ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Generalising Invariant Coordinate Selection to a non-linear dimensionality reduction method

Master Thesis Econometrics and Management Science
Business Analytics & Quantitative Marketing

Author: Christopher Claassen - 456177

Supervisor: dr. Aurore Archimbaud

Second Assessor: dr. Jeffrey Durieux

April 30, 2023

Abstract

Invariant Coordinate Selection (ICS) is a recent unsupervised multivariate method that is able to reveal the structure of high-dimensional data in a low-dimensional space. This is achieved by performing the simultaneous diagonalisation of a pair of scatter matrices. In this paper, we investigate how various kernel techniques can be used to capture non-linear interactions. We find that both kernel smoothing and reproducing kernels can be used to generate new pairs. Some numerical issues of using kernels are also addressed which is useful for practical usage of the method. We show that ICS in combination with kernels compares favorably to previous methods for the tasks of data visualisation, anomaly detection and classification. All results can be easily reproduced via a fresh implementation of ICS in Julia.

Keywords: Dimensionality Reduction · Simultaneous Diagonalisation · Kernel Methods · Data Visualisation · Anomaly Detection · Classification

The content of this thesis is the sole responsibility of the author and does not reflect the view of the supervisor, second assessor, Erasmus School of Economics or Erasmus University.

Contents

1 Introduction

The rate at which new information becomes available to us is ever-growing. Dimensionality reduction is therefore becoming more important than ever for reducing computation times and avoiding the so called curse of dimensionality. This latter phenomenon entails that it is often difficult to intuitively understand all interactions that can occur in a high-dimensional space. Reducing the amount of dimensions is usually done as a preprocessing step for a subsequent task such as classification or clustering. A simple form of dimension reduction is data compression which amounts to discarding dimensions at the cost of a loss of information. A more advanced type of dimension reduction is achieved by constructing new dimensions in which a task can be performed more easily than originally possible. The latter type of dimensionality reduction has an obvious advantage over the former, but it not always possible to be obtained as easily.

The core issue in dimensionality reduction is that a lower dimensional representation cannot always retain all types of interactions of the higher dimensional original. There is usually a trade-off between different types of structure because of this. A good example of this is the trade-off between local and global structure. The local structure represents the relation of an observation to neighbouring observations, whereas the global structure represents the relation of an observation to all other observations. The type of structure that is to be retained is commonly determined by the subsequent statistical task. For example, data visualisation primarily requires faithful local structure, whereas something like regression requires more accurate global structure. The type of structure to be retained is not always as straightforward to diagnose. For example, classification requires the identification of class membership above all and the form of this structure can depend on the type of classifier used.

Recently, Tyler et al. (2009) introduced a method called *Invariant Coordinate Selection* (ICS) which is able to show various different types of structure in a lower dimensional subspace. ICS detects specific types of structure in a low-dimensional space via the simultaneous diagonalisation of two scatter matrices. These scatter matrices generalise the concept of the covariance matrix. The method can therefore be seen as an extension of the widely used *Principle Components Analysis* (PCA; Hotelling, 1933), which performs the diagonalisation of a single scatter matrix. This scatter matrix is usually taken as the regular covariance matrix. PCA essentially exploits the covariance structure in order to find a number of orthogonal components. The components constructed by PCA each represent a certain fraction of the original variance. Subsequent dimensionality reduction is then achieved by discarding components that represent little to no variance. ICS is similar to PCA, but it is different in that not a single covariance structure is of importance, but that the informational contrast between two different scatters is. The type of revealed structure is determined by the choice of scatter matrices. Put differently, ICS uses two matrices that have some form of informational contrast to generate a subspace in which the structure of that certain type of contrast is identifiable. ICS has been successfully applied in different contexts such as blind source separation (Nordhausen and Virta, 2019), clustering (Alfons et al., 2022) and outlier detection (Archimbaud et al., 2018).

A potential drawback of using ICS is that dimensionality reduction via this method will only capture the linear structure of the data. This is unfortunate because including the non-linear structure can often improve the performance of subsequent statistical applications. The main objective of this paper is therefore to investigate how different techniques can be incorporated into the ICS framework such that it captures non-linear structure. More specifically, we address how observational interactions via kernel techniques can be used to obtain suitable scatter matrices for ICS. This is in contrast to obtaining scatter matrices via parametric interactions as done in previous works. We find that both the techniques of kernel smoothing and reproducing kernels can be used in the successful kernelisation of ICS via kernel induced matrix pairs. This is verified by empirical application via the tasks of data visualisation, anomaly detection and classification. The wide selection of applications indicates the potential of these kernel methods. Providing theoretical derivations based on the kernel methods is challenging because the techniques are build on concepts of functional analysis. The contributions of this paper are therefore primarily on the applied side, where the theoretical underpinnings are left for future research. A fresh numerical implementation of the discussed methods is also provided in Julia (Bezanson et al., 2017) as part of this thesis. The code of this package with accompanying documentation can be accessed at: `https://github.com/CClaassen/SimultaneousDiagonalisation.jl`.

The remainder of this paper is structured as follows. We start in Section 2 by giving an overview of dimensionality reduction methods specified in the literature. Next, we take a closer look at the simultaneous diagonalisation framework and related methods in Section 3. Following that, Section 4 explores extensions for ICS in the form of kernel smoothing and reproducing kernels. Afterwards, Section 5 discusses a new Julia package that contains numerical implementations of the various methods. Subsequently, various empirical applications of kernel induced ICS such as data visualisation, anomaly detection and classification are discussed in Section 6. Section 7 concludes by giving an overview of the paper and discussing directions for future works.

2 Previous Works

One of the earliest and most used methods for dimension reduction is called *Principal Component Analysis* (PCA). It was independently discovered by Pearson (1901) and Hotelling (1933). The core idea of PCA is to introduce a change of basis such that the data becomes orthogonal under certain restrictions to the variance. These restrictions to variance are that the variance contained in the first component is maximal, whilst also iteratively maximising the share of the remaining variance of subsequent components. This procedure results in the set of principle components which have the same dimensionality as the original data. Put differently, the principle components represent linear combinations that are an exact reconstruction of the original data. Plain PCA is therefore not actually a strict dimensionality reduction method, but it can be made one by discarding components. A straightforward way to do so is by discarding the last few components that represent little to no variance. This approach makes sense if we are interested in a dimension reduction that retains as much variance as possible. However, retaining a certain amount of variance is not always sufficient for numerous applications. For example, Chang (1983) and De Soete and Carroll (1994) show that different clusters are not necessarily distinguishable by retaining the principal components with the largest variance. It is therefore useful to consider other methods that are able to retain different types of structure.

Tyler et al. (2009) provide a more recent method called *Invariant Coordinate Selection* (ICS) which is able to find different types of structure. The authors derive various properties of the method. One of these properties determines the name, as the resulting components can be affine invariant depending on whether the pair of scatter matrices used are affine equivariant. In that case, changes to the data like rotation, scaling and translation do not affect the components. ICS is able to find different types of structure in low dimensional representation which makes it a suitable dimensionality reduction method. It is able to find this interesting structure by exploiting the difference in information that two scatter matrices provide. These scatter matrices generalise the concept of the regular covariance matrix. The type of structure that is found via ICS is determined by the contrast between the scatter matrices used. It is therefore implied that the best scatter pairings are application dependent. Many different types of structure can be found with ICS. This can be seen by the wide variety of possible applications of the methods such as blind source separation (Nordhausen and Virta, 2019), clustering (Alfons et al., 2022) and outlier detection (Archimbaud et al., 2018). A good overview of different useful scatter pairs is given by Nordhausen and Ruiz-Gazen (2022).

A shortcoming of both PCA and ICS is that they are unable to deal with non-linear structure in the data directly. A potential remedy to this is adding additional data based on non-linear transformation of the original data, but this is computationally expensive in practice. Moreover, adding new data seems counter-intuitive to the objective of dimensionality reduction. Fortunately, the literature provides many examples of more efficient methods that can turn linear methods into non-linear ones. A good example of this is kernel PCA (Schölkopf et al., 1998), which is a non-linear extension of PCA. Kernel PCA uses the so called kernel trick, which replaces the scatter used in the standard version with a kernel matrix. This kernel matrix is able to reproduce the effect of extending the data in various ways. A remarkable result is that kernels can be used to reproduce the effects of infinitely large non-linear datasets whilst only requiring a finite amount of computation time. Other examples of the successful kernelisations of linear methods are discussed in Boser et al. (1992) and Bach and Jordan (2002). Using kernels to extend ICS to a non-linear method is not yet well-studied, but it will be the focus of this paper.

There also exist non-linear dimension reduction methods that reduce the dimensionality of the data based on different principles from covariances and kernels. *T-distributed Stochastic Neighbourhood Embedding* (t-SNE; Van der Maaten and Hinton, 2008) and *Uniform Manifold Approximation & Projection* (UMAP; McInnes et al., 2018) are recent non-linear dimension reduction methods that primarily aim to retain local structure via pairwise distances. They are collectively referred to as manifold learning methods because they both attempt to find a low-dimensional embedding that faithfully represents the close interactions of the original data. T-SNE achieves this by making use of probability distributions, whereas UMAP relies on graph theory. The biggest difference in comparison to PCA and ICS is that manifold learning methods work directly with the desired amount of dimension rather than discarding undesired components. Another difference is that these methods rely on iterative optimisation of loss functions which usually becomes quite computationally expensive in practice. T-SNE and UMAP are both mostly used as general data visualisation tools because of this. A potential application of the computationally cheaper PCA and/or ICS comes in the form of the initialisation step that both these manifold learning methods use. Kobak and Linderman (2021) show that a solid initialisation step is fundamental for a satisfactory performance of these methods.

Another type of non-linear dimensionality reduction methods comes from the field of machine learning. The most widely used class of machine learning dimension reduction methods are referred to as autoencoders (Kramer, 1991). Autoencoders are feedforward neural networks with a specific structure. The structure consists of two parts: an encoder and a decoder, with the smallest hidden layer in between them. The smallest layer is called the bottleneck and is vital in the sense that it allows the network to learn a sparse representation of the input data. The authors show that using an autoencoder with linear activation functions and a single hidden layer leads to a result strongly resembling the PCA solution. More complex non-linear structures can also be retained by changing the activation functions and layer structure of the autoencoder. In this sense, the use of non-linear activation function in an autoencoder leads to a solution that can be related to some form of non-linear PCA (Wang et al., 2014). This remarkable association is promising, but the exact relations are unfortunately not yet well-studied. We will therefore not focus on the relation between autoencoders and our newly proposed methods. There are also some drawbacks to using autoencoders. Examples of this are the increased computational burden and the increased likelihood of overfitting.

3 The Simultaneous Diagonalisation Framework

We continue with the methodology section that gives a basic overview of different ways to perform dimensionality reduction and subsequent visualisation or classification. We start by discussing the fundamentals of both matrix diagonalisations and scatter matrices that form the building blocks for the simultaneous diagonalisation of two scatters matrices. Next, we consider two alternative dimensionality reduction methods that are primarily used for data visualisation. The section is concluded by taking a look at two different types of classifiers and their evaluation methods, for which we use the dimension reduction as the preprocessing step.

3.1 Linear dimension reduction via PCA

Pearson (1901) and Hotelling (1933) introduced one of the earliest methods for dimensionality reduction in the form of *Principal Component Analysis* (PCA). The objective of PCA is to find an orthogonal linear transformation of a dataset X that concentrates the most important variational information in a lower dimensional space. An actual reduction in the amount of dimensions can subsequently be achieved by discarding components that jointly represent only a small proportion of the total variance. The original data matrix $X_{n\times p}$ consists of n observation vectors denoted by x_i, or likewise of p parameter vectors denoted by x_j. This means that there are a total of $n \times p$ scalar elements in X denoted by x_{ij}. The result of dimension reduction via PCA is then defined by the matrix $Y_{n\times r}$ such that $r < p$ holds after discarding some of the principle components.

The computation of principle components of the centered data matrix X can be done via various matrix decompositions. A solid treatment of different matrix decompositions and their relations is given in Golub and Van Loan (2013). For PCA, we can use the eigendecomposition of the sample covariance matrix $C = \frac{1}{n-1}X'X$:

$$C = P\Lambda P^{-1} = P\Lambda P',$$

where P is an orthonormal matrix with columns containing eigenvectors and Λ is a diagonal matrix consisting of ordered eigenvalues λ_i. The eigendecomposition is only defined for certain square matrices. However, the fact that the covariance matrix is symmetric and *positive semi-definite* (PSD) by construction implies that the eigendecomposition is always possible with real non-negative eigenvalues. Equivalently, the diagonalisation of the covariance matrix can be obtained by rewriting the factorisation as:

$$P'CP = \Lambda.$$

The principle components Y are subsequently obtained by multiplying the data with the eigenvectors such that $Y = XP$. This effectively means that Y represents a reflected and/or rotated version of the original centered data, where the proportion of original variance covered by a component is given by its corresponding eigenvalue. We can discard the components that correspond to the smallest eigenvalues for the dimensionality reduction. This is done because these components only represent a small amount of variance.

A drawback of the eigendecomposition is that numerical inaccuracies can force the eigenvalues to become negative or even complex. This issue can be addressed by instead considering the *Singular Value Decomposition* (SVD) of X given by:

$$X = USV',$$

where U and V are orthonormal matrices with columns consisting of left and right singular vectors respectively and S is a diagonal matrix containing the ordered non-negative singular values s_i. The SVD always exists for any matrix. The relation to the eigendecomposition can be obtained by looking at the SVDs of

$$X'X = VS^2V',$$
$$XX' = US^2U'.$$

$X'X$ and XX' do not have the same dimensionality unless X is square, but they do share the same non-zero singular values regardless. Only taking those non-zero singular values and corresponding singular vectors gives the thin factorisation of X. For the thin SVD factorisation it holds that $P = U_{thin} = V_{thin}$ and $\Lambda = \frac{1}{n-1}S^2_{thin}$. Thus the principle components can be calculated via the (thin) SVD as well. The advantage of the SVD is that it can form the eigendecomposition in a more numerically stable way by not explicitly calculating the covariance matrix.

The various methods discussed in this section will all be illustrated by making use of the well-known Iris dataset (Fisher, 1936)[1]. It is a structurally simple dataset that can be used to highlight how dimensionality reduction methods differ. The Iris dataset contains 150 flowers that are equally divided over three types of species. The data consists of four measurements that are recorded for every flower, such as the length and width of its petals for example. The result of applying PCA and retaining the first two components on the Iris data is shown in Figure 1 below. The figure shows the difference between the three species fairly well, but the versicolor and virginica species appear to have some overlap. This is not completely unexpected because the main objective of PCA has to do with the structure of the covariance rather than any class label that is not explicitly included in the data. This makes PCA an unsupervised method, because the class labels are not included in the computation.

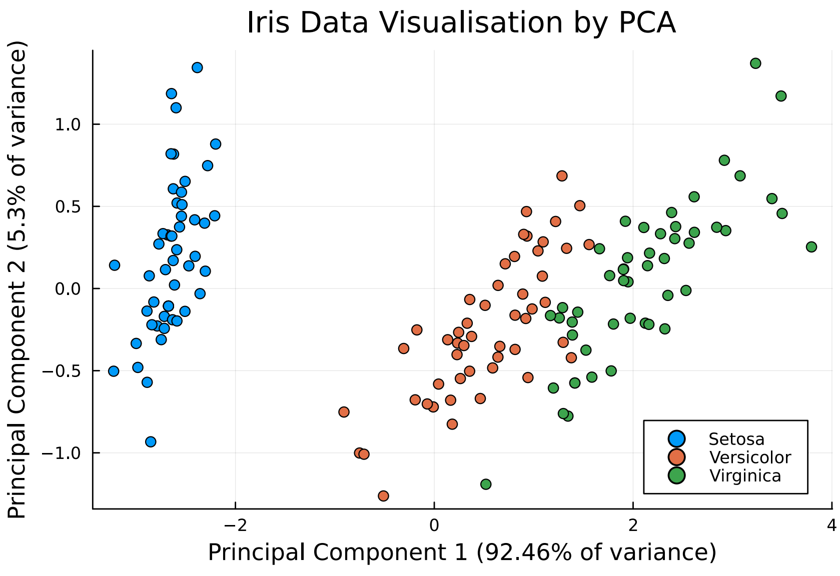


Figure 1: A two-dimensional visualisation of the Iris dataset via PCA.

3.2 From covariance matrices to scatter matrices

The original formulation of PCA uses the covariance matrix, but it is also possible to diagonalise a different matrix in its place. The result of this replacement is that we capture a different type of structure than the regular covariance provides. An example of this is the diagonalisation of the strongly related correlation matrix that results in cleansing the effect of different scales. Other matrices that can be used as a replacement also all share some properties with the covariance matrix. This generalisation of the covariance matrix is called a scatter matrix (Nordhausen and Ruiz-Gazen, 2022). It can be seen as a pseudo-covariance because all scatter matrices consist of symmetric PSD matrices. If we see the scatter as a scatter functional $V(F_X)$, it is sometimes also required to have the affine equivariance property based on the strictness of the definition used. The affine equivariance property is defined as:

$$V(F_{AX+b}) = AV(F_X)A',$$

[1]The Iris dataset can be accessed via: `https://archive.ics.uci.edu/ml/datasets/iris`

where A is a full-rank square matrix and b represents a vector. Using PCA with these scatter matrices requires a different type of centering. This can be achieved in a similar way via a corresponding generalisation of a mean to an affine equivariant location functional T for which it holds that:

$$T(F_{AX+B}) = AT(F_X) + b.$$

Many different scatter matrices can be defined via various statistical principles. Scatters that are obtained from the same statistical principle often capture the same type of information. For example, scatters that capture information of the shape of the underlying distribution of the data can be derived from statistical moments. The most straightforward one based on second moments is the standard covariance defined as:

$$\mathrm{Cov}_2(X) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}_n)(x_i - \bar{x}_n)',$$

where n is the amount of observations and $\bar{x}_n$ is a vector consisting of elements given by the sample mean: $\frac{1}{n}\sum_{i=1}^{n} x_i$. It is also possible to use higher order moments, albeit with some tweaking. It turns out that the so called matrix of fourth moments is also a scatter and it is defined by:

$$\mathrm{Cov}_4(X) = \frac{1}{n}\sum_{i=1}^{n}((x_i - \bar{x}_n)'\mathrm{Cov}_2(X)^{-1}(x_i - \bar{x}_n))(x_i - \bar{x}_n)(x_i - \bar{x}_n)'.$$

A different principle to obtain scatters is that of robustness. The aim of robust methods is to make valid inferences under potential contamination of the data. Two popular related robust scatters are introduced by Rousseeuw (1985). They are called the *Minimum Covariance Determinant* (MCD) and the *Minimum Volume Ellipsoid* (MVE). These scatters operate by finding a subset of observations that minimise a certain criteria. The MCD searches for a subset of proportion α with a minimal covariance determinant, whereas the MVE searches for a subset of proportion β that yields the minimum volume ellipsoid. The parameters α and β act as the desired asymptotic breakdown values, which represent the maximum proportions of observations that may be arbitrarily corrupted before the methods become inconsistent. These values are both usually not taken below $\frac{1}{2}$ out of a robustness point of view. The formulas for the MCD_α and the MVE_β are given by:

$$\mathrm{MCD}_\alpha(X) = \frac{c_\alpha}{n_\alpha}\sum_{j=1}^{n_\alpha}(x_{i_j} - \bar{x}_{\alpha,n})(x_{i_j} - \bar{x}_{\alpha,n})',$$

$$\mathrm{MVE}_\beta(X) = \frac{c_\beta}{n_\beta}\sum_{j=1}^{n_\beta}(x_{i_j} - \bar{x}_{\beta,n})(x_{i_j} - \bar{x}_{\beta,n})',$$

where c_α and c_β are Fisher consistency factors and $\bar{x}_{\alpha,n}$ and $\bar{x}_{\beta,n}$ are the sample means corresponding to the selected observations. Some additional efficiency can be recovered by extending the optimal subset such that additional 'inner' observation are included. This gives the 1-step reweighted versions of the methods denoted as RMCD_α and RMVE_β respectively. We will use these variants for the remainder of this paper. Determining the subsets that yields minimal objective criteria can be done via smart probabilistic enumeration (Rousseeuw and Driessen, 1999) or more recent deterministic methods (Hubert et al., 2012).

