Kernel PCA and the Nyström method

Thesis submitted in partial fulfilment

of the requirements for the degree of

Doctor of Philosophy

27 December 2021

Fredrik Hallgren

Department of Statistical Science

University College London

Declaration

I, Fredrik Hallgren confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

This thesis treats kernel PCA and the Nyström method. We present a novel incremental algorithm for calculation of kernel PCA, which we extend to incremental calculation of the Nyström approximation. We suggest a new data-dependent method to select the number of data points to include in the Nyström subset, and create a statistical hypothesis test for the same purpose. We further present a cross-validation procedure for kernel PCA to select the number of principal components to retain. Finally, we derive kernel PCA with the Nyström method in line with linear PCA and study its statistical accuracy through a confidence bound.

Impact statement

This thesis presents several novel results on kernel principal components analysis (kernel PCA) and the Nyström method, both of which have received considerable interest in the statistics and machine learning literature.

The subsequently presented incremental algorithm for kernel PCA is the most computationally efficient such algorithm in existence, to the best of our knowledge. Incremental algorithms can be highly beneficial, for increased time efficiency in the streaming data setting, and increased memory efficiency in general.

The last part of this thesis for the first time combines kernel PCA with the Nyström method in line with linear PCA, thus providing a new efficient method for non-linear PCA. The supplied confidence bound allows for measuring the statistical accuracy of the method.

The paper version of the incremental algorithm is freely available online at `www.arxiv.org`. It has received a few citations from peer-reviewed research papers since being made public, and has led to reviewing opportunities in related areas for *IEEE Transactions on Signal Processing*. The paper version of the new method for efficient non-linear PCA is also available online. Most of the computer code implemented as part of this research is also freely available at `www.github.com`. Since made public, several people have used the code and asked questions about how it works.

The research has been communicated in a number of ways. I have presented the work to other PhD students and lecturers in the Department of Statistical Science. I have discussed the research with students and lecturers both within and outside of UCL, including when I attended the COLT and ICML conferences in 2018.

Contents

List of Tables

List of Figures

Acknowledgements

Many people have helped and supported me during the work on this thesis. Special thanks to my supervisor *Paul Northrop*, for his excellent guidance and research input, and to my secondary supervisor *Ricardo Silva*. I am also grateful to *Arthur Gretton* for valuable comments on Chapters 1 to 3 during my upgrade viva and to *Michael Arbel* for helpful discussions around Chapters 4 and 6. I would also like to express my sincere gratitude to my thesis examiners *John Shawe-Taylor* and *Dino Sejdinovic*.

Chapter 1

Introduction

Kernel methods discover non-linear patterns in data whilst being able to use linear solution methods, through a non-linear transformation of data into an often high-dimensional feature space where linear methods can be applied [Shawe-Taylor and Cristianini, 2004]. Through a practically arbitrary transformation of the input variables into a Hilbert space they allow for very flexible representations of data whilst providing a precise mathematical framework for statistical analyses. They are closely related to many other areas in statistics and machine learning and the study of kernel methods can provide valuable insights into related methods. A few examples of these are Gaussian processes, functional data analysis, kriging and metric learning [Rasmussen and Williams, 2006, Wild et al., 2021, Ramsay and Silverman, 2005, Cressie, 1990, Dong et al., 2019].

The explicit representations of data points in the feature space are generally not available, but instead one uses the fact that the inner products in this space correspond to the application of a positive definite kernel function between pairs of data points from the original input space. Many kernel methods have been conceived as the application of well-known linear methods in this feature space,

often reformulated to be expressed entirely in the form of inner products [Mika et al., 1999, Rosipal and Trejo, 2001, Bach and Jordan, 2002, Harmeling et al., 2003, Singh et al., 2019, Chau et al., 2021]

Early research within kernel methods largely focused on thus devising new non-linear versions of classical methods. Other strains of research have been concerned with measuring the statistical accuracy of different kernel methods, or developing statistical tests based on feature space data representations, and yet another area of intense research activity has been to improve the scalability of kernel methods, encumbered by the necessity to evaluate the kernel function between all pairs of data points [Shawe-Taylor and Williams, 2003, Zhang et al., 2018, Rindt et al., 2021, Fernández et al., 2021]. In this thesis we attempt to present contributions to all four of these research areas.

Another example of a classical statistical method adapted to be used with kernels is kernel principal component analysis (kernel PCA), obtained through the application of linear PCA in feature space [Schölkopf et al., 1998]. PCA is a ubiquitous method to discover the most important directions of variation in data [Pearson, 1901] and may be used for dimensionality reduction, exploratory data analysis, anomaly detection, discriminant analysis, clustering, or as a general preprocessing step for regression or classification [Jolliffe, 2002]. Kernel PCA has been shown to outperform linear PCA in a number of applications [Chin and Suter, 2007].

Kernel PCA is computed through the eigendecomposition of the kernel matrix instead of decomposing the covariance matrix from the original data. This amplifies the scalability issues of kernel methods since the eigendecomposition is much more expensive to compute than solving a linear system or computing the inverse of a matrix. State-of-the-art algorithms to find the full eigendecomposition of a matrix require around 30 times more floating-point operations (flops) than

solving the equivalent system of linear equations [Golub and Van Loan, 2013].

Many methods introduced to improve the scalability of kernel methods rely on approximations to the kernel matrix or to the explicit feature maps. One popular approximate method is the Nyström method [Nyström, 1930, Williams and Seeger, 2001], which creates a low-rank approximation to the kernel matrix through a randomly sampled subset of data examples.

In this thesis we attempt to contribute to the understanding of kernel PCA, including when used together with the Nyström method. We provide an incremental algorithm for calculation of kernel PCA, develop a statistical test for determining the size of the Nyström subset, present a cross-validation procedure for kernel PCA for selecting the number of principal components to retain and derive kernel PCA with the Nyström method, including an analysis of its statistical accuracy.

1.0.1 Main contributions

A summary of the contributions presented in this thesis that are potentially of the most interest is as follows

- The most efficient incremental algorithm for kernel PCA (*Chapter 3*)[1]
- A principled way to select the size of the Nyström subset with an F-test (*Chapter 4*)
- A cross-validation procedure for kernel PCA to select the number of principal components (*Chapter 5*)
- Making kernel PCA more scalable by deriving the Nyström method for kernel PCA (*Chapter 6*)

[1] Since completion of this thesis a more efficient algorithm for the uncentred case which is partly based on ours has been published in *IEEE Transactions on Signal Processing* Hall et al. [2022]

- Analyzing the statistical accuracy of the proposed method for kernel PCA with the Nyström method (*Chapter 6*)

1.0.2 Auxiliary contributions

A number of contributions have also followed from the main contribution described above, including

- An incremental algorithm for the Nyström approximation (*Chapter 3*)
- Reversing the incremental algorithm for kernel PCA to remove individual data points from the kernel PCA solution (*Chapter 3*)
- Incremental calculation of the kernel PCA reconstruction error, applied to select the size and composition of the Nyström subset (*Chapter 3*)
- A data model where the Nyström method can be considered the true representation for the data (*Chapter 4*)
- Kernel PCR with the Nyström method (*Chapter 6*)
- Novel specification for standard kernel PCR with centred regressors (*Chapter 6*)
- A majorization relation comparing Nyström kernel PCA to kernel PCA created directly from the Nyström subset (*Chapter 6*)
- Sharper versions of concentration results from previous literature (*Chapter 6*)

1.1 MOTIVATION

The research carried out in this thesis has been motivated by the widespread use and applicability of the methods considered, both in industry and academia, and by the lack of a complete theoretical and algorithmic understanding of those methods.

