.1: The lost and kept information when ignoring  $(n - m)$  eigenvectors.

## 3.2 Other Techniques

The  $m$ -method is used in many applications since it is easy and straight forward. However, the method does have its drawbacks and therefore it is interesting to see if a better result can be obtained using another method.

When the input data originates from different classes it is not always a good idea to chose the eigenvectors corresponding to the  $m$  highest eigenvalues. This is illustrated in figure 3.2 where figure 3.2.A shows the transformation of data from two different classes in 2D. For the example in figure 3.2.A the  $m$ -method will ignore the  $y_2$ -axis since it has less information than the other axis, in the sense of variance, but it is clear that it is easier to classify if the  $y_1$ -axis is ignored instead.

If the classes are compact like in figure 3.2.B, then the  $m$ -method will work fine since it will ignore the  $y_2$ -axis. This suggest the need for another method when the classes are not compact. The

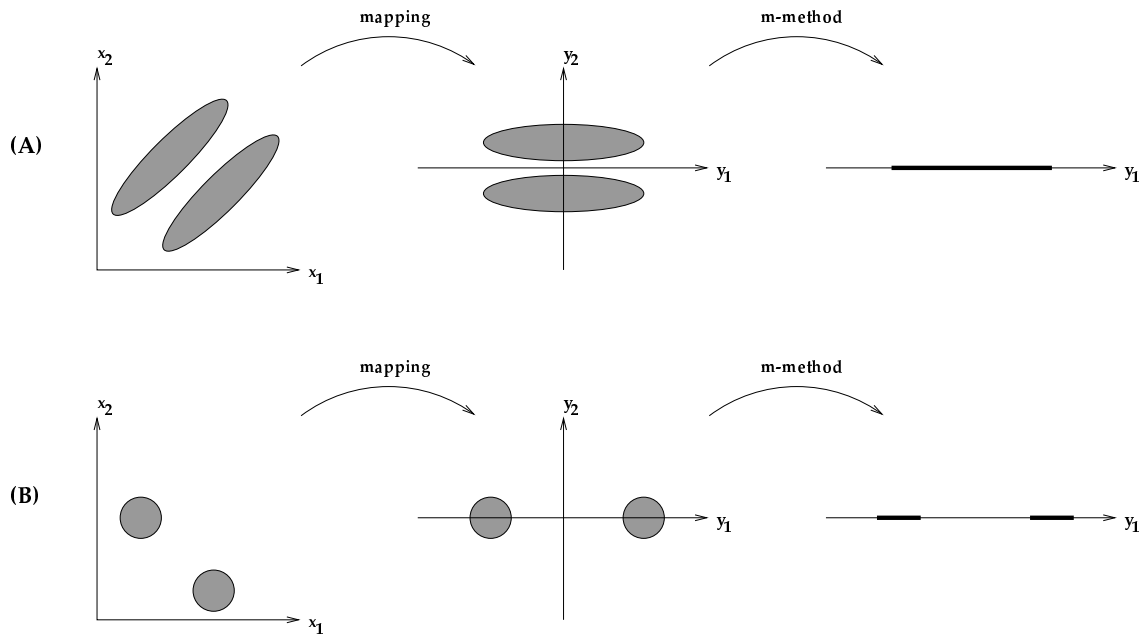


Figure 3.2: A: shows the mapping of two classes into the component space and the result when the second axis is ignored. B: shows the same but for two different classes.

method should be able to find those eigenvectors which separates the classes best instead of the eigenvectors with the highest eigenvalues.

Basically all the combinations of eigenvectors can be tried out and for each combination it can be calculated how good the combination is to separate the classes. When this is done for each combination, the combination which separates the classes best is chosen. This is a simple and straight forward approach, but unfortunately there are some problems. First of all it is not clear how to decide the separability of each combination and second a lot of combinations exist. Usually the data are high dimensional and numerous samples exist, hence a great number of non-zero eigenvalues are involved in the transformation. Therefore quite a few combinations are possible. Equation 3.3 shows how to calculate the number of combinations,  $S$ ,

$$S = \sum_{i=1}^l \binom{l}{i} = \sum_{i=1}^l \frac{l!}{i!(l-i)!} \quad (3.3)$$

where  $l$  denotes the number of non-zero eigenvectors and  $!$  is the factorial operator.