3.3 Using scatter pairs for dimension reduction via ICS

It is interesting to look at how we can use the newly obtained scatter matrices in dimensionality reduction. The most straightforward approach is to replace the regular covariance used in PCA with a single different scatter. However, it is perhaps even better to consider the information of multiple scatters simultaneously. For that purpose, we first recall that PCA makes use of a single scatter matrix S_1 in the standard eigenvalue problem:

$$S_1 v = \lambda v,$$

where v and λ represent an eigenvector and eigenvalue respectively. Using a pair of scatters instead, say S_1 and S_2, leads to the generalised eigenvalue problem:

$$S_1 v = \kappa S_2 v,$$

where v and κ now represent the generalised eigenvector and generalised eigenvalue respectively. The idea to use the generalised eigenvalue problem for statistical analysis was first introduced by Caussinus and Ruiz (1990) as generalised PCA. However, the generalised eigenvalue problem can also refer to other generalisations of the eigenvalue problem. This is partially why it was renamed in Tyler et al. (2009) to *Invariant Coordinate Selection* (ICS). This name is due to the property that the resulting components of ICS are invariant if a pair of affine equivariant scatters is used. ICS becomes a dimensionality reduction method by first solving the generalised eigenvalue problem and subsequently extracting only the most important components.

The generalised eigenvalue problem of the pair $\{S_1, S_2\}$ is the same as the standard eigenvalue problem of $S_2^{-1} S_1$, given that S_2 is invertible. However, this inversion is almost never done in practice due to numerical reasons. We therefore turn to the generalised eigendecomposition of two matrices. Rewriting this generalised decomposition to diagonalisation form gives:

$$W' S_1 W = \Lambda_1$$
$$W' S_2 W = \Lambda_2$$

where W has generalised eigenvectors as its columns and Λ_1 and Λ_2 are both diagonal matrices. The fact that S_1 and S_2 are both scatters ensures that the main entries of the diagonal matrices are real and non-negative. The simultaneous diagonalisation is often simplified such that one of the diagonal matrices is equal to identity. This simplification gives the system:

$$P' S_1 P = \Lambda$$
$$P' S_2 P = \mathrm{I}_p$$

where Λ is a diagonal matrix consisting of generalised eigenvalues, P has generalised eigenvectors as its columns and I_p denotes the identity matrix with p columns and rows. This simplification can essentially be seen as diagonalising S_1 after a previous whitening step via S_2. The choice of which diagonal matrix to simplify influences the eigenvalue ordering given that both matrices are full rank. This is due to the fact that if λ_1 is the first generalised eigenvalue of the pair $\{S_1, S_2\}$, then $\frac{1}{\lambda_1}$ is the last generalised eigenvalue of the the pair $\{S_2, S_1\}$. The simultaneous diagonalisation can be rewritten to a simpler form as $S_1 = S_2 P \Lambda P^{-1}$. The generalised eigenvalues Λ of a scatter matrix pair can be seen as generalised relative kurtosis measures (Tyler et al., 2009). An implication of this is that the components corresponding to the minimal and maximal generalised eigenvalues represent the largest departures from normality.

It is important to note that the matrix P is not always orthonormal like in the standard eigenproblem. This means that $P^{-1} = P'$ does not hold in general. To see this, we can use the QR decomposition to obtain $P = QR$. Subsequently, we can rewrite $S_1 = S_2 P \Lambda P^{-1}$ to

$$\begin{aligned} S_1 &= S_2 Q R \Lambda R^{-1} Q' \\ &= S_2 Q \Lambda R^* Q', \end{aligned}$$

where the latter equality is just a simplification as $R \Lambda R^{-1}$ is an upper triangular matrix with the original generalised eigenvalues on the diagonal. This reveals that the simulaneous diagonalisation of two scatter matrices is in fact a simultaneous (upper) triangularisation if we limit P to being orthonormal, unless R is diagonal. It shows the strong relation to the general simultaneous triangularisation that can be achieved via the generalised Schur decomposition. This decomposition can be rewritten in triangularsation form as

$$\begin{aligned} Q' S_1 Z &= T_1 = \Lambda_1 R_1 \\ Q' S_2 Z &= T_2 = \Lambda_2 R_2 \end{aligned}$$

where Q and Z are orthonormal matrices, R_1 and R_2 are upper triangular matrices and $\Lambda = \Lambda_1 \Lambda_2^{-1}$ is equal to the generalised eigenvalues. The simplified Schur form is given by $S_1 = S_2 Z T_1 T_2^{-1} Z'$. The generalised Schur decomposition is also known as the QZ decomposition. It is used as an intermediate step in the numerical solution of the generalised eigenvalue problem.

The fact that P is not necessarily orthonormal also implies that the solution to the generalised eigenproblem cannot be directly retrieved from the *Generalised Singular Value Decomposition* (GSVD). To see this, we can write the GSVD in triangularisation form as

$$\begin{aligned} U' S_1 Q &= D_1 R \\ V' S_2 Q &= D_2 R \end{aligned}$$

where U, V and Q are orthonormal matrices, R is an upper triangular matrix and $D = D_1 D_2^{-1}$ represents the generalised singular values. The simplified form on the GSVD is given by $S_1 = S_2 V D U'$ in case of two square matrices. This is not quite the same form as the generalised eigenvalue problem, because U is not the same as V in general. Nevertheless, we can obtain the generalised eigenvalues and eigenvectors via an additional spectral decomposition of VDU'. This is possible because $P \Lambda P^{-1} = VDU'$ holds. As such, the GSVD is not necessarily better as was the case with the regular eigenvalue problem. However, the GSVD does provide a benefit if either S_1 or S_2 are not full-rank and/or numerically instable. We will return to this benefit in the kernel extensions section.

Turning ICS into a proper dimensionality reduction method requires component selection. A general method for selecting components resulting from any kind of pair is not yet available in the literature. This is partially because analytical results are difficult to obtain for scatter pairings that do not rely on statistical moments. Another factor is that different applications require different kinds of structure in the components. Most methods are therefore based on the eigenvalues of the simultaneous diagonalisation. It is also possible to do component selection based on the components themselves, see Archimbaud et al. (2018) for an example. A general result is that the interesting components can be found along (a combination of) the first few and/or last few components. This is because these components correspond to maximal and/or minimal generalised relative kurtosis values. We will use this method for components selection in the remainder of the paper. An example of ICS with the first two components of the $\{\text{Cov}_4,\text{Cov}_2\}$ pair on the Iris dataset is given in Figure 2 below. The figure shows that the different species are not perfectly separated, which implies that we should try another scatter pair. The invariance property is highlighted by the fact that the dispersion is mainly along the axes.

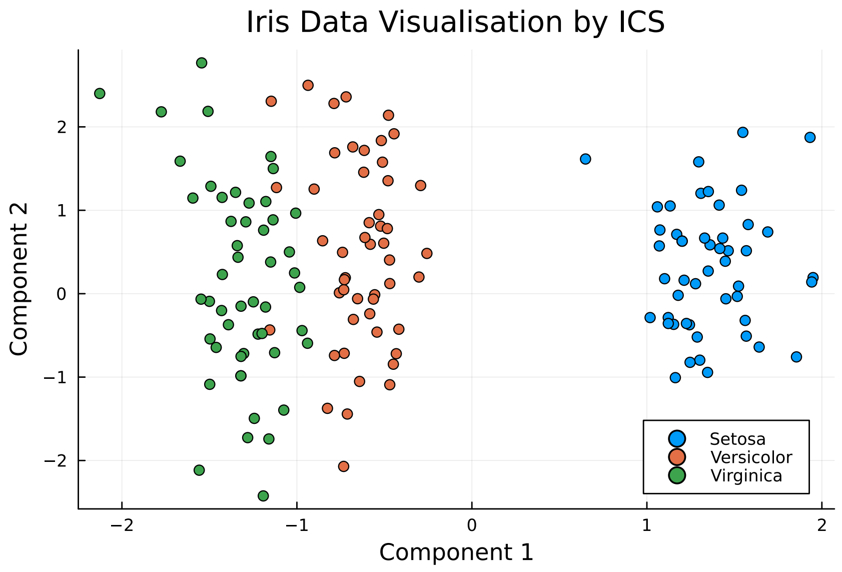


Figure 2: A two-dimensional visualisation of the Iris dataset via ICS with $\{\text{Cov}_4,\text{Cov}_2\}$.

3.4 Non-linear dimension reduction alternatives: t-SNE and UMAP

T-distributed Stochastic Neighbourhood Embedding (t-SNE; Van der Maaten and Hinton, 2008) and *Uniform Manifold Approximation & Projection* (UMAP; McInnes et al., 2018) offer non-linear dimensionality reduction alternatives to ICS. The biggest advantage of using either t-SNE or UMAP over ICS is that they can capture local structure exceptionally well, but there are also quite a few drawbacks. The biggest drawback is arguably that the retention of global structure is often poor. Both methods are primarily used for data visualisation. T-SNE is arguably the simpler method to explain, as it attempts to fit high dimensional datapoints in a low-dimensional embedding by making similar observations attract each other and letting opposites repel. This is achieved by first defining a (different) distribution for the low- and high-dimensional space, calculating the implied pairwise probabilities of being sufficiently close and minimising the divergence between those probabilities. The core steps can thus be concisely summarised in four formulas as follows:

$$\begin{aligned}
p_{j|i} &= \frac{\exp\left(-\frac{||x_i-x_j||^2}{2\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{||x_i-x_k||^2}{2\sigma_i^2}\right)},\\
p_{ij} &= \frac{p_{j|i}+p_{i|j}}{2},\\
q_{ij} &= \frac{\left(1+||y_i-y_j||^2\right)^{-1}}{\sum_{k\neq l}\left(1+||y_k-y_l||^2\right)^{-1}},\\
\text{min.}\quad \mathrm{KL}(P\,||\,Q) &= \sum_{ij} p_{ij}\log\frac{p_{ij}}{q_{ij}},
\end{aligned}$$

where P represents the the (Gaussian) probability matrix of similarity between between the original observations in X, Q represents the the (Student-t) probability matrix of similarity between between the low-dimensional embeddings Y, and KL represents an abbreviation of the Kullback-Leibler divergence measure from Kullback and Leibler (1951).

Good performance of the t-SNE method primarily depends on a good initialisation of the low-dimensional embedding space in order to retain some amount of global structure in addition to good hyperparameter tuning. PCA is often used for the initial preprocessing step and the subsequent initialisation. This is done in order to decrease the computational cost and help with avoiding poor local minima of the loss function. It is interesting to mention here that ICS can potentially be used as an alternative preprocessing step for t-SNE such that it visualises different aspects of the underlying data. The most important hyperparameter of t-SNE is called perplexity. It can effectively be seen as the amount of nearest neighbours because it determines the individual Gaussian bandwidths σ_i though maximising the Shannon entropy. T-SNE is only feasible if the amount of data points and/or embedded dimensions are low due to the high computational cost of minimising the objective function. Other disadvantages of t-SNE include the aforementioned poor retention of global structure and the fact that new points cannot be inserted without rerunning the algorithm from scratch. An example of a t-SNE embedding for the Iris dataset is given in Figure 3 situated on the next page. T-SNE is able to give a good representation of the data in the sense that the different flower species are mostly given their own cluster. However, it can also be seen that the versicolor and virginica species do not strongly repel each other. The scales of the axes are somewhat arbitrary as they are scaled based on a combination of the amount of observations and the perplexity hyperparameter.

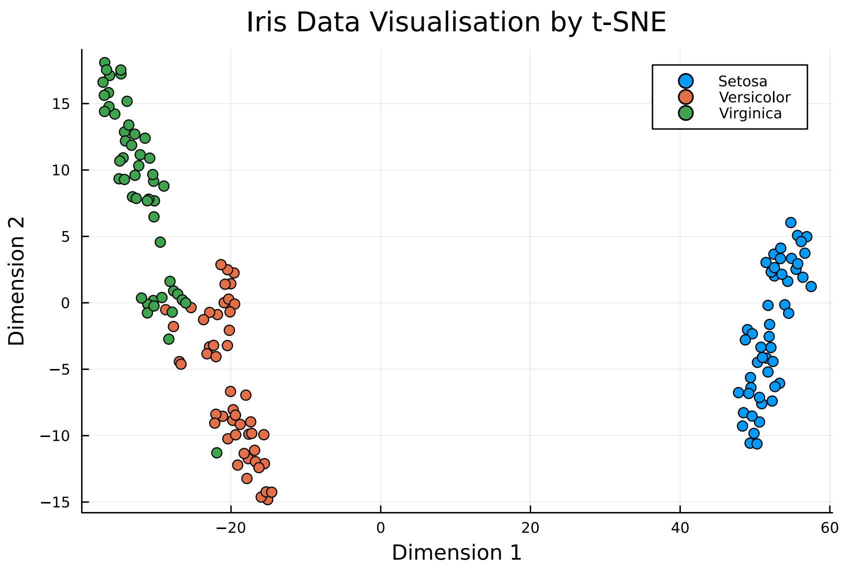


Figure 3: A two-dimensional visualisation of the Iris dataset via t-SNE.

UMAP is strongly related to the principles used in t-SNE, but it is instead based on graph theory. UMAP works similar to t-SNE in the sense that both methods can be seen as n-body simulation via attraction forces versus repulsion forces. T-SNE uses probabilities to determine those forces, whereas UMAP opts for weighted graphs. Intuitively, UMAP works by connecting all points in a weighted graph by basing those weights on the closeness of points in the high dimensional space. Larger weights correspond to observations being put closer together in the lower dimensional embedding. UMAP essentially follows the same principles as t-SNE with a similar objective. As such, it is sufficient to look at the differences in loss function and important hyperparameters. The UMAP cost function is defined as follows:

$$\sum_{e \in E} w_h(e) \log\left(\frac{w_h(e)}{w_l(e)}\right) + (1 - w_h(e)) \log\left(\frac{1 - w_h(e)}{1 - w_l(e)}\right),$$

where e is a single edge out of the set of all possible edges E in the complete weighted graph represented by the weights $w_i(e)$ in dimension i. The function defined above contains two primary parts. The first part of the sum is the attraction force and the second part is the repulsion force. UMAP's main hyperparameters are setting how many nearest neighbour should be connected with an edge in the graph and the minimal distance between points in the low-dimensional embedding. The latter hyperparameter is thus mostly used for achieving better looking visualisations in practice.

The biggest advantage of UMAP over t-SNE is that higher dimensional embeddings are more feasible with UMAP, but this is at the cost of inherently random initialisation and optimisation utilised in most implementations. The inability to provide a consistent structural initialisation also affects the reproducibility of the embedding, which is a drawback of this method. An issue from T-SNE that is not addressed by UMAP is the lack of global structure retention. An implication of this is that metric based clustering methods such as k-means work poorly even though both t-SNE and UMAP graphs show cluster-like output. This is also partially caused by the fact that the distance between possible clusters in the resulting embedding is not necessarily faithful. Figure 4 below shows an example of a UMAP embedding for the Iris dataset. The embedding is similar to the t-SNE visualisation, especially if we take into account that the distance between clusters is mostly arbitrary for both methods.

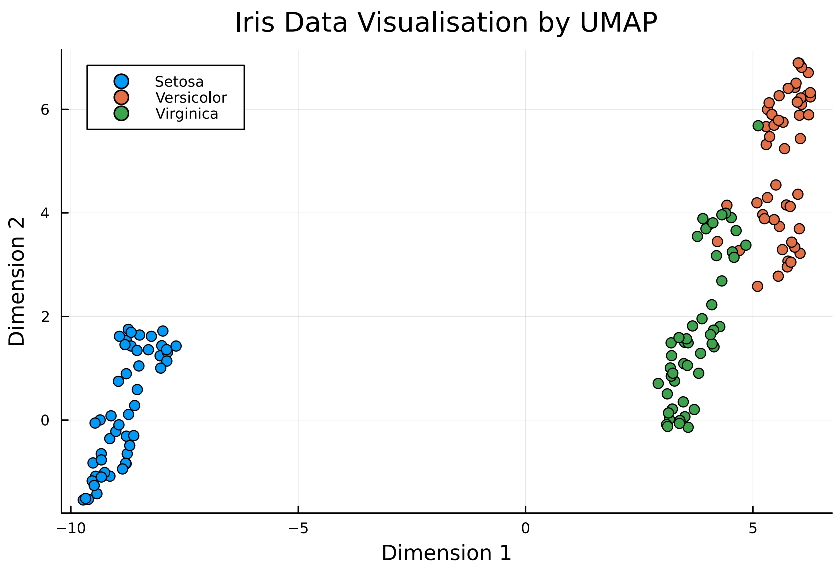


Figure 4: A two-dimensional visualisation of the Iris dataset via UMAP.

3.5 Classification, validation and evaluation metrics

The result of a dimensionality reduction method can be used for multiple subsequent statistical tasks. One such task is classification which tries to determine which of various classes an observation belongs to. The simplest form is a binary classifier that determines if an observation belongs to one group or the other. Not all binary classifiers have a straightforward generalisation to the multiclass case. An important difference in the case of more than two classes is the distinction whether or not these classes possess a natural ordering. There is also a difference between static classifiers that assign a single class per observation and probabilistic classifiers that provide membership probabilities for all classes. We will continue by discussing both a static and a probabilistic classifier.