The report is dedicated to kernel PCA, which is the application of linear PCA after constructing a set of higher-dimensional descriptive features through a mapping $\phi(x)$. Linear PCA was invented in 1901 by Karl Pearson, the founder of UCL's Department of Statistical Science, and has since become ubiquitous in applications. For example, PCA is a standard method used for risk management in the financial industry, where the top principal components of an investment portfolio are treated as risk factors that need to be controlled. It is applied to survey data, to identify underlying explanations that drive responses to survey questions. It is an important tool in psychology research to identify independent dimensions of human behaviour.

Kernel PCA is a generalization of linear PCA that includes linear PCA as a special case, by choosing the feature mapping $\phi(x)$ to be the identity. Kernel PCA can often be used as a drop-in replacement for linear PCA, with the added flexibility of the choice of a near arbitrary feature mapping $\phi(x)$. In experimental analyses kernel PCA has also been shown to outperform linear PCA in a number of settings [Chin and Suter, 2007]. It has been fruitfully applied to many real-world problems, including in biology [Shiokawa et al., 2018], geology [Vo and Durlofsky, 2016], image analysis [Wang, 2012] and medicine [Widjaja et al., 2012].

We wished to introduce an efficient incremental procedure for kernel PCA, for improved time and memory performance of its calculation. Incremental

algorithms often have better memory performance than batch computation, and improved time performance when data arrives in a stream and a solution is to be calculated for each additional data point. We verified the time and memory performance for our algorithm in relation to other similar algorithms.

Incremental kernel PCA has been applied for example to image analysis [Chin et al., 2006] and may also be considered for any problem where kernel PCA is known to be useful, if data arrives sequentially or improved memory efficiency is desired.

We further wished to introduce a general incremental procedure for the Nyström approximation, with the same advantages generally expected of incremental algorithms. Such an algorithm was lacking in the existing literature; an incremental procedure only existed for the case of regression.

Approximate methods introduce an additional hyperparameter, which governs the tradeoff between computational requirements and statistical accuracy. A fairly limited amount of work has been dedicated to the determination of this hyperparameter and we also wished to make contributions to this area. Theoretical results on the approximation accuracy of the Nyström method as a function of this hyperparameter existed and could be applied in the determination of this hyperparameter, but methods adapted to the specific dataset at hand were less readily available. This was the inspiration for our empirical criterion for choosing the number of data points used in the subset for creating the Nyström approximation, and the statistical test developed for the same purpose.

The Nyström method is one of the most popular ways in which to reduce the computational requirements of kernel methods. However, application of the Nyström method to kernel PCA in line with linear PCA is not immediate. To improve scalability for kernel PCA we derived it with the Nyström method in-

cluding all quantities of interest for PCA. We also studied the statistical accuracy of the proposed method, in particular through a confidence interval that allows for measuring the performance of the method for a particular dataset.

1.2 INCREMENTAL KERNEL PCA

Incremental algorithms, where an existing statistical model is updated for additional data examples, are often desirable. If data arrives sequentially in time and a solution is required for each additional data example, more efficient incremental algorithms are often available than repeated application of a batch procedure. Furthermore, incremental algorithms often have a lower memory footprint than their batch counterparts.

We propose a novel algorithm for incremental kernel PCA, which accounts for the change in mean from each additional data example. It works by writing the expanded mean-adjusted kernel matrix from an additional data point in terms of a number of rank one updates, to which a rank one update algorithm for the eigendecomposition can be applied. We use a rank one update algorithm based on work in Golub [1973] and Bunch et al. [1978].

A few previous exact incremental algorithms for kernel PCA have been proposed, some of which are based on the application of an incremental linear PCA method in feature space [Kim et al., 2005, Chin and Suter, 2007, Hoegaerts et al., 2007]. Rank one update algorithms for the eigendecomposition have not previously been applied to kernel PCA, to the best of our knowledge. If the mean of the feature vectors is not adjusted, our algorithm corresponds to an incremental procedure for the eigendecomposition of the kernel matrix, which can be more widely applied.

Our algorithm has the same time and memory complexities as existing algorithms

for incremental kernel PCA and it is more computationally efficient than the comparable algorithm in Chin and Suter [2007], which also allows for a change in mean. Furthermore, it can be considered more flexible, since it is straightforward to apply a different rank one update algorithm to the one we have used, for potentially improved efficiency. Approximate algorithms could also be applied, for example from randomized linear algebra [Mahoney, 2011].

Applying our incremental procedure in reverse, one obtains a decremental algorithm for removing data points from the kernel PCA solution, instead of adding them. A potential use case for decremental learning is leave-one-out cross-validation as detailed in Mertens et al. [1995] and Cauwenberghs and Poggio [2001].

We also extend our algorithm for incremental kernel PCA to incremental calculation of the Nyström approximation to the kernel matrix. We incrementally add data examples to the subset used to create the Nyström approximation to kernel PCA. This allows one to evaluate empirically the accuracy of the Nyström approximation for each added data example. Rudi et al. [2015] presented an incremental updating procedure for the Nyström approximation to kernel ridge regression, based on rank one updates to the Cholesky decomposition. Our proposed incremental procedure can be applied to any kernel method requiring the eigendecomposition or inverse of the kernel matrix. Combining an incremental algorithm with the Nyström method also leads to further improvements in memory efficiency, compared with either method on its own.

Approximate kernel methods introduce a hyperparameter that governs their accuracy, where there is a trade-off between the statistical accuracy of the solution and the time and memory requirements of the algorithm. This hyperparameter may for example be the rank of a low-rank approximation to the kernel matrix or the covariance matrix. The choice of this hyperparameter has not been extensively

studied in the literature, and we have aimed to contribute to this area. Various bounds on the statistical accuracy of different approximate kernel methods have been proposed, which can be used to guide the choice of the hyperparameter. However, these bounds may fail to adapt to the specific subset obtained.

We instead update the approximate solution sequentially and empirically evaluate when sufficient accuracy have been achieved, using the reconstruction error of an additional data point as a criterion for evaluating when a sufficient number of data examples have been included in the Nyström subset. The use of the reconstruction error in this fashion may be motivated by its use as a measure of influence in robust statistics. When additional observations cease to influence the solution, the current subset can be concluded to provide a good representation of the full dataset.

1.3 STATISTICAL TESTING OF THE NYSTRÖM METHOD

We also develop a statistical test to select the number of data points to include in the Nyström subset. Under the null hypothesis, the Nyström method is considered the true model for the data, and if it is rejected then an additional data point should be added to the subset. The test allows for taking a formal decision on when to expand the Nyström subset, which we will know is only the wrong decision with a small probability corresponding to the chosen significance level.

1.4 CROSS-VALIDATION FOR KERNEL PCA

Leave-one-out cross-validation has been used for selecting the number of principal components to retain for linear PCA. In order to apply our decremental algorithm for this purpose, we present a cross-validation procedure for kernel

PCA. It is based on an equivalent algorithm for linear PCA, adapted to be applied in the feature space. It appears to be the first cross-validation procedure that has been presented in the literature specifically for kernel PCA for the purposes of selecting the number of principal components to retain. Resampling estimates for the reconstruction error of kernel PCA were also studied in Opper [2006].

1.5 KERNEL PCA WITH THE NYSTRÖM METHOD

The Nyström approximation is a proven method for increasing the scalability of kernel methods. However, the approximate eigenvectors and eigenvalues from the original Nyström method do not define true PCA as it is typically defined, where the scores are uncorrelated and orthogonal and where the eigenvalues measure the maximum variance captured by the principal components in turn. The eigenvectors of the original Nyström approximation are not orthogonal and the eigenvalues do not measure the variance captured by the principal components, since these are simply the eigenvalues of the kernel matrix from the subset of data points scaled by the fraction of data points in the subset.