If 10 non-zero eigenvalues are used there exist 1023 different combinations and the exponent of the number of combinations is approximately doubled each time the number of non-zero eigenvalues is doubled. Clearly this is far too many combinations to go through especially because the number of non-zero eigenvalues might be rather high. Therefore another method is needed.

### 3.2.1 The J-measure

In the m-method the individual classes are not taking into account. However, it seems reasonable to chose the eigenvectors which best separates the classes instead of choosing the ones with the highest eigenvalues. In the literature Kitler [2] introduces a method which finds the eigenvectors which best separates the mean of the individual classes. The method is based of calculating a J-measure for each component and then choosing the components with the highest J-measure. The J-measures is defined as

$$J(\mathbf{e}_i) = \frac{\mathbf{e}_i^T \mathbf{M} \mathbf{e}_i}{\lambda_i} \quad (3.4)$$

where  $J(\mathbf{e}_i)$  denotes the J-measure of the i'th component,  $\mathbf{e}_i$  denotes the i'th eigenvector,  $\lambda_i$  denotes the i'th eigenvalue, and  $\mathbf{M}$  denotes the scatter matrix of the input data. The scatter matrix contains a measure of the variances between the components in each class and is defined as

$$\mathbf{M} = \sum_{i=1}^K P(k_i) (\boldsymbol{\mu}_i - \boldsymbol{\mu}) (\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \quad (3.5)$$

where  $K$  denotes the number of classes,  $P(k_i)$  denotes the probability of the i'th class,  $\boldsymbol{\mu}_i$  denotes the mean of the i'th class in the input data, and  $\boldsymbol{\mu}$  is the mean of the means.

The numerator in equation 3.4 is a measure of how good the i'th component separates the mean of the classes. The higher this value is the better the separation is. Since the eigenvalues are not normalised the numerator depends on the total variance of the component being investigated. To deal with this problem the denominator is present. It contains the variance along the investigated component and under the assumption that a high variance of a component means a high variance of each class the J-measure is normalised.

### 3.2.2 The SEPCOR-algorithm

Another method which uses the mean of the classes is the SEPCOR-algorithm. In addition to the mean it also uses the variance of the classes. The name of the algorithm is derived from SEPerability and CORrelation and the principle of the algorithm is first to order the components in decreasing order with respect to a variability measure and then correlate the components to see if any are redundant. The variability measure is defined as

$$V(\mathbf{e}_i) = \frac{\text{deviation of the class-means}}{\text{mean of the deviation of each class}} = \frac{\sum_{j=1}^K (\mu_{ji} - \mu_i)^2}{\sum_{j=1}^K \sum_{g \in \text{class } j} (y_{gi} - \mu_{ji})^2} \quad (3.6)$$

where  $\mathbf{e}_i$  denotes the i'th component,  $\mu_{ji}$  denotes the mean of class j on the i'th component,  $\mu_i$  denotes the mean of the i'th component,  $y_{gi}$  denotes a sample from a class on the i'th component,

and  $K$  denotes the number of classes.

The numerator represents how much the means of the classes are separated and is therefore preferred large. The denominator represents the compactness of the classes and is therefore preferred small. The larger the variability measure is for a component the better the component is to discriminate the classes. In figure 3.3 the effect of the variability measure is illustrated. The figure shows two classes each represented by their mean and variance. The classes are shown in 2D for illustrative purposes. When only two classes are present equation 3.6 is reduced to

$$V(\mathbf{e}_i) = \frac{1}{2} \frac{(\mu_{1i} - \mu_{2i})^2}{\sigma_{1i}^2 + \sigma_{2i}^2} \quad (3.7)$$

where  $\mu_{ji}$  denotes the mean of class  $j$  on the  $i$ 'th component and  $\sigma_{ji}^2$  denotes the variance of class  $j$  on the  $i$ 'th component.

It should be noted that the numerator is simply the squared distance between the mean of the classes. In the situation showed in figure 3.3 the numerator is approximately the same for the two components and therefore only the denominator has an effect on  $V(\mathbf{e}_i)$ . Clearly the second component has a higher variability measure than the first component so the second component contains more discriminative information, which also seems intuitively correct. It should be noted that the m-method would have chosen the first component as the most significant since the variance is larger on this component.