A simple binary static classifier called *Support Vector Machines* (SVM) is introduced by Vapnik (1963). The idea behind SVM is to construct a hyperplane that maximises the margin between all observations x_i of two groups represented by labels $y_i \in \{-1, 1\}$. In other words, this hard-margin SVM finds a coefficient vector $\begin{bmatrix} w \\ b \end{bmatrix}$ such that the restriction

$$y_i(w'x_i - b) \geq 1$$

holds for all n observations. The above restrictions need to be relaxed in case the data is not perfectly linearly separable by a hyperplane. This gives the soft-margin SVM classifier with the optimisation criteria:

$$\text{min.} \quad \lambda||w||_2^2 + \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(w'x_i - b)),$$

where λ is an additional penalty parameter. The optimisation of the loss function is usually done via the dual problem in practice. The soft-margin SVM classifier is not scale invariant due to the hinge loss used in the loss function. A generalisation to multiclass SVM on labels $y_i \in \{1, 2, ..., K\}$ can be obtained in multiple ways. One way is to construct K binary SVM classifiers that each perform one versus rest classification of an individual class. These binary classifiers are not necessarily disjoint, but any overlapping classification regions can be resolved by majority voting.

A simple multiclass probabilistic classifier called *Multinomial Logistic Regression* (MLR) is introduced by Theil (1969). The MLR model works by modelling the K class membership probabilities of observations x_i as:

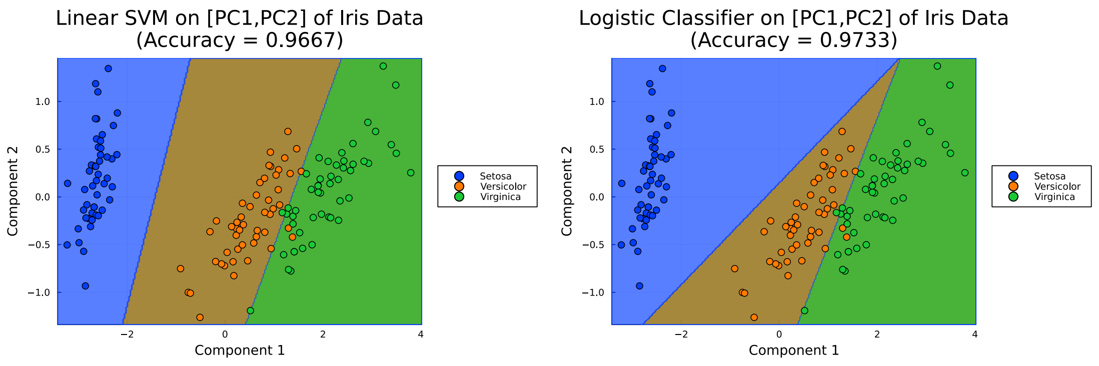
$$\Pr(Y_i = k|x_i) = \frac{\exp(x_i'\beta_k)}{\sum_{l=1}^{K}\exp(x_i'\beta_l)},$$

where all β_l represent class specific parameters. Class K is usually chosen as the base class by setting the corresponding β_K to 0. It does not matter which class is chosen as the base class, but picking a base class fixes the scale of the distribution. This allows the model to be fitted via maximum likelihood estimation of the log-likelihood function given by:

$$\sum_{i=1}^{N}\sum_{j=1}^{K} I[y_i = j]\Pr(Y_i = j|x_i).$$

Actual classification is done by assigning the classes for which the class membership probabilities are maximal. The resulting decision boundaries are linear just as with SVM, but they are not necessarily the same. The logistic classifier has an advantage over SVM by being scale invariant.

Figure 5: Example of the difference between the SVM and MLR classifiers on the Iris dataset.

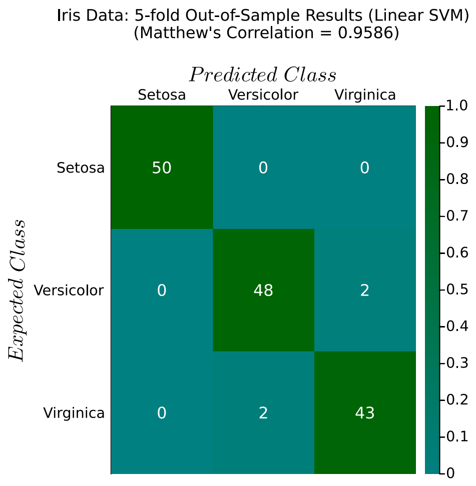


(a) Decision regions of the SVM classifier.

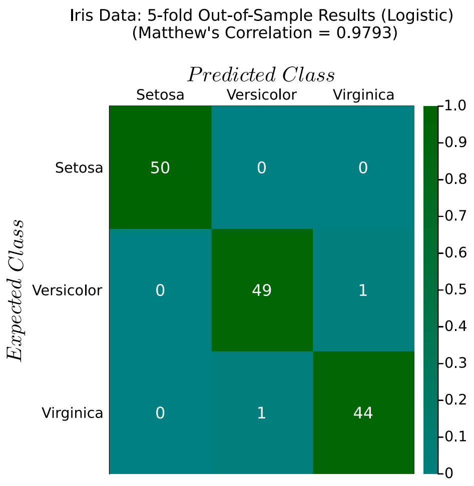
(b) Decision regions of the MLR classifier.

The panels in Figure 5 above give an example of the difference between SVM and MLR for the Iris dataset. We see that the decision regions of the classifiers are different and that neither classifier is able to separate the classes perfectly. A good assessment of the classification performance is obtained via k-fold cross-validation. It consists of randomly partitioning the data in k equal folds and evaluating individual folds by fitting classifiers to the remaining $k-1$ folds. An additional step of stratification can ensure that the class distribution of all folds are similar to the distribution of classes in the complete dataset. This is useful if the distribution of classes is strongly non-uniform. The panels in Figure 6 below show the 5-fold cross validation results of both the SVM and the MLR classifiers on the Iris dataset. Five samples of the viriginica species are discarded to create a small bit of class imbalance. The panels show that the 5-fold performance of MLR is somewhat better than that of SVM.

Figure 6: Example of the difference between the SVM and MLR classifiers on the Iris dataset.



(a) Confusion matrix of the SVM classifier.



(b) Confusion matrix of the MLR classifier.

The table of classification results is also known as the confusion matrix. Binary evaluation metrics are often given in terms of the entries from this confusion matrix. For the binary classification case, the confusion matrix and corresponding terminology look as follows:

(Expected \Predicted)	Positive	Negative
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

There are many ways of combining TP, TN, TP and FP into different evaluation metrics, see Powers (2011) for an overview. The best evaluation metric depends on the application. Some commonly used evaluation metrics are:

$$\begin{aligned}
\text{Accuracy (ACC)} &= \frac{TP + TN}{TP + TN + FP + FN}, \\
\text{Precision (PRC)} &= \frac{TP}{TP + FP}, \\
\text{Recall (REC)} &= \frac{TP}{TP + FN}, \\
\text{F-score } (\mathrm{F}_1) &= \frac{2}{\mathrm{PRC}^{-1} + \mathrm{REC}^{-1}} = \frac{2TP}{2TP + FP + FN}, \\
\text{Matthews Correlation (MCC)} &= \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}.
\end{aligned}$$

ACC, PRC, REC and F_1 can all take values between 0 (worst) and 1 (best). The MCC represents the correlation to a perfect binary classifier as it takes values between -1 (worst) and 1 (best). Most binary classification evaluation metrics cannot be easily generalised to the multiclass case because they depend on the presence of a single positive class. Table 1 below shows the results of the 5-fold SVM classifier on the Iris data with different positive classes, where T represents the total amount of observations. It can be seen that the evaluation metrics are dependent on the choice of positive class. It is therefore better to aggregate all individual results, of which the results are shown in Table 2 below. The 'macro' method refers to averaging all individual results, the 'weighted' method is the same as 'macro' but weighted by class proportions, and the 'micro' method refers to summing all individual results. Arguably the best scalar metric that is defined for multiple classes is the multiclass generalisation of the MCC. This MCC essentially represents the correlation to a perfect multiclass classifier and has output on [-1,1]. The formula given in Gorodkin (2004) for the multiclass MCC of a confusion matrix C with k classes is:

$$\text{MCC} = \frac{(\sum_{i=1}^k C_{ii})(\sum_{i=1}^k \sum_{j=1}^k C_{ij}) - \sum_{j=i}((\sum_{i=1}^k C_{ij})(\sum_{i=1}^k C_{ji}))}{\sqrt{(\sum_{i=1}^k C_{ii})^2 - (\sum_{j=i}(\sum_{i=1}^k C_{ij})^2)}\sqrt{(\sum_{i=1}^k C_{ii})^2 - (\sum_{j=i}(\sum_{i=1}^k C_{ji})^2)}}.$$

Table 1: Example of multiclass evaluation metrics for different classes as the positive class.

Positive Class	TP	TN	FP	FN	T	ACC	F_1	MCC
Setosa	50	95	0	0	145	100.00%	100.00%	100.00%
Versicolor	48	93	2	2	145	97.24%	96.00%	93.89%
Virginica	43	98	2	2	145	97.24%	95.56%	93.56%

Table 2: Example of multiclass evaluation metrics for different aggregation methods

Method	TP	TN	FP	FN	T	ACC	F_1	MCC
Macro	47.00	95.33	1.33	1.33	145.0	98.16%	97.19%	95.82%
Weighted	47.14	95.24	1.31	1.31	145.0	98.19%	97.24%	95.89%
Micro	141.00	286.00	4.00	4.00	435.0	98.16%	97.24%	95.86%

4 Kernel Method Extensions

We now turn our attention to (statistical) kernel techniques and how they can be used to provide additional interesting simultaneously diagonalisable matrix pairs. This is done in order to generate pairs that are able to capture non-linear structure. We first take a more formal look at scatter matrices. Afterwards, both a non-parametric extension in the form of kernel smoothing, and a spatial extension in the form of reproducing kernels are discussed. Potentially interesting kernel pairs for ICS are also considered, where some general differences and numerical issues concerning the simultaneous diagonalisation of these kernels pairs are addressed.

4.1 Scatters in terms of the inner product

In general terms, a kernel refers to something that contains the core part of some other object. In statistical terms, a kernel can be conceptually linked to the principle of scatters. However, kernels are usually defined in relation to the inner product, rather than in terms of statistical functionals. It is therefore important to first look at how we can obtain scatter matrices from inner product spaces in order to construct a basis on which we can build the kernel extensions.

To that end, we first look at forming a scatter matrix S out of a data matrix X that consists of n rows and p columns. This X is also assumed to be centered for simplicity. It is useful to realise that we can formally think of the data X as either the set $\{x_1, x_2, ..., x_n\}$ p-dimensional vectors that are elements of $\mathbb{R}^{\mathrm{p}}$ or the set $\{x_1, x_2, ..., x_p\}$ n-dimensional vectors that are elements of $\mathbb{R}^{\mathrm{n}}$. The symbols $\mathbb{R}^{\mathrm{p}}$ and $\mathbb{R}^{\mathrm{n}}$ denote finite-dimensional vector spaces with different cardinalities over $\mathbb{R}$. These approaches reflect interpreting the data matrix from either an observational or parametric perspective. The concept of a scatter matrix now naturally follows by considering either approach and providing the additional structure of an inner product. In case of the $\mathbb{R}^{\mathrm{n}}$ approach, the inner product is defined as any function $\langle \cdot, \cdot \rangle : \mathbb{R}^{\mathrm{n}} \times \mathbb{R}^{\mathrm{n}} \to \mathbb{R}$ that is:

$$\begin{aligned}
\text{Symmetric: } & \langle x_i, x_j \rangle = \langle x_j, x_i \rangle, \\
\text{Bilinear: } & \langle \alpha x_i + \beta x_j, x_k \rangle = \alpha \langle x_i, x_k \rangle + \beta \langle x_j, x_k \rangle, \\
\text{Postive semi-definite: } & \langle x_i, x_i \rangle \geq 0.
\end{aligned}$$

These requirements are similar to the requirement that scatters should be symmetric PSD matrices. In particular, defining the inner product as the scaled dot product $\langle x_i, x_j \rangle = \frac{1}{n-1} x_i' x_j$ fulfills these requirements and gives the standard covariance scatter $S = \Sigma = \frac{1}{n-1} X'X$. It has elements σ_{ij} that represent the inner products, or covariances, of vectors x_i and x_j. More formally, we say that the covariance matrix resides in the inner product space associated with the original vector space. The definition of a scatter matrix in this interpretation is thus any matrix that can be formed by associating a proper inner product function. The reverse is also true, in the sense that any scatter matrix of some data induces a particular inner product space of the corresponding vector space.

It is important to note that concepts resembling efficiency and robustness can be carried over. The standard covariance matrix is efficient in this framework in the sense that all possible dimensions are included and considered to be of equal importance. Put differently, we have not yet introduced any additional structure by which we could value the information of certain dimensions over others. Adding such structure can, for example, diminish the influence of aberrant dimensions. The previously discussed (R)MCD and (R)MVE estimators can be seen as examples of this. The additional structure of these methods comes in the form of providing a criteria for robustness and subsequently adjusting the inner product function accordingly. Recall that in the case of the (R)MCD, the robustness criteria is provided by the subset of h observations that yield the lowest determinant of the corresponding covariance matrix. Afterwards, the (R)MCD excludes all observations that are not within a certain robust distance from the center of those observations. This leads to a inner product function that is similar to the usual dot product as before, but with the exclusion of certain dimensions that do not fit the robustness criteria. Consequently, the (R)MCD does not induce an inner product space for the original vector space, but for its subspace that contains the subset of h innermost observations.

4.2 Kernel smoothing

Considering the robust scatters as inner products essentially leads to a form of weighted covariance matrices, even though the weights are obtained in a sophisticated manner. An alternative non-parametric weighting scheme for capturing non-linear structure can be obtained by looking at pairwise interactions in the data. We consider the $\mathbb{R}^{\mathrm{p}}$ approach and adjust the inner product function by looking at the distance between individual observations. A so called kernel smoothing function $k : \mathbb{R}^{\mathrm{n}} \rightarrow \mathbb{R}$ is then used to transform the pairwise distances into relative weights. This technique is usually referred to as kernel smoothing, but it was originally known as Parzen-Rosenblatt window estimation (Rosenblatt, 1956; Parzen, 1962). It can be seen as a form of weighted moving average, which means that the name of weighting kernels is also appropriate.

The weighting kernel function determines the type of non-linearity that can be captured. Any proper symmetric probability density function can be used as a weighting kernel function. Some commonly used examples of weighting kernel functions $k(u)$ that can be used for obtaining weights are defined as follows:

$$\begin{aligned}
&\text{Support within unit ball:}\\
\text{Uniform} : k(u) &= \frac{1}{2},\\
\text{Triangular} : k(u) &= 1 - |u|,\\
\text{Epanechnikov} : k(u) &= \frac{3}{4}(1 - u^2),\\
\text{Quartic} : k(u) &= \frac{15}{16}(1 - u^2)^2,\\
\text{Triweight} : k(u) &= \frac{35}{32}(1 - u^2)^3,\\
\text{Tricube} : k(u) &= \frac{70}{81}(1 - |u|^3)^3,\\
\text{Cosine} : k(u) &= \frac{\pi}{4}\cos(\frac{\pi}{2}u),\\
&\text{Infinite support:}\\
\text{Gaussian} : k(u) &= \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}u^2},\\
\text{Logistic} : k(u) &= \frac{1}{e^u + 2 + e^{-u}},\\
\text{Sigmoid} : k(u) &= \frac{2}{\pi}\frac{1}{e^u + e^{-u}},\\
\text{Silverman} : k(u) &= \frac{1}{2}e^{\frac{|u|}{\sqrt{(2}}}\sin(\frac{|u|}{\sqrt{(2)}} + \frac{\pi}{4}),
\end{aligned}$$

where u represents $d(x_i, x_j)$ that is the distance between observations x_i and x_j for which the weight is calculated. An important difference is that some kernels only assign weight to 'effective' nearest neighbours from within the bounded support, whereas others assign some positive weight to any other observation. A graphical overview of this distinction can be seen in Figure 7 on the next page. The figure only shows the right sides of the functions due to pairwise distances being non-negative. It can be seen that several kernel functions are similar to other kernel functions. They can roughly be divided into three groups: discontinuous kernels, bounded support kernels, and infinite support kernels. The differences between the provided weights within these groups are relatively small. Nevertheless, it should be noted that even small differences between functions on an individual scale can have a significant effect on the overall result.

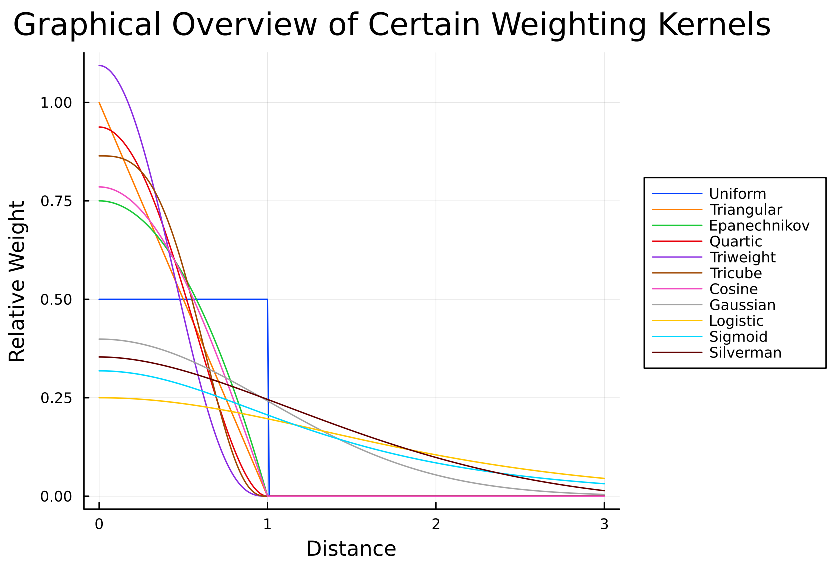


Figure 7: An overview of the relative weights given by certain kernel functions.