In the last part of this thesis we derive kernel PCA with the Nyström method in line with linear PCA. Without an assumption of zero-mean data we provide orthonormal principal components, uncorrelated principal scores, the explained variance with respect to the principal components, and the reconstruction error of the full data on the PCA subspace. The principal scores are new representations of the data points and allows for the method to be used as a preprocessing step before applying supervised learning techniques.

We further study the statistical accuracy of the proposed method, first through a majorization relation and a study of the special case when the number of subsampled data points equals the PCA dimension, and then through a finite-sample confidence bound on the empirical reconstruction error. For the latter, we

show that with high probability the difference between the Nyström and standard empirical reconstruction errors is less than a data-dependent quantity, which is a function of the eigenvalues of the kernel matrix from the subset of data points K_{mm}, the maximum value of the kernel function $\sup_x k(x, x)$, the size of the subset m and the total number of data points n. The bound does not require that we have observed the entire dataset, only the subset of data points. In line with all results on the accuracy of kernel PCA it assumes that data has zero mean in feature space.

To demonstrate the use of the method as a preprocessing step before applying other methods we apply it to the regression problem, presenting kernel principal component regression with the Nyström method. The derivation also leads to a novel specification for standard kernel PCR.

As a corollary to the confidence bound one may also deduce sharper versions of some concentration inequalities from previous literature based on the norm of the difference between positive operators.

1.6 OUTLINE

The document is structured as follows. In Chapter 2 we go through some background on kernel methods, PCA and other relevant areas. In Chapter 3 we present our algorithm for incremental kernel PCA and extend this to incremental updating of the Nyström approximation, as well as present our empirical evaluation criterion. In Chapter 4 we develop the hypothesis test for selecting the size of the Nyström subset. In Chapter 5 we detail our procedure for cross-validation of kernel PCA. We present the new method combining kernel PCA with the Nyström method and its associated statistical investigation in Chapter 6 and conclude with some parting thoughts in Chapter 7.

1.7 NOTATION

Matrices and operators will be denoted by upper-case characters and vectors in $\mathbb{R}^d$ by lower-case characters, occasionally in a bold typeface. Upper-case characters will also be used for random variables, apart from occasionally when they represent data points before they are observed. All Euclidean vectors are column vectors, unless explicitly stated otherwise. Parameters fitted to data are often denoted by letters from the Greek alphabet. The symbols X, Y or Z will be used for a generic random variable, the symbols T or L for a generic operator and the symbol M for a generic matrix.

Let M be a matrix and v a vector in $\mathbb{R}^d$. Then $(M)_{i,j}$ refers to the element in the ith row and jth column of M, $(v)_i$ refers to element i of v. $(M)_{i:j,k:l}$ denotes the submatrix of M containing rows i to j, inclusive, and columns k to l, inclusive, and $(v)_{i:j}$ denotes a vector containing the elements i to j, inclusive, of the vector v. The notation $M_{(i)}$ refers to the ith row of M, and $M^{(j)}$ denotes its jth column. The first element, row or column has index 1. The symbol $\mathbb{1}_n$ denotes an $n \times n$ matrix with each element equal to $1/n$ and $\mathbb{1}_n^{(p\times q)}$ is a $p \times q$ matrix with each element equal to $1/n$. The symbols $\mathbf{1}_n, \mathbf{0}_n$ denote length n vectors of ones and zeros respectively. A row vector with elements $v_1, v_2, ..., v_d$ will be written $(v_1, v_2, ..., v_d)$ or $[\ v_1\ \ v_2\ ...\ v_d\ ]$ and $[\ v_1\ ;\ v_2\ ]$ is a column vector with elements v_1, v_2. The transpose of a vector or matrix is v^T. The arithmetic mean of vector is denoted $\bar{v}$.

The blackboard letter $\mathbb{E}$ denotes expectation and $\|\cdot\|$ denotes a norm. If a specific norm is not specified it is taken to mean the Euclidean 2-norm or the operator norm, depending on the context. An inner product is denoted by $\langle\cdot,\cdot\rangle$, or $\langle\cdot,\cdot\rangle_E$ for a specific inner product space E.

Estimated quantities depending on the empirical distribution $\mathbb{P}_n$ rather than

$\mathbb{P}$ will often be denoted by $\hat{\cdot}$, approximations by $\widetilde{\cdot}$ and centred quantities by $\cdot'$. Empirical quantities may be superscripted or subscripted by the number of observations used in the estimate. The probability density function of a measure $\mathbb{P}_Y$ will be denoted by $p_Y(y)$.

The linear span of the vectors in a set A is written $\mathrm{span}\{A\}$ or $\langle A \rangle$. The dimension of a space V (the cardinality of a basis for the space) is written $\dim(V)$. The symbol $\mathcal{O}(\cdot)$ denotes Big-O notation. The function $\lambda_j(\cdot)$ returns the jth eigenvalue, in decreasing order, of its argument, and the symbol $\lambda_{<d}$ denotes the sum of the largest d eigenvalues $\lambda_1, \lambda_2, ..., \lambda_d$.

If v is majorized by u we write $v \succ u$. The symbol $:=$ denotes the introduction of new notation, i.e. $a := b$ means that b will be denoted by a, and vice versa for $a =: b$. The binary operators $\vee$ and $\wedge$ are defined as $a \vee b = \max\{a, b\}$ and $a \wedge b = \min\{a, b\}$. For a Banach space $\mathcal{B}$, we let $\mathcal{B}^*$ denote the dual space of bounded linear functionals on $\mathcal{B}$. For an operator T, we let T^* denote its adjoint. The image of an operator is $\mathrm{Im}(T)$ and its null space (also called its kernel) is $\mathrm{Ker}(T)$.

1.8 NUMERICAL EXPERIMENTS

Throughout this thesis we present the results of a number of computer experiments on our incremental algorithms for kernel PCA and the Nyström method, on our empirical criterion for selecting the number of data points in the subset, on our cross-validation procedure for kernel PCA and on our method for kernel PCA with the Nyström method. All the experiments are implemented in the Python programming language. Most of the software code is publicly available at `https://github.com/fredhallgren`.

Dataset	Data size	Number of attributes
magic	19020	11
yeast	1484	8
cardiotocography	2126	23
segmentation	2310	19
drug	1885	32
digits	5620	64
dailykos	3430	6906
neurips	1500	12419
airfoil	1503	7

Table 1.1: Datasets used

1.8.1 Datasets

In the experiments we use a selection of nine different datasets from the UCI Machine Learning Repository [Dua and Graff, 2017]. Dimensionality reduction can be particularly important for high-dimensional data, so we include a number of such datasets. We use the simulated `magic` gamma telescope dataset, the `yeast` dataset, containing cellular protein location sites for fungi, the `cardicotocography` dataset, with heart measurements, the `segmentation` dataset containing various data on images, the `drug` dataset with personality traits and drug consumption, the `digits` dataset with flattened 8×8 pixel grayscale images, two bag-of-words datasets with bag-of-words vectors of articles from `www.dailykos.com` and NeurIPS papers, respectively, and the `airfoil` dataset which describes aerodynamic tests of blades in a wind tunnel from NASA. We tabulate some information on the datasets in Table 1.1.

1.8.2 Kernel functions

In this section we describe a few different kernel functions. For the incremental experiments we use the the radial basis functions (RBF) kernel, for the cross-

Kernel	**Functional form** $k(x, y)$	**Parameters**
RBF	$\exp\{-\|x-y\|^2/\sigma\}$	$\sigma \in \mathbb{R}_+ \setminus \{0\}$
Polynomial	$(\langle x, y\rangle + R)^d$	$R \in \mathbb{R},\ d \in \mathbb{N}$
Cauchy	$\frac{1}{1+\|x-y\|^2/\sigma^2}$	$\sigma \in \mathbb{R}_+ \setminus \{0\}$

Table 1.2: Kernel functions used

validation experiments we use both the RBF and the polynomial kernel, and for the confidence bound experiments we in addition use the Cauchy kernel. These kernel functions are defined in Table 1.2.