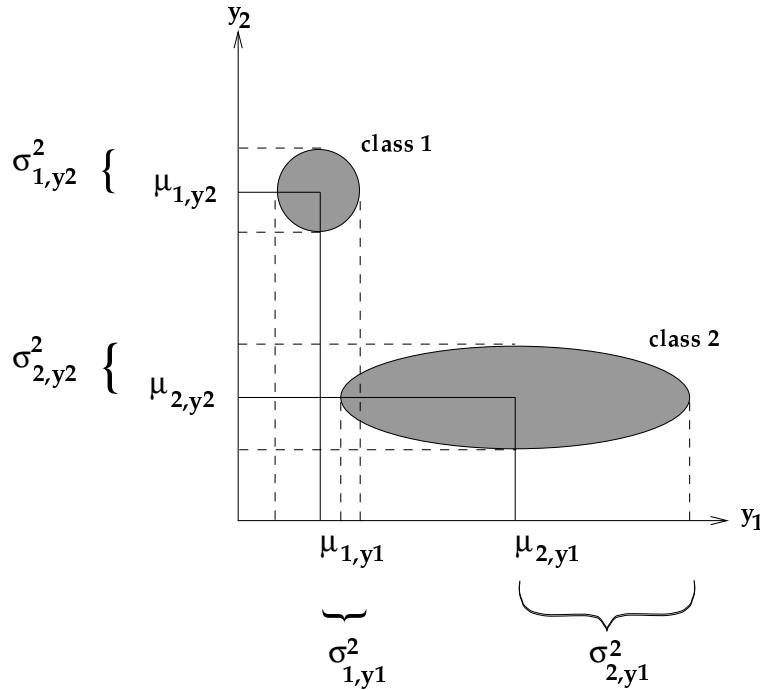


Figure 3.3: An illustration of the parameters in the variability measure used by SEPCOR.

When the variability has been calculated for every component and they have been ordered in decreasing order with respect to  $V(\mathbf{e}_i)$ , a correlation is carried out. First the component with the highest variability measure is correlated with the rest of the components having a variability

measure above a certain threshold. Those components having a high correlation with the first component are ignored since they do not add any new information. Then the component with the second highest variability measure is correlated with the rest of the components which are not ignored. The components with a correlation value above the threshold will then be ignored etc.

When the SEPCOR-algorithm is used on data which have been transformed into a space spanned by the eigenvectors of the covariance of the original data, there is no correlation between the components and therefore the SEPCOR-algorithm is reduced to the variability measure.

### 3.3 Comparing the Different Methods

Which of the three methods to use depends, for one thing, on how similar the classes are. If the classes are symmetric and have similar sizes, meaning the same variance, then the m-method is sufficient. If, however, the classes are very different the more computationally demanding SEPCOR and J-measure algorithms will perform better. One way of figuring out which method to use is to inspect the covariance matrices of the different classes or even better to apply all three methods and chose the one producing the best result for the current application.

It should be noticed that the more alike and symmetric the classes are the closer the result of the J-measure will be to the result of the m-method.

It should also be noticed that the denominator in the variability measure of the SEPCOR-algorithm will have less effect the more symmetric the classes are. If all the classes are completely symmetric the denominator will be exactly the same for every component and could therefore be omitted. Also it should be noticed that the numerator in the variability measure is dependent on the similarity between the classes. If the classes are similar in shape and size the numerator will be proportional to the eigenvalue of this component. This means that if the classes are symmetric and have the same size the variability measures will be proportional to the eigenvalues and the SEPCOR-algorithm will therefore produce the same result as the m-method. This argument does not hold the other way around, i.e. similar results do not necessarily come from symmetric classes with the same size, since different situations results in similar output from the m-method and the SEPCOR-algorithm.

Finally it should be noticed that the three methods mostly tend to differ on low dimensional data or when only a few components are applied to transform a high dimensional data set. As more and more components are applied the methods usually converge toward choosing the same components. For this reason and due to simplicity the m-method is the preferred method in many applications.