The region that contains the effective neighbours is usually adjusted by scaling the distance function by a non-negative scalar bandwidth parameter λ such that $u(x_i, x_j, \lambda) = \frac{1}{\lambda}d(x_i, x_j)$. The bandwidth effectively controls the efficiency-robustness trade-off because it scales the region from which potential neighbours are selected. The result is called a local covariance (Nordhausen et al., 2015) as it replaces every point with a normalised weighted linear combination of its (effective) neighbours. The local covariance matrix is subsequently calculated over those newly constructed points. The same linear combinations can also be used to obtain a location estimate. We will refer to the local covariance as 'LCov' from now on. In mathematical terms, the kernelised version of the LCov can be defined as

$$\begin{aligned}\mathrm{LCov}(X, \lambda) =& \frac{1}{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{k(\frac{1}{\lambda}d(x_i, x_j))}{\sum_{j=1}^{n} k(\frac{1}{\lambda}d(x_i, x_j))}(x_i - \bar{x}_n)(x_j - \bar{x}_n)' \\ =& \frac{1}{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n} k^*(\frac{1}{\lambda}d(x_i, x_j))(x_i - \bar{x}_n)(x_j - \bar{x}_n)',\end{aligned}$$

where k^* denotes that the relative weights given by the kernel function k are normalised by their sum. The corresponding inner product function is a weighted version of the usual dot product, where the normalised weights are determined by the kernel functions.

We can extend the kernelised LCov in two ways. The first way is to set individual λ_i such that they become inversely proportional to how densely the region surrounding observation i is populated. The effect of this asymmetry is that points without many close neighbours are then unilaterally forced towards farther neighbours. However, adjusting individual λ_i for already densely populated regions yields little effective change for reasonable computation cost of their distribution. We therefore look at an alternative to setting all λ_i individually.

We propose to calculate λ_i via an initial global λ with a subsequent individual refinement step. This is achieved by first calculating the pairwise distances for a global bandwidth λ, looking at the implied amount of effective neighbours for each point, and then scaling individual λ_i such that at least a proportion of $\alpha\%$ points is considered to be within unit distance of every point. This leads to a local covariance that uses λ for densely populated regions and guaranteeing at least $\lfloor \alpha n \rfloor$ effective neighbours for more sparsely populated regions. The individual scaling required can easily be obtained by partially sorting the implied pairwise distances of the initial λ. Using these newly constructed $\lambda_i^{(\alpha)}$ gives an adaptive local covariance, or ALCov, which follows a similar idea to the 1-step re-weighted robust covariance methods. The formula of ALCov_α is

$$\text{ALCov}_\alpha(X, \beta, \lambda) = \frac{1}{n-1} \sum_{i=1}^{n} \sum_{j=1}^{n} K^*(\frac{\beta}{\lambda_i^{(\alpha)}} d(x_i, x_j))(x_i - \bar{x}_n)(x_j - \bar{x}_n)',$$

where $\beta \in (0, 1]$ is an additional parameter to add some refinement to kernels that only offer support within the unit ball. The idea behind the β parameter is to add some differentiation in relative weight to points that are on the edge of the unit ball. It can be set equal to 1 for kernels that offer infinite support.

The second way to extend LCov is to increase the robustness by replacing the local weighted mean that constructs new points with a more robust version such as the weighted geometric median. The weighted geometric median y can be found via the following criteria:

$$\underset{y \in \mathbb{R}^p}{\arg\min} \sum_{i=1}^{p} ||w_i' x_i - y||_2,$$

where the relative weights w_i are again given by a kernel function. It corresponds to the point that minimises the weighted distance between all points. There is no closed form expression for y, but it can be found via the Weiszfeld algorithm (Weiszfeld, 1937) which is a form of iteratively reweighted least squares. The geometric median attains an asymptotic breakdown value of 50%, but it is not necessarily amongst the set of points over which it is calculated. A combination of adaptively scaling an individual λ and using a robust plug-in estimator gives the adaptive robust local covariance, or ARLCov_α, defined as:

$$\text{ARLCov}_\alpha(Y, \beta, \lambda) = \frac{1}{n-1} \sum_{i=1}^{n} \sum_{j=1}^{n} (y_i - \bar{y}_n)(y_j - \bar{y}_n)',$$

where Y represents the collection of weighted geometric medians. The α, β and λ parameters are implicitly included in this definition via the kernel function that determines the relative weights that are used for the weighted geometric median.

4.3 Reproducing kernels

The term statistical kernel is also used for various other types of techniques than the one described in the previous section. A particularly useful one is called the reproducing kernel and it is based on the theory of integral operators. Ghojogh et al. (2021) give a good overview of the theory behind reproducing kernels. Reproducing kernels are mainly used for extending methods that only capture linear information to the non-linear case. They have already been successfully used for kernelising several linear methods of which the most prominent methods are kernel PCA (Schölkopf et al., 1998) and kernel SVM (Boser et al., 1992). The reproducing kernel is used to generalise the inner product to allow for infinitely large sets of elements. For the construction of these reproducing kernels, we start with providing additional types of structure to the inner product space of the $\mathbb{R}^p$ approach. The first type of structure is that of a norm function $||\cdot|| : \mathbb{R}^p \to \mathbb{R}$ that has the following properties:

$$\begin{aligned} \text{Non-negativity: } & ||x_i|| \geq 0, \\ \text{Linearity: } & ||\alpha x_i|| = ||\alpha||||x_i||, \\ \text{Triangle inequality: } & ||x_i + x_j|| \leq ||x_i|| + ||x_j||. \end{aligned}$$

The most common norm function is the Euclidean norm $||x|| = \sqrt{x_i^2 + ... + x_n^2}$ as it represents the regular notion of distance. Defining the norm is essential in the sense that exchanging norm functions yields different reproducing kernels.

The other type of required structure is that the space must be complete with respect to its norm, which results in a complete metric space. This formally means that any Cauchy sequence of elements of the space cannot converge to something outside of these elements. Adding the requirement that the inner product induces the norm gives a Hilbert space $\mathcal{H}$. It is defined as an inner product space that is complete with respect to a norm which is induced by its inner product. The elements of Hilbert spaces are functions and features. A feature ϕ is a map of element-wise functional transforms of another vector, such as $\phi(x_i) = [1 \quad x_i \quad x_i^2]'$ for example. Hilbert spaces can therefore be seen as a generalisation of the standard vector space to a function space.

We can now use the dot product to define a corresponding inner product for features in this Hilbert space, which becomes $\langle x_i, x_j \rangle = \phi(x_i)'\phi(x_j)$. Something remarkable is that dimensionality of the mapping of feature $\phi(x_i)$ is allowed to be infinitely dimensional in $\mathcal{H}$. Using the inner product of infinitely many elements would be practically infeasible if not for the kernel trick. The kernel trick consists of replacing the inner product of feature maps in Hilbert space with the inner product of a certain kernel function. In mathematical terms, this can be represented as:

$$\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}} = \langle \kappa_{x_i}, \kappa_{x_j} \rangle_{\mathbb{R}^p} = \kappa(x_i, x_j),$$

where x_i and x_j are both observations of the original data. The Riesz representation theorem (Riesz, 1907) ensures that there always exists a unique kernel function κ that achieves this. This result is called the reproducing property because some inner product in functional Hilbert space is reproduced by another inner product in regular vector space. The main appeal of the kernel trick is that features do not need to be formed explicitly, as the same inner products can be obtained via kernel functions. A kernel that has the reproducing property induces the *Reproducing Kernel Hilbert Space* (RKHS), which is a subspace of $\mathcal{H}$. It is a subspace because it only spans the part of $\mathcal{H}$ that can be reproduced via (kernel) function evaluations. The reproducing kernels are therefore also known as RKHS kernels.

There are many different types of kernel functions possible that all reproduce different types of feature maps. Asserting which features are reproduced by a specific kernel can be done by using Mercer's conditions (Mercer, 1909). Some common examples of kernel functions $\kappa(x_i, x_j)$ that can be used are:

$$
\begin{aligned}
\text{Linear}: &\quad \kappa(x_i, x_j; c) = (x_i' x_j + c), \\
\text{Polynomial}: &\quad \kappa(x_i, x_j; c, d) = (x_i' x_j + c)^d, \\
\text{Laplacian}: &\quad \kappa(x_i, x_j; \sigma) = \exp(-\frac{||x_i - x_j||}{\sigma}), \\
\text{Gaussian}: &\quad \kappa(x_i, x_j; \sigma^2) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2}), \\
\text{Rational}: &\quad \kappa(x_i, x_j; \alpha, \gamma) = (1 + \frac{||x_i - x_j||^{2\gamma}}{\alpha}), \\
\text{Cosine}: &\quad \kappa(x_i, x_j) = \cos(\pi |x_i - x_j|), \\
\text{NeuralNet}: &\quad \kappa(x_i, x_j) = \arcsin(\frac{x_i' x_j}{\sqrt{(1 + x_i' x_i)(1 + x_j' x_j)}}), \\
\text{Mahalanobis}: &\quad \kappa(x_i, x_j; P) = \exp(-(x_i - x_j)' P (x_i - x_j)),
\end{aligned}
$$

where $\{c, d, \alpha, \gamma, \sigma, \sigma^2, P\}$ are all kernel specific parameters. The reproduced feature map of the polynomial kernel is a feature map of all powers and cross products up to order d. By contrast, the reproduced feature map of the Gaussian kernel consists of infinitely many power interactions. The regular covariance matrix is proportional to the Linear kernel in the sense that it gives proportional results. An interesting note is that new reproducing kernels can be constructed from the (infinite) sum and/or product of 'base' reproducing kernels. This is akin to the concept that more sophisticated function can be written as a Taylor series expansion. Exchanging the norm function also yields a new valid kernel as mentioned previously. This can be seen by comparing the definitions for the Gaussian and Laplacian kernels given above. These kernels are essentially the same outside of the norm used.

The relation between covariance matrices and reproducing kernels can be made more formal. Mercer's theorem (Mercer, 1909) states that any kernel matrix that has the reproducing property can exclusively be found among symmetric PSD matrices. The reverse is also true, as the Moore-Aronzajn theorem (Aronszajn, 1950) states that every symmetric PSD matrix induces some particular RKHS. As such, a reproducing kernel allows for the fast calculation of inner products between (in)finitely large features. The reproduced features consist of non-linear transformations of the original data. A subsequent (simultaneous) diagonalisation then provides the orthogonal projection of those non-linear features without ever forming them explicitly.

4.4 Kernel pairs and numerical considerations

The previously discussed kernel methods provide scatters that capture non-linear structure in the data. We now consider the practical differences between using ICS with either scatter or kernel matrices. For the remainder of this section, we treat the kernelised versions of the LCov as a scatter. This is because the most important difference for ICS is that scatter pairs provide up to p components, whereas reproducing kernel pairs provide up to n components. For data with fewer parameters than observations ($p < n$), this means that using reproducing kernels does not directly reduce the dimensionality of the original space. Rather, the dimensionality reduction occurs in comparison to the feature space instead. The fact that the original feature map can be infinite dimensional also means that the estimation has a form of automatic regularisation. This is because we only extract the projections of a finite amount of components from the infinite dimensional feature map. The regularisation is even necessary as it is not feasible to directly decompose matrices of infinite size.

A drawback of using either weighting or reproducing kernel matrices for ICS is that the components resulting from kernel methods will not be affine invariant. This is because any data transformation has a non-linear effect on the result via the kernel function. The weighting kernels are the most straightforward to incorporate because they can be used in place of other robust scatters. Pairs like $\{\text{LCov},\text{Cov}_2\}$ and $\{\text{ALCov}_\alpha,\text{Cov}_4\}$ can be tried with different kernel functions and parameter settings for example. The most important parameter to set for ALCov is the amount of guaranteed neighbours. This can be done via a grid search over different percentages of observations. The required computational time for trying different settings is much lower in comparison to methods like the RMCD.

An advantage of obtaining components generated via reproducing kernels is that they are automatically centered in feature space as part of the estimation. A drawback of using ICS with reproducing kernels is that transformation–retransformation of new datapoints is no longer directly possible. This is due to the feature projection step that prohibits an exact reconstruction. Interesting pairs can be obtained by considering different forms of informational contrast such as linearity versus non-linearity and a kernel versus its generalisation. This leads to pairs such as {Gaussian, Linear} and {Rational, Gaussian} respectively. The Hadamard product $\odot$ can also be used to generate new pairs. This element-wise matrix product retains the PSD property as a consequence of the Schur product theorem (Schur, 1911). It can effectively be seen as adjusting the kernel to reproduce a different feature map. Optimally performing parameters for RKHS kernels can be obtained via cross-validation. A good value for the norm scaling parameter that most kernels use can be obtained via the median heuristic, which is thoroughly studied in Garreau et al. (2017).

It is important to note that the numerical rank of reproducing kernels can decrease due to different factors. One numerical rank is always lost due to the required centering in feature space and further numerical ranks can be lost due to observations being colinear in feature space. The latter mostly occurs when the data does not provide sufficient distinct information to completely fill the feature space. The fact that kernel matrices are always singular means that a direct generalised eigendecomposition will run into unexpected results like negative or complex eigenvalues. We can resolve this by using the GSVD as shown by Archimbaud (2018). Given a kernel pair $\{K_1, K_2\}$ with $\text{rank}(K_2) = r < n$, it is still possible to decompose $V_{n\times r}D_{r\times r}U'_{r\times n}$ as given by the GSVD for the required solution.

The computation of ICS with kernel pairs is straightforward in the sense that the kernel pair just replaces the scatter pair. However, one kernel matrix is often of much more lower numerical rank than the other in practice. An unfortunate result of this is that even the GSVD method can run into numerical problems in that situation, based on experimentation with different kernel pairs. This issue can be addressed via the reduced rank simultaneous diagonalisation discussed in Yu and Yang (2001). Given a kernel pair $\{K_1, K_2\}$ with $\mathrm{rank}(K_2) = r < n$, the reduced rank procedure consists of whitening K_2 via the first r singular vectors and subsequently using that to transform and decompose K_1 separately. The rank reduced simultaneous diagonalisation therefore consists of the following two modified SVD steps:

$$\begin{aligned} S_{r\times r}^{-\frac{1}{2}} P'_{r\times n} K_2 P_{n\times r} S_{r\times r}^{-\frac{1}{2}} &= \mathrm{I}_{r\times r}, \\ Q'_{r\times r} K_1^* Q_{r\times r} &= \Lambda_{r\times r}. \end{aligned}$$

The matrices $S_{r\times r}$ and $P_{n\times r}$ represent the restricted singular values and singular vectors of K_2, and the matrices $\Lambda_{r\times r}$ and $Q_{r\times r}$ are the restricted singular values and singular vectors of $K_1^* = (S_{r\times r}^{-\frac{1}{2}} P'_{r\times n} K_1 P_{n\times r} S_{r\times r}^{-\frac{1}{2}})$. The required generalised eigenvectors are subsequently given by the matrix $T_{n\times r} = P_{n\times r} S_{r\times r}^{-\frac{1}{2}} Q_{r\times r}$, with the corresponding generalised eigenvalues contained in the diagonal matrix $\Lambda_{r\times r}$.

Another consequence of numerical rank deficiency imbalance is that kernel pairs cannot be reversed for the inverted results in general. Based on experimentation with various kernel pairs, we recommend to apply the reduced rank simultaneous diagonalisation via whitening of the kernel matrix with lowest rank. Additionally reversing the order of the remaining components is sometimes also useful in that case. The goal of this reversion is to move important components to the front. Subsequently, we can also discard any components exceeding numerical rank of the kernel matrix with the highest rank. We find that the most interesting components can be found along either the first or last few components in general.

5 A New Julia Package: SimultaneousDiagonalisation.jl

The original numerical implementation of ICS is written in the R programming language by Nordhausen et al. (2008). The package is designed for the use of ICS with scatter matrices. Some initial testing reveals that it is unable to deal with the additional numerical issues that arise from using kernel matrices. This means that a new numerically stable implementation of ICS is required for using the various kernel induced pairings. An additional benefit of creating a new implementation can come in the form of using a more modern faster programming language that handles large datatsets more efficiently.

A fresh code implementation of PCA, ICS and other related methods is created in the Julia programming language Bezanson et al. (2017) as part of this thesis. Julia is a relatively new dynamically typed programming language with multiple dispatch which makes it fitting for scientific computing with large datasets. The package is called "SimultaneousDiagonalisation.jl" and can be downloaded at: `https://github.com/CClaassen/SimultaneousDiagonalisation.jl`. The primary function of the package is to obtain scatter and kernel pairs, accomplish the diagonalisation, and do component selection. The package can also be used for classification and evaluation in addition to generating figures like the ones seen in this paper.

A quick summary of the package is that it contains implementations for 8 scatter matrices, 11 smoothing kernels, 19 reproducing kernels, 10 component selection methods, 3 classifiers and 30 classification evaluation metrics. Appendix A contains more details and a table of all functions per source file. An important implementation detail is that the sign ambiguity of the eigenvectors is broken by setting the signs such that the absolute largest entry is positive. Moreover, kernel matrices are decomposed via the rank reduced SVD procedure by default for numerical stability. The matrix decompositions are done by calling the appropriate functions from LAPACK (Anderson et al., 1999). Some of the more complex methods such as SVM[2], t-SNE[3] and UMAP[4] are included via bindings to separate packages.