The radial basis functions kernel has one hyperparameter σ, sometimes termed the *bandwidth*, which can take any value on the positive real line and which governs the magnitude of the inner product as a function of the distance between pairs of data points. For low values of σ, only data examples that are close together in Euclidean distance will have inner products that are very different from zero. If σ has a large value, then there is less difference in inner product between data examples that are far away or close to each other.

The polynomial kernel creates a polynomial of the inner product between two data points $\langle x, y\rangle$, given by

$$(\langle x, y\rangle + R)^d = \sum_{k=0}^{d} \binom{d}{k} R^{d-k} \langle x, y\rangle^k$$

by the binomial theorem. It has two hyperparameters. The dimension d determines the maximum degree of the polynomial and the constant R determines the relative weights of different powers of $\langle x, y\rangle$. If $R > 1$ then more weight will be given to smaller powers of $\langle x, y\rangle$, whilst if $R \in (0, 1)$ then more weight will be

given to larger powers. If $R < 0$ then the kernel is not positive definite.

If the dimension $d = 1$ and $R = 0$ we recover standard linear PCA since the feature map in this case is the identity mapping. For this reason, kernel PCA can be seen as a generalization of linear PCA, and any kernel method as a generalization of the equivalent linear method.

The Cauchy kernel can be used as an alternative to the RBF kernel when one wants to capture long-range dependencies. It has the same hyperparameter, i.e. the bandwidth σ.

Certain heuristics are available when choosing values for the hyperparameters of the kernel function. For example, one popular heuristic is to set the bandwidth of the radial basis functions kernel to be the median distance between pairs of data points, either calculated across the whole dataset or across a subset of the dataset [Garreau et al., 2017].

There is less applicability for heuristics for the hyperparameters of the polynomial kernel, and data exploration or a practitioner's insight into the behaviour of the data can be considered more important [Tukey, 1977].

Kernel functions that are not positive definite are sometimes used in practice (e.g. Schölkopf [2001]). In this case the kernel does not correspond to an inner product of feature mappings and the kernel matrix is not guaranteed to be symmetric positive definite.

Chapter 2

Background

In this chapter we go through background theory and previous literature relevant to the results presented later. In the first section we give an overview of estimation theory, on which much of the subsequent work rests. In the following section we describe the computational framework based on which the time and memory performance of our algorithms are calculated. We next describe subspace learning of which principal components analysis is an example – we contrast principal components analysis to several other subspace learning techniques. We devote a large part of the background to kernel methods, explaining their derivation and providing examples of kernel methods, including kernel ridge regression and kernel SVM, possibly the most popular kernel methods, and kernel PCA. We further outline a number of ways to make kernel methods scalable, with a particular focus placed on the Nyström method. We then give a brief overview of incremental learning, to provide the relevant context for our incremental algorithms, and finally describe two resampling methods that appear later, cross-validation and the bootstrap.

2.1 THEORY OF ESTIMATION

In this section we present general background on estimation theory, which provides the foundation on which much of the work in this thesis rests, including the statistical test in Chapter 4, the cross-validation procedure in Chapter 5, and the new method for kernel PCA with the Nyström method and the study of its statistical accuracy in Chapter 6.

Statistics and related disciplines, such as statistical learning, is concerned with the drawing of conclusions from data. The scientific foundation rests on mathematics and probability theory, but with the introduction of data points that are generated from some probability distribution. The essential problem that statistics attempts to solve is to determine the probability distribution, or functions thereof, from the information conveyed by the data points.

In statistics we thus have a set of n data points $\{x_i\}_{i=1}^n$ generated by probability distributions according to $x_i \sim p_i(x|x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n) := p_i(x|x^{(-i)})$. Oftentimes, the data are assumed to be identically distributed, i.e. they are generated from the same probability distribution, $x_i \sim p(x|x^{(-i)})$, and/or independent, $x_i \sim p_i(x)$. If the data is generated independently from the same distribution it is termed iid.

The x_i belong to some set $\mathcal{X}$, often taken to be $\mathbb{R}^d$ or a subset of $\mathbb{R}^d$. In the rest of this section we will assume that $\mathcal{X} \subseteq \mathbb{R}^d$, but later we allow $\mathcal{X}$ to be any set, including a reproducing kernel Hilbert space $\mathcal{H}$. In parametric statistics the probability distribution is parametrized by a real vector θ with dimension D, written $p_\theta(x)$ [Sen et al., 2010]. In non-parametric statistics the object of interest is instead some functional F of a probability distribution that can not be succintly parametrized.

Each x_i is a realization of a real-valued random variable X_i on some measure

space $(\Omega, \mathcal{A}, \mathbb{P})$ where $X_i : \Omega \to \mathbb{R}^d$ [Cohn, 1980]. In the iid case, each data point is often an independent realization from the same random variable X.

A common assumption in statistics is thus that the random variable X admits a distribution, in other words that

$$\mathbb{P}(A) = \int_A p(x) d\mu(x)$$

for all $A \in \mathcal{A}$ where μ is the Lebesgue measure. The function $p(x)$ is called the Radon-Nikodym derivative of $\mathbb{P}$ and is also denoted by [Tankov and Touzi, 2010]

$$p(x) = \frac{d\mathbb{P}}{d\mu(x)}$$

The Radon-Nikodym derivative exists if $\mathbb{P} \ll \mu$, i.e. if $\mathbb{P}$ is absolutely continuous with respect to the Lebesgue measure [Cohn, 1980]. The function $p(x)$ is unique apart from possibly on sets of measure zero.

Note that in the above the measures $\mathbb{P}$ and μ are defined on the same space $\Omega = \mathcal{X}$, while random variables often take values in a space $\mathcal{X}$ which is differnet from Ω. However, the distinction of whether $\mathcal{X}$ is different from Ω is unimportant, since X is always assumed measurable and if μ is a measure on Ω there is a measure λ on $\mathcal{X}$ for which $\lambda(B) = \mu(A) \;\; \forall A \in \mathcal{A}$, where $A = X^{-1}(B)$.

A common assumption is that the random variable X is square-integrable, i.e.

$$\int_\Omega X^2 d\mathbb{P} < +\infty$$

This is a critical assumption in probability theory, and one on which many important theorems depend, such as the Central Limit Theorem and the Berry-Esséen theorem [Graham and Talay, 2011].

The fundamental problem to which statistical inference is dedicated is to estimate the parameters θ of $p_\theta(x)$ by calculating some function of the data $T_n = T_n(x_1, x_2, ..., x_n)$, termed an *estimator*. When calculating the estimator we thus fix the data and consider $p_\theta(x)$ as a function of θ. The task is then to determine a suitable function T_n, with the hope that most often we would have $T_n \approx \theta$.

Many different techniques are available to determine a suitable value of the function T_n, for example the least squares, method of moments and maximum likelihood estimation methods.

2.1.1 Statistical decision theory

Statistical decision theory is a general framework for arriving at decisions based on random data [Berger, 2013]. Many other methods of estimation can be framed as statistical decision problems. In addition to an unknown parameter $\theta \in \Theta$ we have a set of possible actions to take $a \in \mathcal{A}$ and a loss function $L(\theta, a)$. The goal is to select an action a such that the expected loss is minimized. Actions are taken by determining a decision rule δ that maps from the data to the set of possible actions $a = \delta(X)$. The expected loss is also termed the *risk* and is given by

$$R(\theta, \delta) = \mathbb{E}\left[L(\theta, \delta(X))\right]$$

For estimation problems, the action corresponds to an estimated parameter value $\hat{\theta}$ and the decision rule corresponds to the estimator $T_n(X)$. Determining unknown quantities by minimizing an expected loss between true and estimated values is also called *empirical risk minimization*.