# Chapter 4

## Summary

In this report the Principal Component Analysis (PCA) has been described. It is a linear transformation where the input data is transformed into another represented where a new set of variables is used. These variables are also denoted components and are found as the eigenvectors of the covariance matrix of the input data. The transformation can be expressed as a translation followed by a rotation

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \boldsymbol{\mu}_x) \quad (4.1)$$

where  $\mathbf{y}$  is the transformed data,  $\mathbf{x}$  is the input data,  $\mathbf{A}$  contains the eigenvectors as row-vectors, hence  $\mathbf{A} = [\mathbf{e}_1 \ \mathbf{e}_2 \ \cdots \ \mathbf{e}_n]^T$ , and  $\boldsymbol{\mu}_x = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$ .

The purpose of the transformation is twofold. Firstly, the new set of variables are found as the direction in the data set where the highest variance is presents, given the variables are orthonormal. Secondly, a number of the components are ignored hereby reducing the dimensionality of the data. The transformation is optimal with respect to keeping as much variance as possible when ignoring some of the components.

In short, PCA finds the directions in the input data with maximum variance and then ignores those directions (components) with least variance.

Which and how many components to ignore depends on the task PCA is used for. Three methods for choosing which components to ignore are presented. The m-method where those components with least variance are ignored. The other two methods, J-measure and SEPCOR, are applied when the input data originates from different classes. The former takes the mean of the different classes into consideration. This is also the case for the latter but in addition it considers the variances of the different classes.

The transformation has two major properties besides those mentioned above. First of all the mean of the transformed data is zero and secondly the covariance matrix of the transformed data is given as

$$\mathbf{C}_y = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix} \quad (4.2)$$

It can be seen that the covariances are zero which means that the data are uncorrelated with respect to the different variables. Furthermore the variances of the different variables are given as the eigenvalues of the covariance matrix of the input data.

## 4.1 How to Do a PCA

In this section the tasks to be carried out in PCA are listed and explained. To use PCA five things must be done:

- Acquire data
- Calculate covariance matrix
- Calculate eigenvalues and eigenvectors
- Chose eigenvectors
- Map the data

### 4.1.1 Acquire Data

If PCA is used to transform data which have not been included in the covariance matrix then the “training” data have to be representative, or in other words, to contain the variance structure expected to be contained in new samples. In this case the more training data the better. Furthermore, if the PCA is used in a classification problem where the covariance of the classes is used then the dataset must include at least as many data points in each class as the number of chosen eigenvectors. In other words, if it is chosen to use  $m$  eigenvectors then the dataset must include at least  $m$  different samples from each class. This is necessary in order to avoid an underdetermined problem.

### 4.1.2 Calculate Covariance Matrix

The covariance matrix,  $\mathbf{C}_x$ , contains the correlation within the input data. It is estimated as

$$\mathbf{C}_x = \mathbf{V}\mathbf{V}^T \quad (4.3)$$

where  $\mathbf{V} = [\mathbf{x}'_1 \ \mathbf{x}'_2 \ \cdots \ \mathbf{x}'_n]$  and  $\mathbf{x}'_i = \mathbf{x}_i - \boldsymbol{\mu}_x$

### 4.1.3 Calculate Eigenvalues and Eigenvectors

The eigenvalues,  $\lambda_i$ , and eigenvectors,  $\mathbf{e}_i$ , are calculated by solving equation 4.4 which is known as the eigenvalue problem. The eigenvectors span an orthonormal space where the input data will be mapped into.

$$(\mathbf{C}_x - \lambda\mathbf{I}) \cdot \mathbf{e}_i = \mathbf{0} \quad (4.4)$$

### 4.1.4 Chose Eigenvectors

When the eigenvalue problem has been solved it must be determined which eigenvectors to use. The choice depends on the application. In chapter 3 some of the methods which can be used to decide which eigenvectors to use are described. These are the m-method, the J-measure, and the SEPCOR-algorithm.

### 4.1.5 Map the Data

The mapping of the data is done using equation 4.5.

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \boldsymbol{\mu}_x) \quad (4.5)$$

where  $\mathbf{y}$  denotes the transformed data,  $\mathbf{A}$  denotes the transformation matrix containing the chosen eigenvectors as row vectors,  $\mathbf{x}$  denotes the input data, and  $\boldsymbol{\mu}_x$  denotes the mean of the input data.

# Bibliography

- [1] J. Edwards Jackson. *A Users Guide to Principal Components*. Wiley Series in Probability and Mathematical Statistics, 1991.
- [2] Kittler J.P. and Devijver P. Selected topics in statistical pattern recognition. Unknown Publisher, 1978.