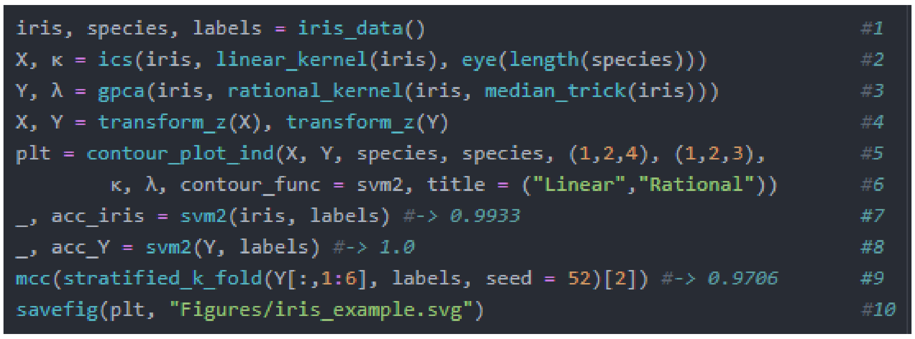
The most important function of the Julia package is the *ics* function. This function computes the diagonalisation of the scatter or kernel pair and subsequently performs component selection if desired. A similar procedure is followed for the decomposition of a single scatter or kernel via the *gpca* function. These functions allow for the decompositions to be done via several methods. The default choices depend on if a simultaneous diagonalisation is required and whether the pair consists of kernels or not. The default method that is used to simultaneously decompose a pair of kernel matrices in a numerically stable way is given in Algorithm 1 below. The method is equal to the generalised eigendecomposition if the matrix $B_{n\times n}$ is full rank. Moreover, the method is equal to the standard eigendecomposition if the matrix $B_{n\times n}$ is taken to be the identity matrix. The positions of $A_{n\times n}$ and $B_{n\times n}$ should be reversed if $A_{n\times n}$ has a lower rank than $B_{n\times n}$ as mentioned before. The components can be computed via $Y_{n\times r} = V_{n\times r}\Lambda_{r\times r}^{\frac{1}{2}}$ after running the rank reduced SVD method. The square roots of the eigenvalues are used in the kernel case as this standardises the data in feature space.

```
Algorithm 1: $(V_{n\times r}, \Lambda_{r\times r})$ = RankReducedSVD($A_{n\times n}$,$B_{n\times n}$)
    Input: Kernels or Scatters $A_{n\times n}$ and $B_{n\times n}$
    1: Compute rank($B_{n\times n}$) = $r$
    2: Compute $P_{n\times r}$ and $S_{r\times r}$ via the thin SVD of $B_{n\times n}$: $P'_{r\times n}B_{n\times n}P_{n\times r} = S_{r\times r}$
    3: Compute $T_{n\times r} = P_{n\times r}S_{r\times r}^{-\frac{1}{2}}$
    4: Compute $A^*_{r\times r} = T'_{r\times n}A_{n\times n}T_{n\times r}$
    5: Compute $Q_{r\times r}$ and $\Lambda_{r\times r}$ via the SVD of $A^*_{r\times r}$: $Q'_{r\times r}A^*_{r\times r}Q_{r\times r} = \Lambda_{r\times r}$
    6: Compute $V_{n\times r} = T_{n\times r}Q_{r\times r}$
    7: Fix the signs of $V_{n\times r}$
    Output: Generalised Eigenvectors $V_{n\times r}$, Generalised Eigenvalues $\Lambda_{r\times r}$
```

We conclude this section by looking at a demonstration of the package. Our goal is to reduce the dimensionality of the Iris dataset by using kernel PCA via two different kernels. Kernel PCA with kernel K is the same as ICS with the kernel pair $\{K, \mathrm{I}\}$, where the identity matrix I is also referred to as the white noise kernel. The command lines that are used for the demonstration are visible in Figure 8 situated on the next page. The procedure is to obtain the data and make the decomposition via either the *ics* or *gpca* functions. Next, we standardise the data such that we can make a SVM based scatter-wise contour plot. Subsequent evaluation is done by both calling the classifier directly and using stratified cross-validation. An interesting note here is that Kernel PCA with the rational kernel is able to classify all species perfectly. The resulting plot is shown in Figure 9 on the next page, where the clear differences between the results of the kernels can be seen. The bottom left part contains the results of the Linear kernel

[2]The source code for the SVM package can be found at: https://github.com/JuliaML/LIBSVM.jl
[3]The source code for the t-SNE package can be found at: https://github.com/lejon/TSne.jl
[4]The source code for the UMAP package can be found at: https://github.com/dillondaudert/UMAP.jl

with corresponding eigenvalues κ_i and the top right part contains the results of the Rational kernel with corresponding eigenvalues λ_j. The non-diagonal entries of the figure contain pair-wise scatter plots of certain components. The diagonal entries of the figure contain comparisons of the shape of individual components to the shape of the normal distribution. This is of interest if we want to base our component selection on (non-)normality of the data.



```
iris, species, labels = iris_data()                                       #1
X, κ = ics(iris, linear_kernel(iris), eye(length(species)))               #2
Y, λ = gpca(iris, rational_kernel(iris, median_trick(iris)))              #3
X, Y = transform_z(X), transform_z(Y)                                     #4
plt = contour_plot_ind(X, Y, species, species, (1,2,4), (1,2,3),          #5
        κ, λ, contour_func = svm2, title = ("Linear","Rational"))         #6
_, acc_iris = svm2(iris, labels) #-> 0.9933                               #7
_, acc_Y = svm2(Y, labels) #-> 1.0                                        #8
mcc(stratified_k_fold(Y[:,1:6], labels, seed = 52)[2]) #-> 0.9706         #9
savefig(plt, "Figures/iris_example.svg")                                  #10
```

Figure 8: A small demonstration of the package.

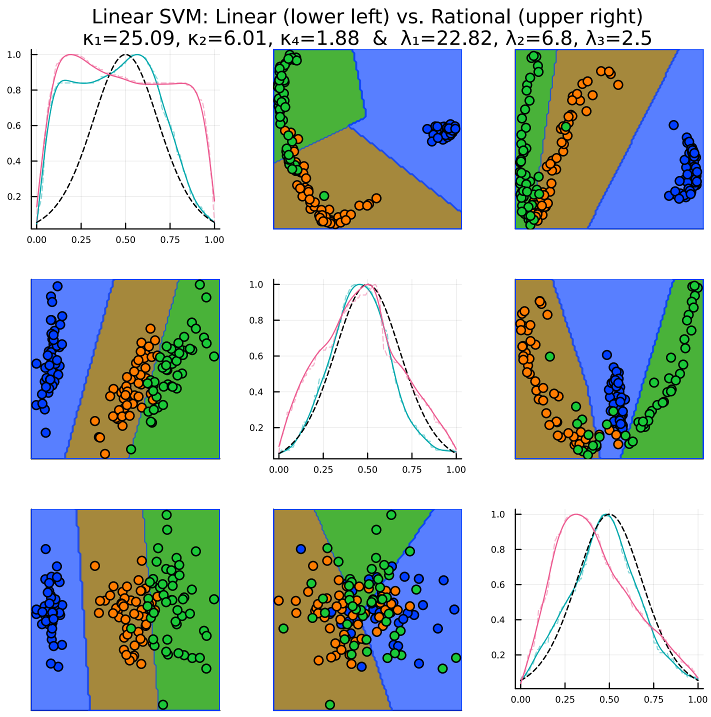


Figure 9: Pair-wise scatters of kernel PCA with two different kernels on the Iris data.

6 Empirical Applications

We now turn our attention to applying the previously discussed methods and extensions to different datasets. We compare the performance of the various kernel induced pairs against each other and alternative widely used scatter pairs. This is done in order to assess whether capturing non-linear structure can be useful in practice. All figures and tables are reproducible by calling the appropriate functions from the package.

6.1 Data visualisation: Wine data

We first analyse the performance of our newly proposed kernel induced methods for data visualisation. This is done by using the Wine dataset[5]. The data consists of 129 wine samples with a chemical study determining 13 measurements for all samples. Ten wine samples are taken from a different cultivar than the rest. The goal is therefore to see if we can identify these outliers. In particular, we look at whether we are able to visualise this difference in two dimensions. A t-SNE visualisation of the Wine data is given in Figure 10 below. T-SNE is able to reveal that the structure is fairly simple in the sense that the outliers are positioned together. However, we would not be able to easily see this if the data was to be unlabeled. We therefore turn to a different dimensionality reduction method in the form of ICS.

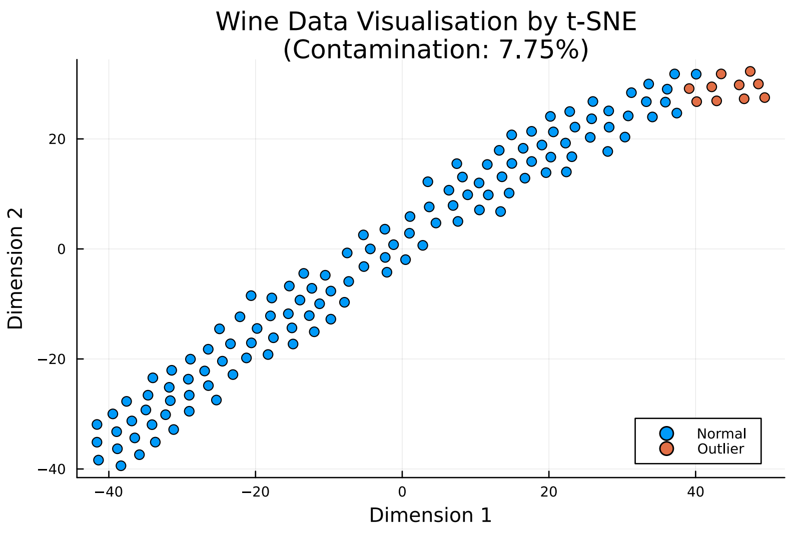
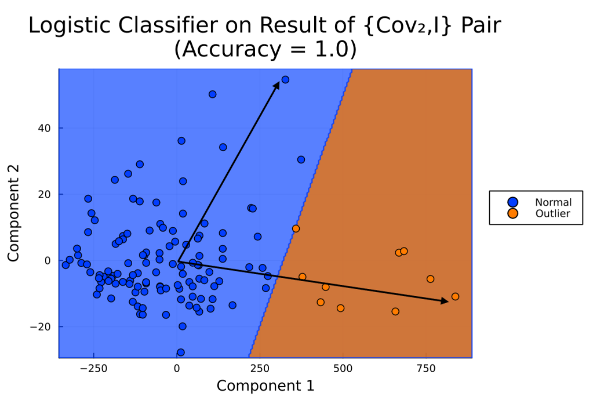


Figure 10: A two-dimensional visualisation of the Wine data via t-SNE.

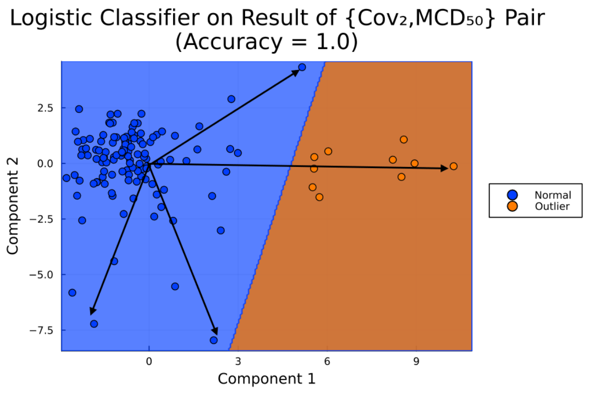
Further analysis is done by extracting components of ICS with various scatter pairings and running a linear classifier on the result. For the comparison, we consider the first, second and last components produced by all different setups. A small caveat here is that outlier detection is rarely done in a supervised context in practice. Nevertheless, it makes sense to do so here out of a practical evaluation perspective. Moreover, a simple linear classifier that is almost parallel to an axis is an indication that the structure is potentially also visible without labeling. The visualisation results of various scatter pairings on the Wine data can be found in Appendix B, of which we will discuss the best results here next.

[5]The Wine dataset can be accessed via: https://archive.ics.uci.edu/ml/datasets/Wine

Figure 11: Best subspaces of Wine data for different PCA and ICS methods.



(a) Best visualisation using PCA $\{\mathrm{Cov}_2\}$.

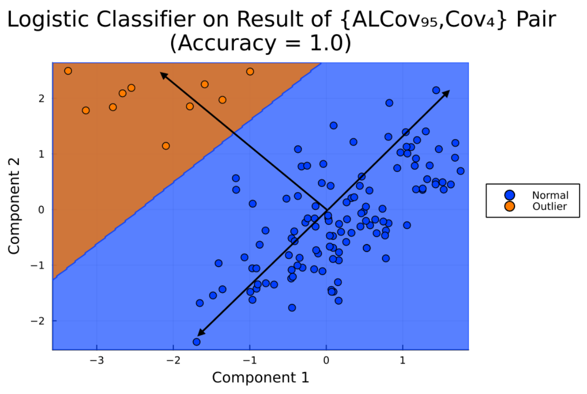


(b) Best visualisation using ICS $\{\mathrm{Cov}_2,\mathrm{MCD}_{50}\}$.

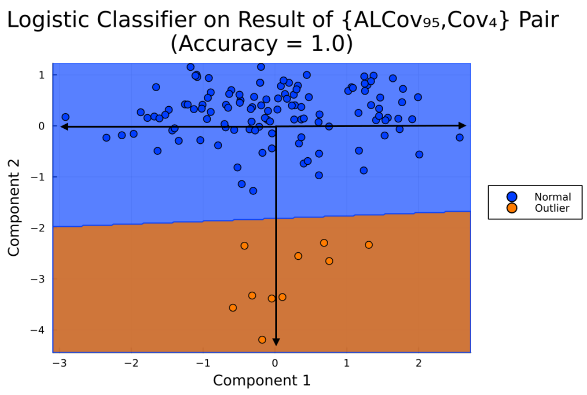
The best visualisation results of the $\{\mathrm{Cov}_2,\mathrm{I}\}$ and $\{\mathrm{Cov}_2,\mathrm{MCD}_{50}\}$ scatter pairings are shown in the panels of Figure 11 above. The subscript of the MCD denotes the proportion of observations over which the MCD is minimised in percentages. The I denotes that an identity matrix is used such that the pair simplifies to classic PCA. The left panel shows that we can perfectly separate the outliers if we use supervision with PCA. The arrows in the panel denote the two main directions of point dispersion. The fact that there are multiple arrows implies that it would be hard to find all outliers consistently if we do not have access to the labeling. The right panel shows the best result of using the $\{\mathrm{Cov}_2,\mathrm{MCD}_{50}\}$ pair, which is also able to separate the outliers under supervision. We again encounter multiple dispersion directions such that consistent unsupervised outlier detection would be difficult.

The best visualisation result of using a kernel smoothing induced pair is shown in the panels of Figure 12 on the next page. In particular, we use the ALCov via the Epanechnikov kernel in combination with the Cov_4 scatter. The left panel shows that the regular points form an ellipse-like form and that the outliers are dispersed differently. The arrows indicate that this is roughly in a direction orthogonal to the main axis of the normal points. It is therefore not hard to identify potential outliers in this picture. This is a good result given that ICS did not use any information of the labels in the computation. An even clearer picture is obtained by rotating the result via PCA. The right panel shows the effect of this, where the classification boundary is now almost parallel to the x-axis. A practical detail is that the ALCov nearest neighbour parameter was set via a grid search over equally spaced values without touching the scale parameter. The value that gives the best visualisation is found at a proportion of 95% of the total number of observations. It is therefore not hard to find this visualisation in practice with this method. We find that pairings containing the ARLCov were not able to generate interesting visualisations in this application.

Figure 12: Best subspaces of Wine data for pairs containing ALCov.



(a) Best visualisation using Epanechnikov kernel.



(b) Rotated visualisation using Epanechnikov kernel.

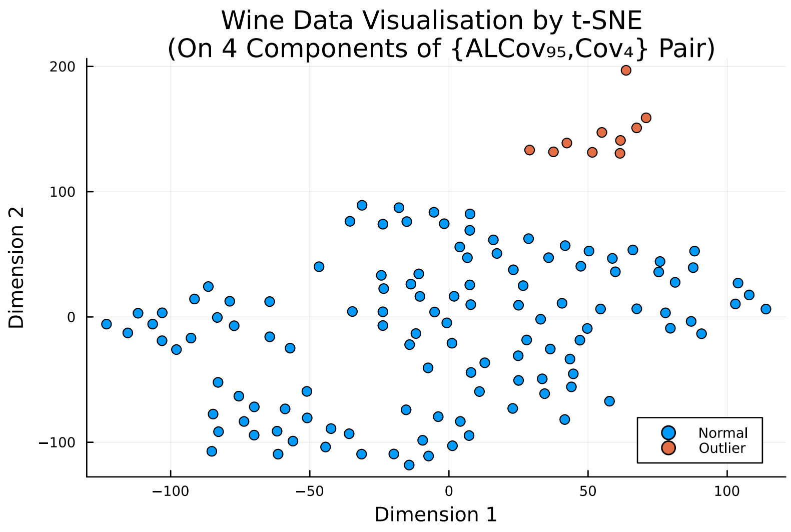


Figure 13: An alternative two-dimensional visualisation of the Wine data via t-SNE.

We now investigate whether we can use the components of kernel induced ICS as a pre-processing step for a different visualisation method. For that purpose, we generate a t-SNE embedding based on the first two and last two components of kernel induced ICS in Figure 13 given above. T-SNE now clearly displays that outliers are structurally different from normal observations. This could not be seen in the initial t-SNE embedding of Figure 10 which is based on PCA preprocessing. The new figure shows that it is possible to highlight different types of structure with t-SNE if we use a few components of ICS for the initialisation. It can be a good idea to investigate in future works whether using ICS with t-SNE is better than using PCA with t-SNE in general. This is because it appears that the benefits of using ICS here are not only that the computational burden is reduced, but also that a different type of structure is revealed.

6.2 Anomaly detection: Wisconsin Breast Cancer data

The next task we evaluate our methods on is anomaly detection. For this context, we use the *Wisconsin Breast Cancer* (WBC) data[6]. The WBC dataset consists of 357 benign breast cancer cases, whereas the other 21 included cases are malignant. There are a total of 30 measurements per sample. The goal is obviously to find whether we can isolate the malignant cases from the benign ones. Finding such structure can help in screening future cases. Figure 14 below shows that UMAP is unable to cluster all malignant cases together. Unfortunately, we also cannot find a matrix pairing for ICS that perfectly separates the two cases in two dimensions. It is therefore a better idea to expand the analysis to the situation where we retain more than two dimensions.

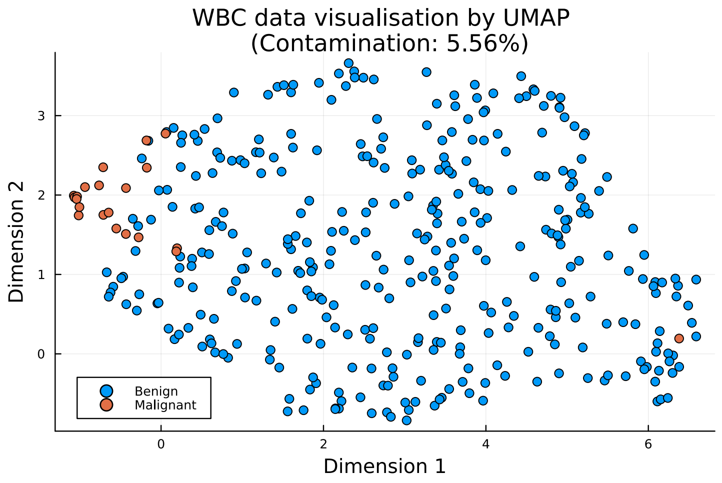


Figure 14: A two-dimensional visualisation of the WBC data via UMAP.