In some situations one wishes to take random actions, instead of taking a specific action for each value of the data x. In this case one specifies a probability distribution at each point $x \in \mathcal{X}$ and $\delta = \delta(x, a)$ is a probability density of a for

each fixed x. To calculate the risk $R(\theta, \delta)$ one takes the expectation over both the distribution of the data X and the distributions $\delta(x, y)$.

The minimax principle

Instead of determining the decision rule such that the expected loss is minimized, one can pick the decision rule that is best in the worst-case scenario,

$$\hat{\delta} = \inf_{\delta} \sup_{\theta} L(\theta, \delta)$$

This is called the minimax decision principle [Berger, 2013]. The decision rule is often randomized, in which case one takes the expectation of the loss with respect to the distributions $\delta(x, y)$.

Most often it will be most sensible to minimize the loss on the average, but one situation in which the minimax principle can be considered a more sensible approach is when the state of nature (parameter) θ is not determined randomly, but by an intelligent adversary that after observing our choice of decision rule picks the worst state of nature for that decision rule. Solving minimax decision problems often involves game theory [Myerson, 2013].

2.1.2 The invariance principle

The invariance principle for statistical estmation is different from other estimation methodologies, in that it does not result in a maximization or minimization problem. Loosely described, it specifies that parameters should be determined such that they are invariant to a set of specified transformations of the input data.

More formally, a set of densities $\{p(x \mid \theta)\}_{\theta \in \Theta}$ is said to be invariant under transformations by a group G if for every $g \in G$ and $\theta \in \Theta$ there is a unique $\theta^* \in \Theta$ such that $p(g(x) \mid \theta^*) = p(x \mid \theta)$ [Hungerford, 2003, Berger, 2013].

Similarly, a decision rule is invariant if, for all $x \in \mathcal{X}$ and $g \in G$

$$\delta(g(x)) = \tilde{g}(\delta(x))$$

for some $\tilde{g}$ in a group $\tilde{G}$ that is isomorphic to G.

Often the invariance principle does not lead a unique set of values for the parameters of interest, but rather limits the search space. Specific parameter values can then be determined by applying another estimation methodology on the restricted search space. For example, the *Minimum Risk Equivariant* (MRE) estimator of a parameter θ is obtained by minimizing the risk subject to the decision rule being invariant to some specified transformations.

2.1.3 Limit theorems

In this section we review a few limit theorems for random variables. Limit theorems treat the behaviour of sequences of random variables, as the number of random variables in the sequence tends to infinity. For a sequence of random variables $\{X_n\}_n$, they investigate whether

$$X_n \xrightarrow[n\to\infty]{} C$$

for some constant C, for some mode of convergence, or whether

$$X_n \xrightarrow[n\to\infty]{d} X$$

for some random variable X, where d indicates convergence in distribution [Gut, 2013]. The most well-known examples of the former and latter are the Law of Large Numbers (LNN) and the Central Limit Theorem (CLT), respectively [Gut, 2013]. Another important example of a limit theorem is the Law of the Iterated Logarithm.

2.1.4 Probability inequalities

Concentration inequalities for probabilites are an essential tool to measure the tails of random variables when their full distributions are unknown, as is often the case, and when the extreme values are of primary interest. We use several concentration inequalities in the proof of the confidence bound on our method for kernel PCA with the Nyström method, including Hoeffding's inequality in Banach spaces and inequalities based on Hilbert space theory. Not all of these are stated here, we refer to the proof for further details.

The random variables in question might be part of a sequence of random variables, and the bounds may depend on the index of a variable in the sequence. This is the case that we will be most interested in. Throughout this section any sequence of random variables will be iid, unless stated otherwise.

Markov's inequality

We have that if g is a non-decreasing positive function on $x > 0$ with $\mathbb{E}[g(|X|)] \leq +\infty$, then [Gut, 2013]

$$\mathbb{P}(|X| > x) \leq \frac{\mathbb{E}[g(|X|)]}{g(x)}$$

Setting $g(x) = x^r$ for $r > 0$ we obtain *Markov's inequality*.

Chebyshev's inequality

Applying the above inequality for $X - \mathbb{E}[X]$ with $g(x) = x^2$ one obtains

$$\mathbb{P}(|X - \mathbb{E}[X]| > x) \leq \frac{\mathrm{Var(X)}}{x^2}$$

which is called *Chebyshev's inequality*.

Hoeffding's inequality

Hoeffding's inequality [Hoeffding, 1963] can be used to bound the deviation of the empirical mean of a bounded random variable from its expectation. If X is a random variable such that $|X| \leq R$, for some positive constant R, and $X_i,\ i = 1, 2, ..., n$ are independent copies of X, then

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}[X]\right| \geq t\right) \leq 2\exp\left\{-\frac{2nt^2}{R^2}\right\}$$

If X is symmetric we can bound either of the tails with half the probability

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}[X] \geq t\right) \leq \exp\left\{-\frac{2nt^2}{R^2}\right\}$$

Hoeffding's inequality in Banach spaces

For a Banach space valued random variable $Z \in \mathcal{B}$ there exists an analogous inequality to Hoeffding's inequality, with the absolute value replaced by the norm [Pinelis, 1994, Pinelis and Sakhanenko, 1986, Ledoux and Talagrand, 2013]. One of several versions of the inequality states that if $Z - \mathbb{E}[Z]$ is bounded by B, then with probability at least $1 - 2e^{-\delta}$ we have

$$\left\|\frac{1}{n}\sum_{i=1}^{n} Z_i - \mathbb{E}[Z]\right\|_{\mathcal{B}} \leq \frac{\sqrt{2\delta}B}{\sqrt{n}}$$

The inequality still holds if the iid assumption is relaxed and $Z_1, Z_2, ..., Z_n$ is instead a martingale.

Method of bounded differences

An inequality that has been profitably applied to measure the accuracy of kernel PCA is the *Method of bounded differences* or *McDiarmid's inequality* [Shawe-Taylor et al., 2005, McDiarmid, 1997]. Similar in spirit to Hoeffding's inequality

and with the same exponential decay of the tails, it states that if there are real-valued functions $f, f_1, f_2, ..., f_n$ such that for $i = 1, 2, ..., n$

$$\sup_{x_1,x_2,...,x_n} |f(x_1, x_2, ..., x_n) - f_i(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n)| \leq c_i$$

or

$$\sup_{x_1,x_2,...,x_n,\hat{x}_i} |f(x_1, x_2, ..., x_n) - f_i(x_1, x_2, ..., x_{i-1}, \hat{x}_i, x_{i+1}, ..., x_n)| \leq c_i$$

then

$$\mathbb{P}\left(f(X_1, X_2, ..., X_n) - \mathbb{E}[f(X_1, X_2, ..., X_n)] \geq t\right) \leq \exp\left\{-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right\}$$

Bernstein's inequality

Hoeffding's inequality can be derived from this inequality, which states that [Sen et al., 2010]

$$\mathbb{P}(X \geq x) \leq \inf_{\mathbb{E}[e^{tX}] \text{ exists}} \frac{\mathbb{E}[e^{tX}]}{e^{tx}}$$

Inequalities from Banach and Hilbert space theory

When X is an element in a Banach space with norm $\|X\|$, or a Hilbert space with inner product $\langle X, Y\rangle$, all results on Banach and Hilbert spaces carry over to random variables.

In particular, the space of all random variables for which $\mathbb{E}[|X|^p] < +\infty$ form a Banach space with the norm $\|X\|_p = \mathbb{E}[|X|^p]$, and all random variables with $\mathbb{E}[X^2] < +\infty$ constitute a Hilbert space with inner product $\langle X, Y\rangle = \mathbb{E}[XY]$.

2.2 THEORY OF COMPUTATION

The theory of computation lays the foundation for reasoning about algorithms devised to solve computational problems. It provides the framework for measuring the performance of computational algorithms, which becomes particularly important for kernel methods, since their high computational demands is often what limits their practical use. For example, we had to measure the computational requirements of our incremental algorithm for kernel PCA, to verify that it was more efficient than comparable existing algorithms and the corresponding batch algorithm. It was also important to be able to determine that our new method for kernel PCA with the Nyström method had the same computational complexity as the application of the Nyström method to other tasks.

A computational problem, or computation, is defined as the task of calculating a mathematical function $f(x)$ of some input data $x = x_1, x_2, ..., x_n$. The computational problem is solved by devising an *algorithm*, a set of logical and mathematical operations that describes how a specific function can be calculated for different input data [Sipser, 2013].

There are many different types of computational problems that can be studied. A class of problems of particular importance is that of *decision problems*, where the function to be calculated takes values in $\{0, 1\}$, often taken to mean TRUE or FALSE from Boolean logic [Chiswell and Hodges, 2007]. The input data may also be of different types. In its most general form, the input data examples $x_1, x_2, ..., x_n$ are strings of symbols from an alphabet and the output is another string from the same or another alphabet. In *digital computation*, the input and output strings are finite strings of bits, representing integer or floating-point numbers, mirroring the workings of a modern computer [Blondel and Tsitsiklis, 2000, Goldberg, 1991]. In *real* or *continuous* computation, the input can be arbitrary real numbers (see for example Blum et al. [1989]).

Important questions in the theory of computation are those regarding *decidability* and *solvability*. A problem is solvable if there is an algorithm that always stops and returns the right answer for any input data. A problem is decidable if the computation is only guaranteed to stop and return the right answer when a right answer exists.

In order to be able to reason about algorithms more formally one is in need of some model of computation. Given a model of computation and an algorithm, a *program* is the specific instructions used to implement that algorithm, chosen from the set of admissible instructions from the computational model. Ideally this model should mirror the functioning of the device one uses to solve the problem practically. Many such models of computation exist and are often some form of theoretical device. We will describe a few of them below.

The related theory of *complexity* studies the resources required by specific algorithms or programs, often as a function of the size of the input data, suitably defined. For example, the resources studied may be the time and memory requirements, which can be defined in different ways, often depending on the specific model of computation used.

2.2.1 Turing machine

The most important model of computation is the *Turing machine*, devised by Alan Turing in 1936. It was the first general-purpose theoretical machine devised, that could describe solutions to many common computational problems. Many other models of computation have been shown to be equivalent to the Turing machine, in that the Turing machine can simulate any algorithm implemented with such a model of computation with at most polynomial slowdown, in other words with at most an additional polynomial factor in the problem size added to the total running time.

The Turing machine consists of a number of components. It has an infinite tape consisting of a sequence of cells where symbols are written, one in each cell. It has a read-write head that reads symbols from and writes symbols to the tape and which can move in both directions along the tape. Each symbol belongs to a predefined *alphabet*. The machine has a number of internal states and a transition function that determines the next internal state from the current state and the symbol read from the input tape. At each transition, the head also writes a new symbol into the current cell and possibly moves left or right to an adjacent cell.

In the starting state of the machine there are a number of symbols written on the tape and the head is located at one of the cells. The machine proceeds to reading the symbol in the cell, changing its internal state, writing a symbol in the cell, and possibly moving left or right, all in accordance with the transition function. The machine continues like this until it reaches the *halting state*, at which point the output of the computation is written on the tape.

One Turing machine then represents a specific program. Each step of the machine is generally defined to have unit cost, and the total cost of the program is then the total number of steps until the halting state is reached.

Turing machines can be combined in sequence to produce a new Turing machine, representing the sequential execution of algorithms. They may also be run repeatedly, with the output of one run of the machine forming the input to a subsequent run, which mirrors an iterative algorithm.

The Turing machine can be considered as a (discrete) dynamical system if we maintain a cell on the tape where we write the internal state at each step [Moore, 1990]. The action of the Turing machine then becomes a map $\phi : S \to S$ [Barreira and Valls, 2012].

2.2.2 λ-calculus

The *lambda calculus* is a computational model which is equivalent to the Turing machine [Rosser, 1984]. Lambda calculus forms the basis for functional programming, whereas the Turing machine is more closely aligned to the imperative programming paradigm [Gyori et al., 2013]. In lambda calculus a program is built entirely from the application of a number of functions of a single argument. For details see Rosser [1984].

2.2.3 Random access machine

A model of computation that is more intuitive than the Turing machine, in its ressemblance to the workings of a modern computer, is the Random Access Machine (RAM) model [Blondel and Tsitsiklis, 2000]. It is equivalent to the Turing machine in the sense that that any algorithm implemented on a RAM has an equivalent implementation on a Turing machine, with the same set of decidable problems and belonging to the same complexity class. The RAM model consists of tapes for storing the input and output, an arbitrary number of registers for storing intermediate values, and a set of instructions that a program can use. The instructions include the usual arithmetic operations, instructions for reading and writing data in different locations on the tapes and registers, including memory addresses, and the halting instruction, which stops the program.

2.2.4 Running times

The running time of an algorithm for a specific set of input data is defined as the total time a program takes before it halts, given specified times taken for individual instructions. Often the time of each instruction is assumed to be the same, whether for example an addition or a multiplication, and regardless of the size of the ingoing numbers, but there are other models that can be more realistic.

For example, the *bit model* assumes that the time taken for arithmetic operation equals the number of bits in the input data.

For a given algorithm, the running time will be different for different instances of input data of the same size, depending on what the data looks like, where the size of input data is often measured as the number of data points or the number of bits. To achieve a measure of running time for an algorithm that is independent of the data, one must specify how to aggregate the running time across different input data. Most often, one considers the worst-case running time across all possible input data, but other options are available, such as the average running time.

If we let s denote the size of a problem and $T(s)$ the worst-case running time for that size, we say that an algorithm runs in polynomial-time if

$$T(s) = \mathcal{O}(s^k) \tag{2.1}$$

for a finite constant k, where $\mathcal{O}(\cdot)$ refers to Big-O notation [Cormen, 2009].

2.2.5 Complexity classes

The class or set of decision problems that are solved with algorithms taking polynomial-time is denoted P. The class NP consists of problems whose solution can be verified to be correct in polynomial time and is a superset of P. Problems in NP are often combinatorial in nature and involves checking a large number of solutions. The classes were explained by Donald Knuth as follows [Knuth, 2018]. Problems in P can be solved through a number of sequential steps that grows polynomially with the problem size. Problems in NP can be represented as a (rooted) tree, where each path from the root has the same length that grows with the problem size and each node has the same branching factor. Each path represents one attempt at a solution, exactly one attempt succeeds and each attempt takes polynomial time.

Equivalently, a problem in NP can be solved in polynomial-time by a non-deterministic Turing machine. A non-deterministic Turing machine works like the deterministic Turing machine described above, but a number of possible next states are specified in the transition function, instead of a single one.

A problem is called NP-complete if any problem in NP can be reduced to such a problem in polynomial time. These are in a sense the hardest problems in NP, since any problem in NP can be converted into an equivalent NP-complete problem, which can be solved instead. A problem is called NP-hard if it is at least as hard as some NP-complete problem. Whether in fact P = NP, i.e. whether a problem that can be verified in polynomial time necessarily also can be solved in polynomial time, is not known, despite progress towards a solution in recent years [Aaronson, 2016].