A thorough analysis of the performance of ICS on this data can be done by evaluating different matrix pairings. For the smoothing kernel induced pairs, we pick the Epanechnikov kernel from the unit ball support group and the Gaussian kernel from the infinite support group, both with different setups. The scale parameter of LCov is set by optimisation, whereas the nearest neighbour parameter of ALCov is again set via grid search over equally spaced proportion values. These methods are compared with various other scatter pairs found in the literature. Table 3 on the next page shows the results of using a linear SVM classifier via stratified 10-fold cross-validation on 5 components of all these matrix pairings. These 5 components consists of the first 3 in addition to the last 2 components. The best score per evaluation metric is in bold, where we treat the malignant class as the primary class in all evaluations. All pairs score well on accuracy, but this not a useful metric for this application due to the significant class distribution imbalance. Arguably the most important evaluation metrics are recall and F-score in this case. This is because would like to classify all malignant cases as malignant (=recall) whilst not classifying any benign cases as malignant (=F-score). Modified PCA using Cov_4 performs the best of all traditional scatter pairings. The table shows that parameter tuning is essential for good performance of pairs containing a kernel. A non-optimised parameter can give very poor results, as is evident with the last pair in the table. The ALCov pair containing the

[6]The WBC dataset can be accessed via: https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)

Epanechnikov kernel with 10% guaranteed nearest neighbours performs the best overall by quite a margin. However, the absolute performance is still not exactly satisfactory.

Table 3: Results of stratified 10-fold classification for 5 components of various scatter pairs (in %).

Evaluated Pair	ACC	PRC	REC	F_1	MCC
$\{Cov_2,I\}$	96.83	80.00	57.14	66.67	66.07
$\{Cov_4,I\}$	97.09	81.25	61.90	70.27	69.47
$\{Cov_4,Cov_2\}$	96.03	**100.0**	28.57	44.44	52.36
$\{Cov_2,MCD_{90}\}$	97.35	100.0	52.38	68.75	71.38
$\{Cov_2,MCD_{50}\}$	96.56	90.00	42.86	58.06	60.77
$\{MCD_{50},MCD_{90}\}$	96.56	**100.0**	38.10	55.17	60.63
$\{G\_LCov,Cov_2\}$	96.83	80.00	57.14	66.67	66.07
$\{G\_ALCov_{10},Cov_2\}$	97.09	85.71	57.14	68.57	68.63
$\{E\_LCov,Cov_2\}$	96.83	80.00	57.14	66.67	66.07
$\{E\_ALCov_{10},Cov_2\}$	**97.62**	83.33	**71.43**	**76.92**	**75.93**
$\{G\_ALCov_{90},Cov_2\}$	96.56	78.57	52.38	62.86	62.51
$\{E\_ALCov_{90},Cov_2\}$	94.71	60.00	14.29	23.08	27.52

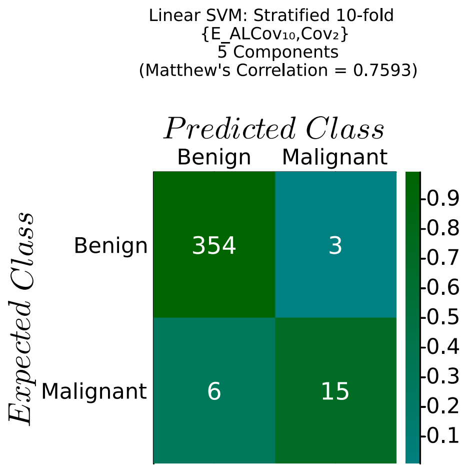
Table 4: Results of stratified 10-fold classification for 20 components of various scatter pairs (in %).

Evaluated Pair	ACC	PRC	REC	F_1	MCC
$\{Cov_2,I\}$	98.41	89.47	80.95	85.00	84.28
$\{Cov_4,I\}$	98.94	94.74	85.71	90.00	89.57
$\{Cov_4,Cov_2\}$	97.62	80.00	76.19	78.05	76.82
$\{Cov_2,MCD_{90}\}$	97.62	83.33	71.43	76.92	75.93
$\{Cov_2,MCD_{50}\}$	98.15	88.89	76.19	82.05	81.35
$\{MCD_{50},MCD_{90}\}$	96.83	76.47	61.90	68.42	67.18
$\{G\_LCov,Cov_2\}$	98.41	89.47	80.95	85.00	84.28
$\{G\_ALCov_{10},Cov_2\}$	98.41	89.47	80.95	85.00	84.28
$\{E\_LCov,Cov_2\}$	98.41	89.47	80.95	85.00	84.28
$\{E\_ALCov_{10},Cov_2\}$	**99.21**	95.00	**90.48**	**92.68**	**92.29**
$\{G\_ALCov_{90},Cov_2\}$	98.94	**100.0**	80.95	89.47	89.47
$\{E\_ALCov_{90},Cov_2\}$	98.15	88.89	76.19	82.05	81.35

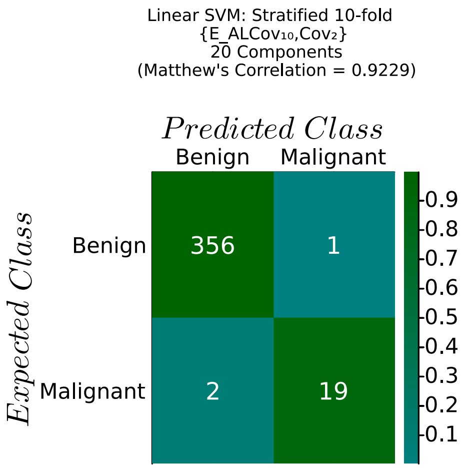
For this reason, it makes sense to expand the amount of components we retain from all pairs. Table 4 above shows the adjusted results of retaining the first 20 components, with the best results per metric in bold. All methods gain a considerable boost in performance, but this is at the cost of quite a few extra dimensions. The best performing matrix pair is still the same one as with 5 components, but some other scatter pairings come much closer in terms of performance now. Kernel induced pairs score somewhat better overall than the traditional scatter pairs. This is partially because the information provided by the MCD does not seem to be of much use with this dataset. The table once again shows that good performance of pairs containing a kernel is dependent on the kernel parameter settings.

The choice for picking these specific components are motivated via the classification results on the panels of Figure 15 given on the next page. The amount of components to retain are manually chosen. They are picked in such a way that the total amount of missclassifications of the best method increases by a factor of 3 for each further reduction step. Put differently, the step from 5 to 20 dimensions reduces the amount of missclassifications threefold, and the latter's missclassifications are three times that of the original data. This is done to show the trade-off between performance and dimensionality reduction. Based on the results in all these tables, it would be wise to retain 20 components rather than 5 via the {E_ALCov$_{10}$,Cov$_2$} for this application. This incurs a loss in performance when compared to the original, but it could still be acceptable for an application like preliminary screening. We can also choose to retain fewer components, but this does not seem sensible in an application like this. Dimensionality reduction should not be applied in case we absolutely want maximal performance, but there is also the danger of overfitting to our available data. Switching to a computationally simpler method such as PCA is possible, but this is also at the cost of quite a bit of performance. In summary, we have seen that using kernel induced scatters compares favorably to other scatter pairs in applications like this.

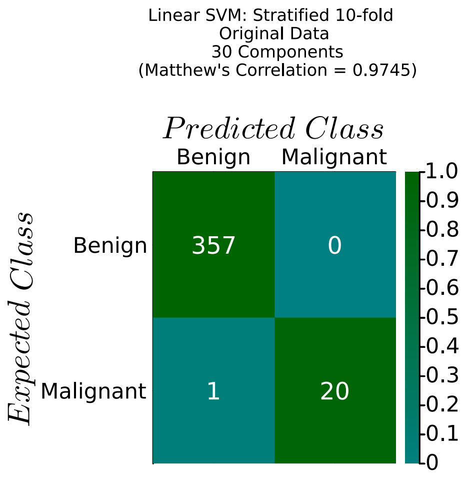
Figure 15: Overview of confusion tables for the best results and the orginal



(a) Best results for 5 components.



(b) Best results for 20 components.



(c) Best results for original 30 components.

6.3 Classification: word embbeding data - Word2Vec, GloVe and FastText

The last task we use for the evaluation of our methods is classification. The datasets we will use for this purpose are generated by various deep learning models. In particular, we consider word embeddings generated by Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and FastText (Bojanowski et al., 2017). A word embedding is a vector representation of a word that is constructed in such a way that further analysis can be improved. For example, words with similar meanings get vector representations with small angles between them. The most important thing to know is that word embedding models primarily rely on contextual information for a good representation. More details on the construction and inner workings can be found in the papers that introduced these word embeddings models. We sample 601 word embeddings from pre-trained versions of the three models that all have a dimensionality of 300 each. These words are all labeled as being part of one of eleven (unbalanced) classes. Some classes are expected to be strongly interconnected, such as the group consisting of countries and the group consisting of the corresponding capitals.

It is often the case that a particular application does not actually require the full 300 dimensions needed to train millions of words and their interactions. Put differently, it is likely that the inherent dimensionality of only part of the data is much lower than 300. Another challenge is that it is not always clear which model should work best for a given application. These are the issues that we will attempt to address via dimensionality reduction for this application. Figure 16 below shows a t-SNE embedding of the Word2Vec data. It can be seen that multiple clusters form, but also that various clusters attract observations from different classes. T-SNE is also unable to strongly separate some groups like countries and capitals for example. Perhaps we can remedy this by using the contrast between non-linear structures provided by reproducing kernel induced ICS.

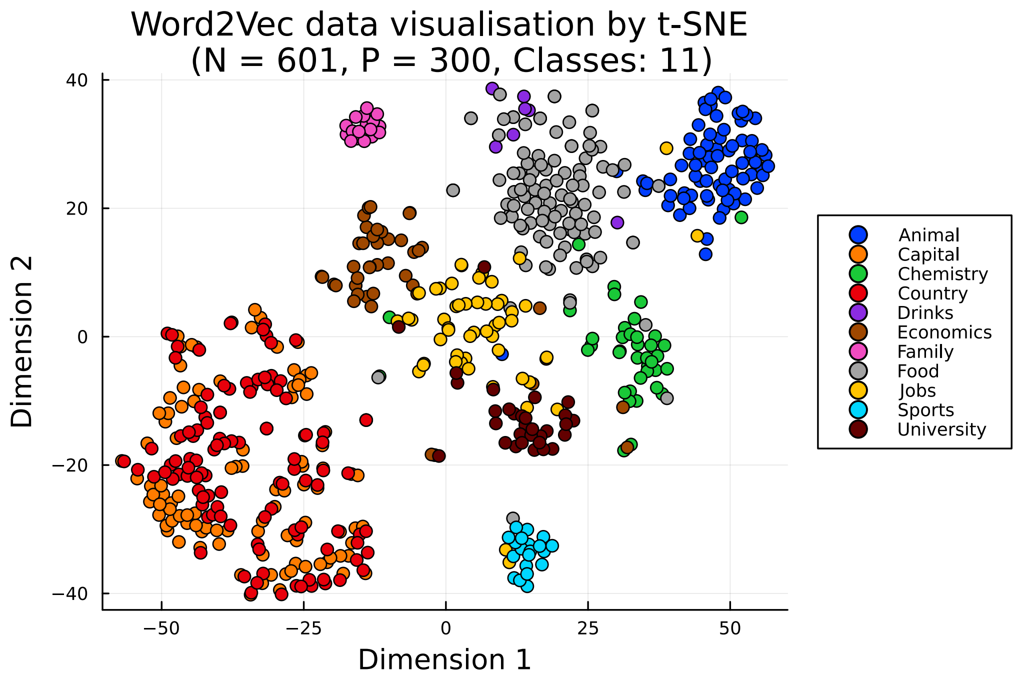


Figure 16: A two-dimensional visualisation of the Word2Vec data via t-SNE.

We judge the performance of the three word embedding models by using stratified 10-fold cross-validation in combination with the linear SVM classifier. We evaluate retaining different amounts of components for various different kernel pairings. Amongst the investigated setups are also the original data, standard PCA via the linear kernel and various other forms of kernel PCA. Comparison is done by calculating the multiclass MCC instead of using the aggregation of binary classification metrics. We are primarily interested in whether the non-linear contrast of reproducing kernels can be used to improve the results. This results in kernel pairings like {Gaussian,Linear} and {Rational, Linear$\odot$Gaussian}. This latter pair is used to analyse whether using the Hadamard product can be beneficial.

Tables 5, 6 and 7 show all relevant evaluation results with the best results per column in bold. The tables are situated on the next page. The 'All' column represents all components up to numerical rank and the 'Best' column represents the best result of an exhaustive search over the first 20 components. We first notice that general performance initially rises as components are added, and it then seemingly declines after a certain amount of components. The optimal amount of components seems to be around 15 for most methods. It can also be seen that Word2Vec does not really benefit from using two kernels as opposed to one, whereas the other two models do. The best result per model is achieved by kernel pairs and it is interesting to see that these are different pairs for every model. Next, it is remarkable that certain kernel pairs have worse than random performance when all components are used. This is an indication that there are still numerical rank deficiencies that lead to significant noise. We should therefore be careful with selecting components for reproducing kernel induced ICS. A last remark is that the performance of the complete original data is lower than that of one or more kernel pairs for all models.

The FastText embeddings should be preferred over Word2Vec and GloVe for the application of classifying the word embeddings. This is because FastText achieves the best performance of all methods with only 16 components via the {Gaussian,Linear} kernel pair. A solid result is that these 16 components perform better in cross-validation than the 300 dimensional original. FastText therefore benefits from reproducing kernels fairly well, but the differences in evaluation metrics between the different methods are somewhat small. Nevertheless, it is an indication that using ICS for classification can benefit from capturing non-linear interactions. A small caveat here is that the MCC represents a correlation such that small differences can still be impactful at this sample size. A more detailed overview of the best classification result given by FastText can be accessed in Appendix C. The primary takeaway of that classification table is that the classifier is able to separate all classes well, except for the class consisting of drinks and the class consisting of foods. Most drinks are classified as food instead. This is perhaps because these groups of words are almost always used in the same context as the other. It should also be noted that there are many more samples of the food class than there are of the drinks class.

Table 5: Word2Vec: MCC of stratified 10-fold validation for various pairings and components (in %).

Evaluated Pair \Components	5	10	15	25	50	100	All	Best
Original Data	30.35	54.29	63.58	69.35	82.49	87.64	**90.31**	67.85
{Linear,I}	**85.20**	89.96	89.18	**89.00**	87.08	86.12	77.41	91.08
{Gaussian,I}	85.00	89.91	89.16	88.80	87.27	86.14	-1.03	91.08
{Rational,I}	83.15	88.41	89.18	88.98	**88.61**	86.69	-0.92	89.94
{Gaussian,Linear}	84.06	**90.69**	**91.26**	88.60	86.32	86.31	76.93	91.26
{Rational,Linear}	81.41	88.60	88.99	86.35	86.89	86.11	67.79	89.57
{Rational,Gaussian}	83.32	89.74	90.88	88.42	86.32	85.75	-6.85	90.88
{Rational, Linear$\odot$Gaussian}	80.79	88.60	91.07	88.42	86.32	**87.65**	-7.25	**91.45**

Table 6: GloVe: MCC of stratified 10-fold validation for various pairings and components (in %).

Evaluated Pair \Components	5	10	15	25	50	100	All	Best
Original Data	46.33	58.11	67.57	74.11	81.75	88.21	**90.50**	70.52
{Linear,I}	84.47	90.12	89.17	88.8	87.27	87.45	82.21	90.34
{Gaussian,I}	79.60	88.41	89.36	89.35	88.61	85.92	-1.06	89.36
{Rational,I}	81.60	84.98	86.52	89.36	88.03	**89.73**	-0.59	88.41
{Gaussian,Linear}	**85.04**	90.32	89.36	88.6	86.54	87.64	82.08	90.91
{Rational,Linear}	84.81	89.56	88.41	89.55	86.74	87.45	80.58	89.56
{Rational,Gaussian}	64.86	**90.94**	91.07	89.93	89.35	89.73	-5.81	**92.03**
{Rational, Linear$\odot$Gaussian}	82.57	90.55	**91.85**	**91.46**	**89.55**	84.03	-18.83	91.85

Table 7: FastText: MCC of stratified 10-fold validation for various pairings and components (in %).

Evaluated Pair \Components	5	10	15	25	50	100	All	Best
Original Data	37.46	65.82	65.94	72.61	83.45	88.02	**90.68**	71.84
{Linear,I}	83.11	89.59	91.46	**90.50**	89.18	87.27	81.06	91.65
{Gaussian,I}	83.69	89.57	91.83	90.46	89.37	87.65	-1.06	92.02
{Rational,I}	73.79	85.38	85.74	89.93	89.92	88.60	-1.48	88.42
{Gaussian,Linear}	82.17	91.13	**91.85**	90.13	88.61	86.32	80.93	**92.21**
{Rational,Linear}	85.21	**91.49**	91.66	87.67	86.55	84.79	76.35	92.02
{Rational,Gaussian}	**86.16**	89.39	91.07	90.31	**89.94**	**88.97**	-6.85	91.64
{Rational, Linear$\odot$Gaussian}	79.02	87.86	90.53	88.78	88.02	84.05	-14.75	91.68

We close this section by briefly looking at using ICS with reproducing kernels for preprocessing purposes. In particular, we look at the difference between PCA and ICS for the initialisation of a t-SNE visualisation. Figure 17 below shows the result of using t-SNE via a PCA initialisation on the original FastText data. The figure shows that the classes are less easily distinguishable when compared to the t-SNE visualisation of Word2Vec data previously seen in Figure 16. An improved t-SNE visualisation of the FastText data via an ICS initialisation can be seen in Figure 18 below. The initialisation consists of using ICS with the first 16 components of the {Gaussian,Linear} pair because that was the best performing setup in the cross-validation analysis. The resulting clusters are now much tighter, although some observations are still attracted towards the wrong class. Some notable changes are the clearer split between capitals and countries, and the link between the animal and food classes. These results are an indication that reproducing kernel induced ICS is able to provide non-linear structure that PCA is unable to represent.

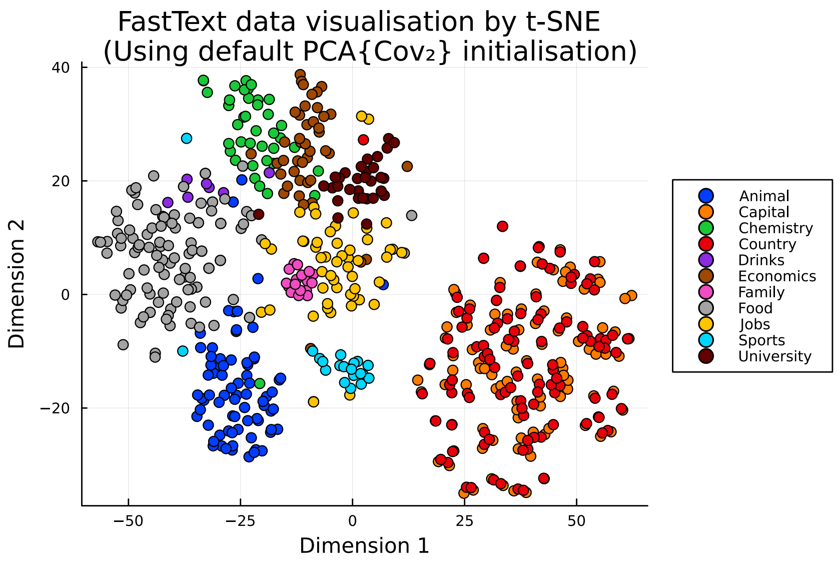


Figure 17: A two-dimensional visualisation of the FastText data via t-SNE.

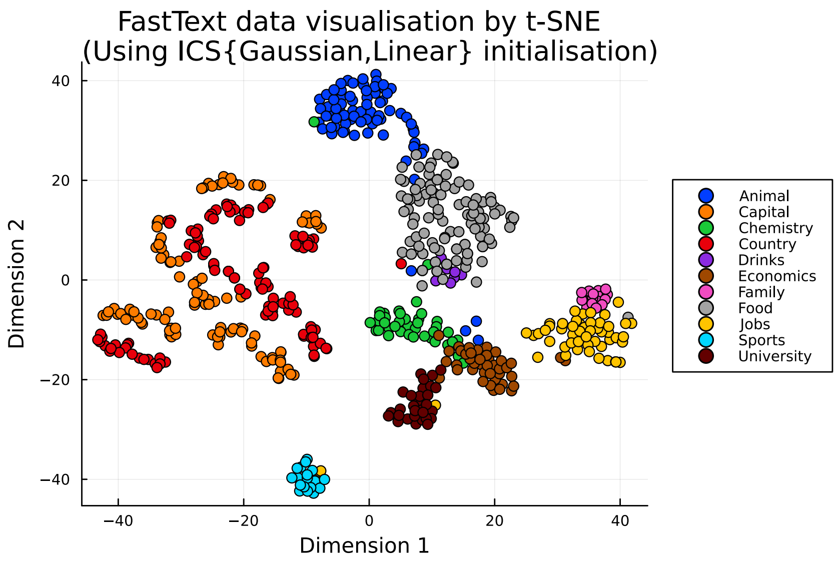


Figure 18: An improved two-dimensional visualisation of the FastText data via t-SNE.

7 Conclusion

In this paper, we generalised Invariant Coordinate Selection to a non-linear dimensionality reduction method via kernelisation. This was done by first studying the relevant literature and providing an overview of the simultaneous diagonalisation framework. The next step consisted of providing extensions in the form of kernel techniques and addressing practical numerical issues concerning these extensions. A new Julia package that implements all these methods was also discussed. We have seen that the results of these newly introduced kernel pairs compare favorably to the results of previously used scatter pairs. More specifically, we have empirically verified that this is the case for applications like data visualisation, anomaly detection and classification.

The starting point of this paper is Invariant Coordinate Selection. It is a method that is able to reveal structure in a low-dimensional space. ICS works by using the informational contrast between two scatter matrices. These scatter matrices can be thought of as a generalisation of the standard covariance matrix and they are able to capture different kinds of structure. The potential of ICS lies in the fact that there are many types of scatters, and that these can all be combined for revealing various types of structures. However, a drawback of ICS is that it is only able to deal with linear structure. It is this limitation that is addressed in this paper.

The first main contribution of this thesis comes in the form of providing additional pairs for use in the simultaneous diagonalisation framework. These new pairs make use of kernels to allow for capturing non-linear structure. Both the extensions of smoothing kernels and reproducing kernels are used to provide kernel induced scatters that can form pairings for ICS. Some practical details on the implementation of these kernels techniques are also given. This is part of the second main contribution of this thesis, which consists of providing a numerically stable way to practically apply ICS with kernels pairings. A brand new Julia package was created for that purpose as part of the process. Using kernels for ICS comes with its own set of challenges. The most noteworthy ones are the requirement of kernel parameter optimisation and the existence of noisy components due to kernel matrix rank deficiency. It is unfortunate that not all nice properties of ICS with scatters carry over to ICS with kernels, but the results of using kernel methods are promising nonetheless.

The work done in this thesis builds on that of many others, and much more can still be achieved. We see three primary directions for future works that use ICS with kernel pairs. The first direction concerns component selection. More work on determining which components should be extracted from ICS with kernels is needed and it could be interesting to see whether this differs from component selection using scatter pairs. The second direction comes in terms of the kernels themselves. It would be useful to analyse the effect of jointly optimising kernel hyperparameters via cross-validation further. This poses a big computational burden, but that can be mitigated by using the Nyström approximation (Li et al., 2010) that is already supported by the package. For the last direction, we note that it could be useful to investigate the use of (kernel induced) ICS as a preprocessing step for t-SNE. This can be fruitful because the initialisation step is of fundamental importance to the type of structure uncovered by t-SNE (Kobak and Linderman, 2021). We have already briefly seen the potential of it in this thesis.

References

Alfons, A., Archimbaud, A., Nordhausen, K., and Ruiz-Gazen, A. (2022). Tandem clustering with invariant coordinate selection. *arXiv preprint arXiv:2212.06108*.

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. (1999). *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition.

Archimbaud, A. (2018). *Méthodes statistiques de détection d'observations atypiques pour des données en grande dimension*. PhD thesis, Université de Toulouse. Thèse de doctorat dirigée par Ruiz-Gazen, Anne et Nordhausen, Klaus Mathématiques appliquées Toulouse 1 2018.

Archimbaud, A., Nordhausen, K., and Ruiz-Gazen, A. (2018). ICS for multivariate outlier detection with application to quality control. *Computational Statistics & Data Analysis*, 128:184–199.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404.

Bach, F. R. and Jordan, M. I. (2002). Kernel independent component analysis. *Journal of machine learning research*, 3(Jul):1–48.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.

Caussinus, H. and Ruiz, A. (1990). Interesting projections of multidimensional data by means of generalized principal component analyses. In *Compstat*, pages 121–126. Springer.

Chang, W.-C. (1983). On using principal components before separating a mixture of two multivariate normal distributions. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 32(3):267–275.

De Soete, G. and Carroll, J. D. (1994). K-means clustering in a low-dimensional euclidean space. In *New approaches in classification and data analysis*, pages 212–219. Springer.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

Garreau, D., Jitkrittum, W., and Kanagawa, M. (2017). Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*.

Ghojogh, B., Ghodsi, A., Karray, F., and Crowley, M. (2021). Reproducing kernel Hilbert space, mercer's theorem, eigenfunctions, Nystrom's method, and use of kernels in machine learning: Tutorial and survey. *arXiv preprint arXiv:2106.08443*.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix computations*. JHU press.

Gorodkin, J. (2004). Comparing two K-category assignments by a K-category correlation coefficient. *Computational biology and chemistry*, 28(5-6):367–374.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417.

Hubert, M., Rousseeuw, P. J., and Verdonck, T. (2012). A deterministic algorithm for robust location and scatter. *Journal of Computational and Graphical Statistics*, 21(3):618–637.

Kobak, D. and Linderman, G. C. (2021). Initialization is critical for preserving global data structure in both t-sne and umap. *Nature biotechnology*, 39(2):156–157.

Kramer, M. A. (1991). Non-linear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Li, M., Kwok, J. T.-Y., and Lü, B. (2010). Making large-scale nyström approximation possible. In *Proceedings of the 27th International Conference on Machine Learning, ICML 2010*, page 631.

McInnes, L., Healy, J., and Melville, J. (2018). UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society*, 209:4–415.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nordhausen, K., Oja, H., Filzmoser, P., and Reimann, C. (2015). Blind source separation for spatial compositional data. *Mathematical Geosciences*, 47:753–770.

Nordhausen, K., Oja, H., and Tyler, D. E. (2008). Tools for exploring multivariate data: The package ICS. *Journal of Statistical Software*, 28(6):1–31.

Nordhausen, K. and Ruiz-Gazen, A. (2022). On the usage of joint diagonalization in multivariate statistics. *Journal of Multivariate Analysis*, 188:104844.

Nordhausen, K. and Virta, J. (2019). An overview of properties and extensions of FOBI. *Knowledge-Based Systems*, 173:113–116.

Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Powers, D. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63.

Riesz, F. (1907). *Sur une espèce de géométrie analytique des systèmes de fonctions sommables.* Gauthier-Villars.

Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function. *The annals of mathematical statistics*, pages 832–837.

Rousseeuw, P. J. (1985). Multivariate estimation with high breakdown point. *Mathematical statistics and applications*, 8(283-297):37.

Rousseeuw, P. J. and Driessen, K. V. (1999). A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.

Schur, I. (1911). Bemerkungen zur theorie der beschränkten bilinearformen mit unendlich vielen veränderlichen.

Theil, H. (1969). A multinomial extension of the linear logit model. *International economic review*, 10(3):251–259.

Tyler, D. E., Critchley, F., Dümbgen, L., and Oja, H. (2009). Invariant coordinate selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):549–592.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Vapnik, V. (1963). Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780.

Wang, W., Huang, Y., Wang, Y., and Wang, L. (2014). Generalized autoencoder: A neural network framework for dimensionality reduction. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 490–497.

Weiszfeld, E. (1937). Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386.

Yu, H. and Yang, J. (2001). A direct lda algorithm for high-dimensional data—with application to face recognition. *Pattern recognition*, 34(10):2067–2070.

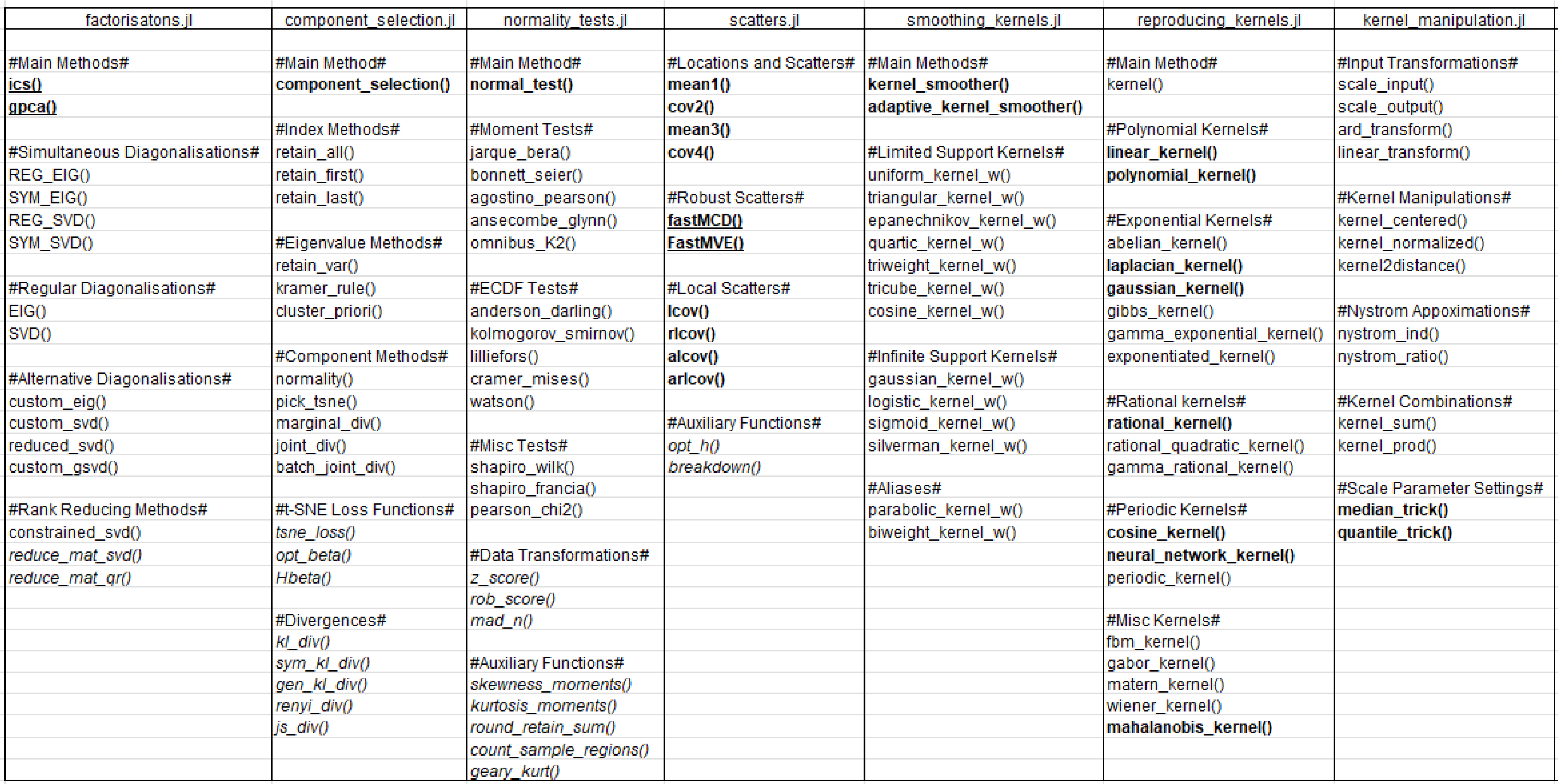
Appendices

A Code Overview

The complete SimultaneousDiagonalisation.jl package consists of 15 source files and the accompanying module file. The package was originally written in Julia version 1.7.3, but is also tested and confirmed to be working on the latest stable release which is Julia version 1.8.5 at the time of writing. Any breaking changes caused by dependencies should not occur before the release of Julia version 2.0. A short description of all source files is given below. An overview of all functions defined in the package can be seen in two tables on the next two pages. More details about individual functions can be found in the documentation of the package. The function *all_ experiments()* from the file *experiments.jl* reproduces all figures and tables from the paper. The complete package can be accessed and downloaded at: `https://github.com/CClaassen/SimultaneousDiagonalisation.jl`.

Short description of all source files:
- *SimultaneousDiagonalisation.jl* contains the main module of the package.
- *factorisatons.jl* contains the main methods to compute PCA and ICS.
- *component_ selection.jl* contains functions related to component selection.
- *normality_ tests.jl* contains functions for univariate normality tests.
- *scatters.jl* contains functions for computing various scatter matrices.
- *smoothing_ kernels.jl* contains functions for using smoothing kernels.
- *reproducing_ kernels.jl* contains functions for using reproducing kernels.
- *kernel_ manipulation.jl* contains functions for transforming kernels.
- *classification.jl* contains functions related to classification.
- *evaluation.jl* contains functions for evaluating classification results.
- *external_ methods.jl* contains bindings to external methods.
- *figures.jl* contains functions for plotting different types of figures.
- *data_ manipulation.jl* contains functions for transforming data.
- *utilities.jl* contains various useful auxiliary functions.
- *external_ data.jl* contains functions for loading external data.
- *experiments.jl* contains functions for replicating all results from the paper.

Figure 19: Overview of all implemented functions from the SimultaneousDiagonalisation.jl package per source file (1/2).



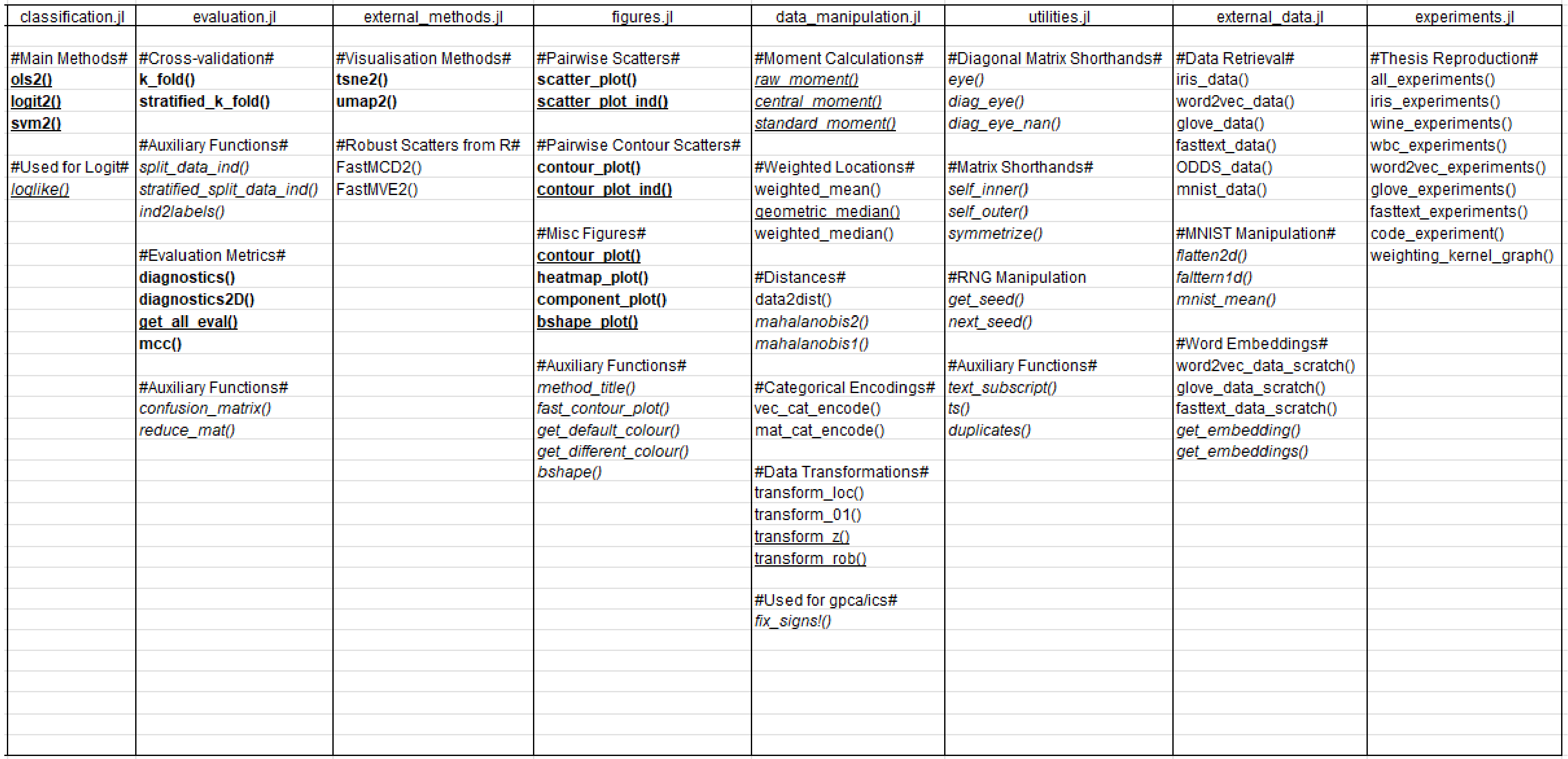
factorisatons.jl	component_selection.jl	normality_tests.jl	scatters.jl	smoothing_kernels.jl	reproducing_kernels.jl	kernel_manipulation.jl
#Main Methods#	#Main Method#	#Main Method#	#Locations and Scatters#	#Main Methods#	#Main Method#	#Input Transformations#
ics()	**component_selection()**	**normal_test()**	**mean1()**	**kernel_smoother()**	kernel()	scale_input()
gpca()			**cov2()**	**adaptive_kernel_smoother()**		scale_output()
	#Index Methods#	#Moment Tests#	**mean3()**		#Polynomial Kernels#	ard_transform()
#Simultaneous Diagonalisations#	retain_all()	jarque_bera()	**cov4()**	#Limited Support Kernels#	**linear_kernel()**	linear_transform()
REG_EIG()	retain_first()	bonnett_seier()		uniform_kernel_w()	**polynomial_kernel()**	
SYM_EIG()	retain_last()	agostino_pearson()	#Robust Scatters#	triangular_kernel_w()		#Kernel Manipulations#
REG_SVD()		ansecombe_glynn()	**fastMCD()**	epanechnikov_kernel_w()	#Exponential Kernels#	kernel_centered()
SYM_SVD()	#Eigenvalue Methods#	omnibus_K2()	**FastMVE()**	quartic_kernel_w()	abelian_kernel()	kernel_normalized()
	retain_var()			triweight_kernel_w()	**laplacian_kernel()**	kernel2distance()
#Regular Diagonalisations#	kramer_rule()	#ECDF Tests#	#Local Scatters#	tricube_kernel_w()	**gaussian_kernel()**	
EIG()	cluster_priori()	anderson_darling()	**lcov()**	cosine_kernel_w()	gibbs_kernel()	#Nystrom Appoximations#
SVD()		kolmogorov_smirnov()	**rlcov()**		gamma_exponential_kernel()	nystrom_ind()
	#Component Methods#	lilliefors()	**alcov()**	#Infinite Support Kernels#	exponentiated_kernel()	nystrom_ratio()
#Alternative Diagonalisations#	normality()	cramer_mises()	**arlcov()**	gaussian_kernel_w()		
custom_eig()	pick_tsne()	watson()		logistic_kernel_w()	#Rational kernels#	#Kernel Combinations#
custom_svd()	marginal_div()		#Auxiliary Functions#	sigmoid_kernel_w()	**rational_kernel()**	kernel_sum()
reduced_svd()	joint_div()	#Misc Tests#	*opt_h()*	silverman_kernel_w()	rational_quadratic_kernel()	kernel_prod()
custom_gsvd()	batch_joint_div()	shapiro_wilk()	*breakdown()*		gamma_rational_kernel()	
		shapiro_francia()		#Aliases#		#Scale Parameter Settings#
#Rank Reducing Methods#	#t-SNE Loss Functions#	pearson_chi2()		parabolic_kernel_w()	#Periodic Kernels#	**median_trick()**
constrained_svd()	*tsne_loss()*			biweight_kernel_w()	**cosine_kernel()**	**quantile_trick()**
reduce_mat_svd()	*opt_beta()*	#Data Transformations#			**neural_network_kernel()**	
reduce_mat_qr()	*Hbeta()*	*z_score()*			periodic_kernel()	
		rob_score()				
	#Divergences#	*mad_n()*			#Misc Kernels#	
	kl_div()				fbm_kernel()	
	sym_kl_div()	#Auxiliary Functions#			gabor_kernel()	
	gen_kl_div()	*skewness_moments()*			matern_kernel()	
	renyi_div()	*kurtosis_moments()*			wiener_kernel()	
	js_div()	*round_retain_sum()*			**mahalanobis_kernel()**	
		count_sample_regions()				
		geary_kurt()				