A problem belongs to exponential-time EXP if it can always be solved with a running time $\mathcal{O}(2^{s^k})$ for a problem size s. A problem belongs to PSPACE if it can always be solved by a Turing machine using a polynomial amount of memory in the problem size, i.e. a polynomial amount of cells used to store intermediate numbers.

2.3 SUBSPACE LEARNING

In subspace learning one attempts to create an often low-dimensional subspace to represent the data at hand, rather than relying on the n-dimensional Euclidean space enforced by the nature of the data one has collected. Various statistical techniques can then be applied in this subspace instead. Apart from as a tool for dimensionality reduction and feature extraction, subspace learning can for example also be used for classification, by attempting to create a subspace representative of each class, or anomaly detection, also known as one-class classification or novelty detection, by categorizing those points as anomalies that

are far from a fitted subspace. Examples of subspace learning methods include PCA and manifold learning, described below.

If the estimated subspace has lower dimensionality than the data, then subspace learning constitutes dimensionality reduction, a broad term for any method that in some way reduces the dimensionality of the data.

2.3.1 Principal component analysis

PCA finds the set of orthogonal linear combinations of variables that maximizes the variance of each linear combination in turn. This corresponds to calculating the eigendecomposition of the covariance matrix of observations. PCA can be applied for dimensionality reduction, by choosing the top principal components and discarding the rest. It can be applied in regression problems by performing a regression in the new coordinate system defined by the principal components, termed principal component regression (PCR). It can also be used in classification problems and clustering, and to detect outliers and for novelty detection. For a definitive treatment see Jolliffe [2002].

The principal components are equal to the eigenvectors of the sample covariance matrix of the data, $C = X^T X$ for a data matrix of (centred) observations X, where each observation occupies a row, although sometimes these quantities are referred to as the *principal axes* instead. We have the decomposition $C = V \Lambda V^T$ where the columns of V are the eigenvectors and the directions of maximum variance, and the diagonal matrix Λ contains the eigenvalues in descending order along its diagonal, which are termed the *principal values* and equal the variances captured by the principal components. The data points in terms of the principal components, known as the scores, are given by XV. The scores are often denoted by T, in matrix form, or $\mathbf{t}_i$ for the weightings of the different principal components when reconstructing data point x_i.

The principal components can also be obtained through the singular value decomposition (SVD), given by $X = U\Sigma V^T$, where the columns of U are the eigenvectors of $X^T X$, i.e. the principal components, and the columns of V the eigenvectors of XX^T.

The eigenvectors U_d corresponding to the d largest eigenvalues minimize the *reconstruction error* [Li, 2004]

$$\sum_{i=1}^{n} \|U_d U_d^T x_i - x_i\|_2^2 \tag{2.2}$$

over all $n \times d$ matrices. $U_d U_d^T$ is the projection matrix onto the linear span of the columns of U_d, i.e. the top d eigenvectors. If $d = n$ this quantity will be zero for all orthogonal matrices U_d.

PCA is known to be sensitive to outliers, which may complicate practical application. This has motivated various robust PCA approaches, for example using influence functions [Huber, 1996]. Also see De La Torre and Black [2003], Candès et al. [2011], Jolliffe [2002] for further details.

Probabilistic PCA

Principal component analysis is not based on a probabilistic model for the data, but it can be related to factor analysis and latent variable models, which rely on the specification of an explicit model where the quantities of interest are assumed to follow specific distributions [Tipping and Bishop, 1999]. First, we have that the reconstruction of a data point x_i in terms of the projections onto the principal components is given by

$$\hat{x}_i = U_d U_d^T x_i = U_d z_i$$

where we have denoted $z_i = U_d x_i$ the projection coefficients in the d-dimensional eigenspace. Adding noise to account for the difference between the true data and

the reconstruction, one obtains the model

$$x_i = U_d z_i + \varepsilon_i$$

which is termed a factor analysis or latent variable model. If the projection coefficients z_i and the errors ε_i are iid and Gaussian and assumed to be known, then the maximum likelihood estimates of U_d correspond to the principal components. Various extensions also exist with less restrictive assumptions, such as not assuming the error variances to be equal and estimating them as part of the maximum likelihood procedure.

Sparse PCA

In the high-dimensional setting when the number of variables increases at the same rate as the number of data points, $p/n \to (0, 1)$, the standard estimate of the first principal component as the leading eigenvector of the empirical covariance matrix is inconsistent [Wang et al., 2016]. Sparse PCA, which constrains the principal components to be sparse, remedies this issue, as well as improving interpretability of the principal components, since each principal component is forced to be a linear combination of only a subset of variables.

A natural sparse estimate of the leading eigenvector is

$$\hat{u}_1 = \arg \max_{u \in B_0(k)} u^T \hat{\Sigma} u$$

where $\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$ is the empirical covariance matrix and

$$B_0(k) = \left\{ u \in \mathbb{R}^p \;\middle|\; \sum_{i=1}^{p} \mathbb{1}_{\{u_i \neq 0\}} \leq k, \|u\|_2 = 1 \right\}$$

This estimate achieves the minimax optimal rate over sub-Gaussian distributions, but calculating it is NP-hard. Wang et al. [2016] showed that there is no

(randomized) polynomial time algorithm that achieves the optimal minimax rate.

Functional PCA

Functional PCA is PCA not applied to data as vectors in $\mathbb{R}^p$, but to data as functions in some Hilbert space $\mathcal{H}$, sampled at some arbitrary or evenly spaced intervals [Besse and Ramsay, 1986, Hall et al., 2006]. Therefore, functional PCA corresponds to kernel PCA when this Hilbert space is also a reproducing kernel Hilbert space, as is often the case, but the functions are accessed directly and not through the kernel function.

2.3.2 Multi-dimensional scaling

Multi-dimensional scaling (MDS) finds a lower-dimensional representation of data from a matrix of distances between data points [Hout et al., 2013]. MDS is equivalent to kernel PCA when the kernel is *isotropic*, i.e. on the form $f(\|x-y\|)$ for some function f [Williams, 2002]. Many of the methods presented in this thesis can therefore also be applied to MDS.

2.3.3 Canonical correlation analysis

Canonical correlation analysis (CCA) is related to principal component analysis, but instead of looking for a linear combination of a random variable X that has maximum variance, it looks for two linear combinations of variables X, Y such that their correlation is maximized [Hardoon et al., 2004]. If X, Y have dimensions d_1, d_2 respectively, one wishes to solve

$$a, b = \arg\max_{\mathbb{R}^{d_1}, \mathbb{R}^{d_2}} \operatorname{Corr}(a^T X, b^T Y) \tag{2.3}$$

In other words, one finds two subspaces $a^T X$ and $b^T Y$ in $L^2(\mathbb{P})$ that are as parallel as possible, where $\mathbb{P}$ is the joint law of X and Y.

Replacing the expectations by their empirical estimates and solving the optimization problem (2.3) one can obtain the vector a by solving the eigenproblem

$$\Sigma_{xx}^{-1}\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}a = \lambda a \tag{2.4}$$

for the top eigenvector, and the vector b through

$$b = \frac{1}{\lambda}\Sigma_{yy}^{-1}\Sigma_{yx}a \tag{2.5}$$

where

$$\begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yy} & \Sigma_{yx} \end{bmatrix} = \hat{\mathbb{E}}[[X \; ; \;\; Y][X^T \quad Y^T]]$$

with $\hat{\mathbb{E}}$ denoting the empirical expectation.

2.3.4 Manifold learning

In manifold learning the data examples are assumed to lie on a lower-dimensional manifold, i.e. a space with local Euclidean structure, and the learning task is to find this manifold. Prominent examples are the isomap algorithm, which is based on multi-dimensional scaling (MDS) [Tenenbaum et al., 2000, Silva and Tenenbaum, 2002], and Locally-linear embedding (LLE) [Roweis and Saul, 2000]. In the experiments in Chapter 6 we compare our proposed method to a number of other unsupervised learning methods, including LLE.