Functions in **bold** are exported by the package.

Functions in *italics* are used as part of larger functions.

Functions with an underline have multiple definitions, for example for optional arguments.

Figure 20: Overview of all implemented functions from the SimultaneousDiagonalisation.jl package per source file (2/2).



classification.jl	evaluation.jl	external_methods.jl	figures.jl	data_manipulation.jl	utilities.jl	external_data.jl	experiments.jl
#Main Methods#	#Cross-validation#	#Visualisation Methods#	#Pairwise Scatters#	#Moment Calculations#	#Diagonal Matrix Shorthands#	#Data Retrieval#	#Thesis Reproduction#
ols2()	**k_fold()**	**tsne2()**	**scatter_plot()**	*raw_moment()*	*eye()*	iris_data()	all_experiments()
logit2()	**stratified_k_fold()**	**umap2()**	**scatter_plot_ind()**	*central_moment()*	*diag_eye()*	word2vec_data()	iris_experiments()
svm2()				*standard_moment()*	*diag_eye_nan()*	glove_data()	wine_experiments()
	#Auxiliary Functions#	#Robust Scatters from R#	#Pairwise Contour Scatters#			fasttext_data()	wbc_experiments()
#Used for Logit#	*split_data_ind()*	FastMCD2()	**contour_plot()**	#Weighted Locations#	#Matrix Shorthands#	ODDS_data()	word2vec_experiments()
loglike()	*stratified_split_data_ind()*	FastMVE2()	**contour_plot_ind()**	weighted_mean()	*self_inner()*	mnist_data()	glove_experiments()
	ind2labels()			geometric_median()	*self_outer()*		fasttext_experiments()
			#Misc Figures#	weighted_median()	*symmetrize()*	#MNIST Manipulation#	code_experiment()
	#Evaluation Metrics#		**contour_plot()**			*flatten2d()*	weighting_kernel_graph()
	diagnostics()		**heatmap_plot()**	#Distances#	#RNG Manipulation	*faltttern1d()*	
	diagnostics2D()		**component_plot()**	data2dist()	*get_seed()*	*mnist_mean()*	
	get_all_eval()		**bshape_plot()**	*mahalanobis2()*	*next_seed()*		
	mcc()			*mahalanobis1()*		#Word Embeddings#	
			#Auxiliary Functions#		#Auxiliary Functions#	word2vec_data_scratch()	
	#Auxiliary Functions#		*method_title()*	#Categorical Encodings#	*text_subscript()*	glove_data_scratch()	
	confusion_matrix()		*fast_contour_plot()*	vec_cat_encode()	*ts()*	fasttext_data_scratch()	
	reduce_mat()		*get_default_colour()*	mat_cat_encode()	*duplicates()*	*get_embedding()*	
			get_different_colour()			*get_embeddings()*	
			bshape()	#Data Transformations#			
				transform_loc()			
				transform_01()			
				transform_z()			
				transform_rob()			
				#Used for gpca/ics#			
				fix_signs!()			

Functions in **bold** are exported by the package.

Functions in *italics* are used as part of larger functions.

Functions with an underline have multiple definitions, for example for optional arguments.

B Wine Data: Visualisation of Multiple Pairs

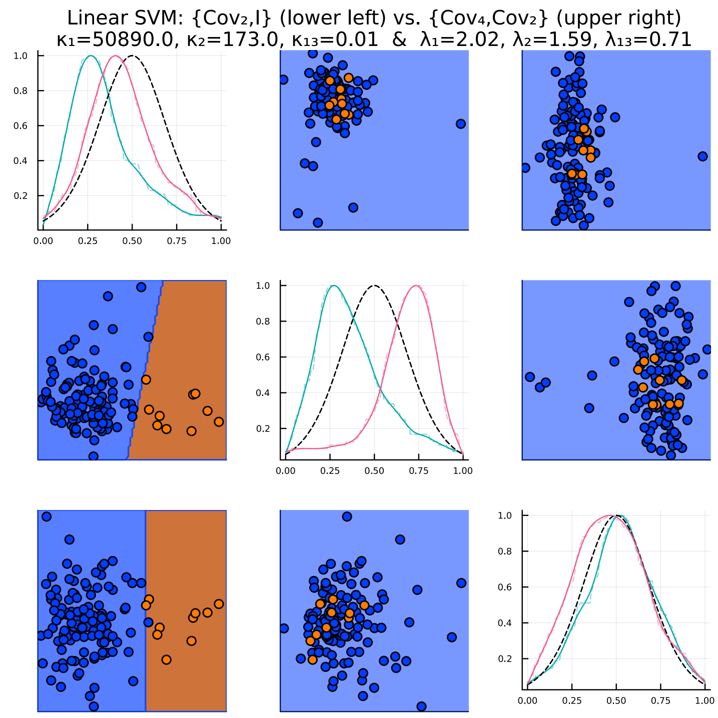


Figure 21: First, second and last components of $\{\mathrm{Cov}_2,\mathrm{I}\}$ and $\{\mathrm{Cov}_4,\mathrm{Cov}_2\}$ pairs.

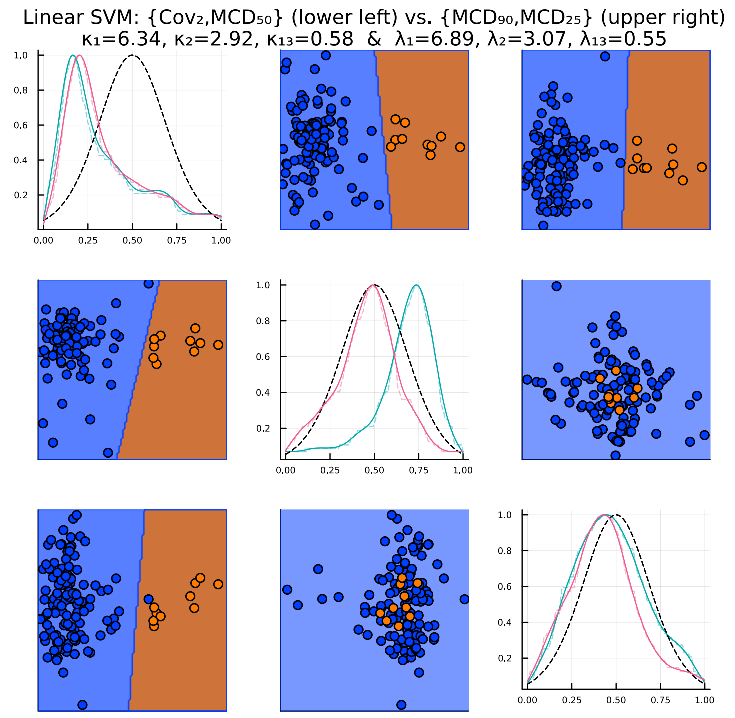


Figure 22: First, second and last components of $\{\mathrm{Cov}_2,\mathrm{MCD}_{50}\}$ and $\{\mathrm{MCD}_{90},\mathrm{MCD}_{25}\}$ pairs.

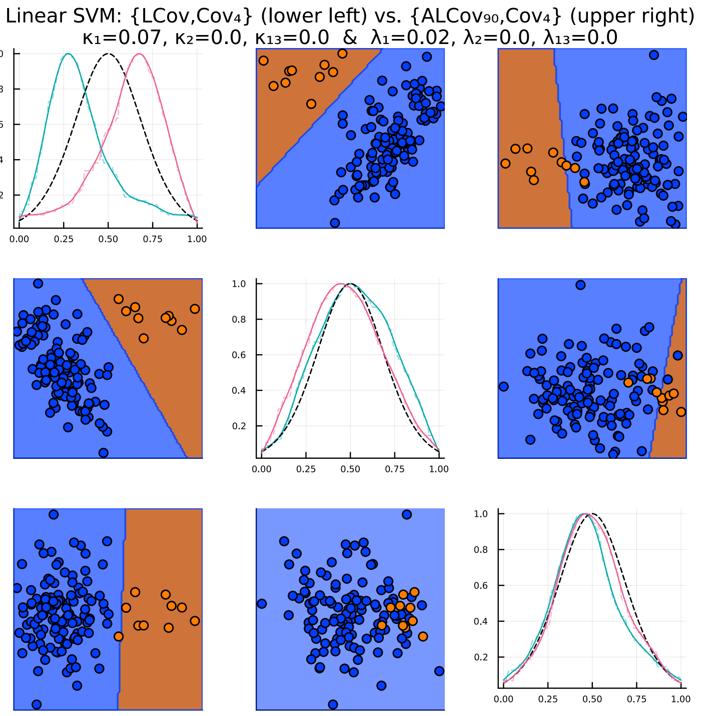


Figure 23: First, second and last components of {LCov,Cov$_4$} and {ALCov$_{90}$,Cov$_4$} pairs, making use of the Gaussian kernel.

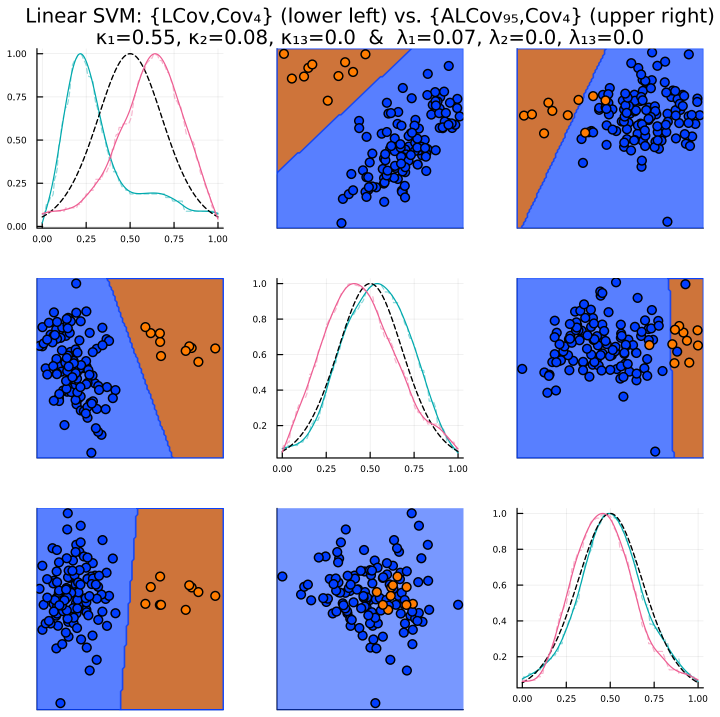
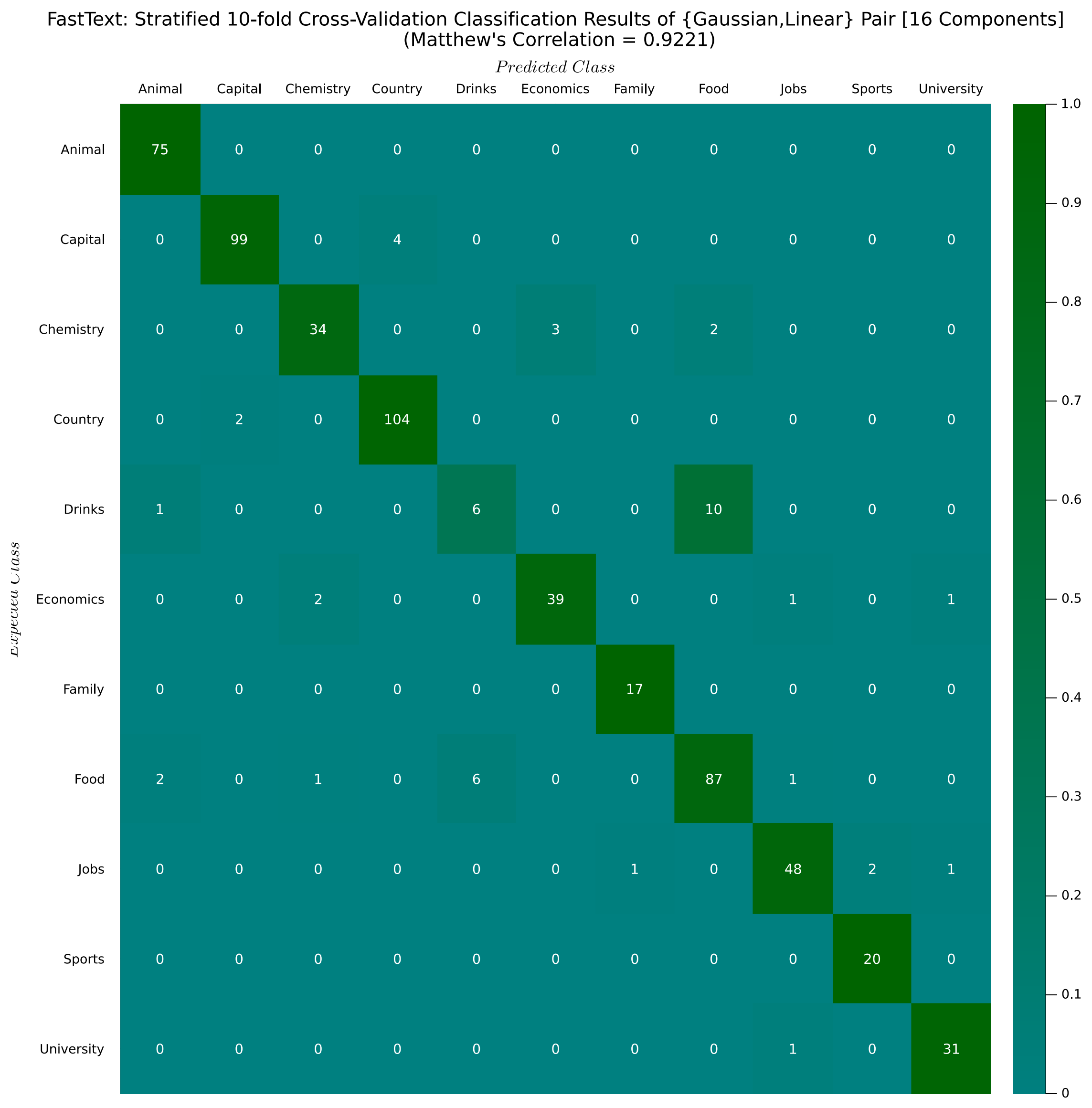


Figure 24: First, second and last components of {LCov,Cov$_4$} and {ALCov$_{95}$,Cov$_4$} pairs, making use of the Epanechnikov kernel.

C FastText: Detailed Classification Table



	Animal	Capital	Chemistry	Country	Drinks	Economics	Family	Food	Jobs	Sports	University
Animal	75	0	0	0	0	0	0	0	0	0	0
Capital	0	99	0	4	0	0	0	0	0	0	0
Chemistry	0	0	34	0	0	3	0	2	0	0	0
Country	0	2	0	104	0	0	0	0	0	0	0
Drinks	1	0	0	0	6	0	0	10	0	0	0
Economics	0	0	2	0	0	39	0	0	1	0	1
Family	0	0	0	0	0	0	17	0	0	0	0
Food	2	0	1	0	6	0	0	87	1	0	0
Jobs	0	0	0	0	0	0	1	0	48	2	1
Sports	0	0	0	0	0	0	0	0	0	20	0
University	0	0	0	0	0	0	0	0	1	0	31

Figure 25: Overview of best FastText classification results.