2.4 KERNEL METHODS

Let X be a vector of square-integrable random variables $X_1, X_2, ..., X_q$ on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ to a space $\mathcal{X}$, i.e. $X_i \in L^2(\Omega, \mathcal{A}, \mathbb{P}; \mathcal{X})$. If X_i is real-valued the inner product becomes $\langle X_i, X_j \rangle = \mathbb{E}[X_i X_j] = \int_\Omega X_i X_j d\mathbb{P}$. Note that any set can be supplied with a σ-algebra hence be made measurable, since

$\{\emptyset, \mathcal{X}\}$ is a σ-algebra. Linear methods consider elements in the linear subspace given by the span of X, i.e. linear combinations of the form $c^T X$ where $c \in \mathbb{R}^q$. We let $\{x_i\}_{i=1}^n$ denote iid realizations from X and elements in $\mathcal{X}$.

Kernel methods allow for the application of linear methods to discover non-linear patterns between variables, through a non-linear transformation of data points $\phi(x)$ on which linear algorithms can be applied [Hofmann et al., 2008]. They rely on two things. First, the calculation of inner products between transformed data examples through a symmetric positive definite *kernel* $k(x, y)$; second, the expression of a solution linearly in the space of transformed data examples, rather than in the space of transformed variables. We place no further restrictions on the set $\mathcal{X}$. The possibility of letting $\mathcal{X}$ be any set is one of the great benefits of kernel methods. $\mathcal{X}$ can for example be a collection of text strings or graphs [Lodhi et al., 2002, Vishwanathan et al., 2010].

For ease of mathematical exposition we let $\mathcal{X}$ be a vector space from now on, i.e. closed under vector addition and multiplication by a scalar. We will take the scalar field of the vector space to be the real numbers $\mathbb{R}$. Let $\mathcal{H}$ be a Hilbert space of functions on $\mathcal{X}$. We will denote the inner product in this space by $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Consider $\mathcal{H}'$, the dual of $\mathcal{H}$, i.e. the space of linear functionals on $\mathcal{H}$. Then for every $x \in \mathcal{X}$ there is a unique $\delta_x \in \mathcal{H}'$ such that $\delta_x(f) = f(x)$, termed the evaluation functional. If δ_x is bounded (i.e. continuous), then by the Riesz representation theorem there is a unique element $g_x \in \mathcal{H}$ such that $\delta_x(f) = \langle g_x, f \rangle_{\mathcal{H}}$ [Bollobás, 1999], and hence $\langle g_x, f \rangle_{\mathcal{H}} = f(x)$. Any function in $\mathcal{H}$ can therefore be evaluated through the inner product with some other function $g_x \in \mathcal{H}$. If we consider g_x as a function of x, say $k(x, \cdot)$, then this function has the so called reproducing property, i.e. $\langle k(x, \cdot), f(\cdot) \rangle_{\mathcal{H}} = f(x)$. Furthermore, by the reproducing property, we have $\langle k(x, \cdot), k(y, \cdot) \rangle_{\mathcal{H}} = k(x, y)$. The function $k(x, y)$ is then a symmetric positive definite function, by the symmetric positive definiteness of the inner product. The space $\mathcal{H}$ is known as a *reproducing kernel*

Hilbert space (RKHS) [Berg et al., 1984].

The function $k(x, \cdot)$ is often denoted by $\phi(x)$, termed a *feature map*, and can be seen as a transformation of a point x into the space of functions $\mathcal{H}$. The feature map $\phi(x)$ then maps each element $x \in \mathcal{X}$ to a function in $\mathcal{H}$, which can be evaluated through the inner product with some other element in $\mathcal{H}$, which in turn can be calculated through the kernel function $k(x, y)$.

The space $\mathcal{H}$ may have uncountable dimension, but it is often assumed to be separable, and is then isometrically isomorphic to ℓ^2, the space of square-summable sequences [Bollobás, 1999]. Each element $\phi(x_i)$ then has a representation as a real countably infinite sequence $\phi(x_i) = (\phi_1(x_i), \phi_2(x_i), ...)$, with $\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}} = \sum_{k=1}^{\infty} \phi_k(x_i)\phi_k(x_j)$. We call these *feature vectors*. However, this representation is often not known, or the dimension is very large, so it might not be possible to apply a linear method directly on the variables $\phi_1(x), \phi_2(x), ...$

Alternatively, let $k(x, y)$ be a symmetric positive definite kernel on $\mathcal{X} \times \mathcal{X}$. Then by the Moore-Aronszajn theorem [Aronszajn, 1950] there is a unique Hilbert space $\mathcal{H}_k$ of functions on $\mathcal{X}$, for which k has the reproducing property, i.e. $\forall f \in \mathcal{H}_k, \quad \langle f(\cdot), k(x, \cdot) \rangle_{\mathcal{H}_k} = f(x)$. This space is known as the reproducing kernel Hilbert space (RKHS) and $\langle f(\cdot), k(x, \cdot) \rangle_{\mathcal{H}_k}$ is called the evaluation functional. If we let a point in $\mathcal{H}_k$ be denoted $\phi(x) = k(x, \cdot)$, then $\langle \phi(x), \phi(y) \rangle_{\mathcal{H}_k} = \langle k(x, \cdot), k(y, \cdot) \rangle_{\mathcal{H}_k} = k(x, y)$ by the reproducing property.

The key to the application of kernel methods in statistics is that any function $f \in \mathcal{H}$ can be evaluated through the inner product, hence through the kernel function. Since any f is a function from $\mathcal{X}$ to $\mathbb{R}$, it is often natural to look in the space $\mathcal{H}$ for a function of the data with good inferential properties.

We thus want to perform statistical inference to learn one or more functions

$f \in \mathcal{H}$ for some specific purpose. This can be accomplished in the setting of empirical risk minimzation

$$f^* = \arg\min_{f \in \mathcal{H}_k} \left\{ \sum_{i=1}^{n} L(y_i, f(x_i)) + \Omega\left(\|f\|_{\mathcal{H}_k}\right) \right\} \tag{2.6}$$

where L is a loss function (not necessarily convex in the second argument) and Ω is a strictly increasing function. The representer theorem [Schölkopf et al., 2001] states that the solution to this problem can be written as $f^* = \sum_i \alpha_i k(x_i, \cdot)$ for coefficients α, i.e. the minimizer is in the span of the functions induced by the data points.

We also note that each symmetric positive definite kernel k defines a linear operator $T_k : L^2 \to \mathcal{H}$ through

$$T_k f = \int_{\mathcal{X}} k(x, y) f(y) dy$$

where $\mathcal{H}$ is again the space of functions on $\mathcal{X}$ to $\mathbb{R}$. Its spectral decomposition is given by $T_k \phi = \lambda \phi \;\Leftrightarrow\; \int_{\mathcal{X}} k(x, y)\phi(y)dy = \lambda\phi(y)$ giving eigenfunctions $\phi_1, \phi_2, ..., \phi_d$ where $\langle \phi_i, \phi_j \rangle = \delta_{ij}$, with δ_{ij} the kronecker delta, and eigenvalues $\lambda_1, \lambda_2, ..., \lambda_d$. The kernel function then has the representation $k(x, y) = \sum_{i=1}^{d} \lambda_i \phi_i(x)\phi_i(y)$.

A kernel function can also be characterized by Mercer's theorem, provided $\mathcal{X}$ is compact – each continuous symmetric positive semi-definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ has a representation $k(x, y) = \sum_{i=1}^{d} \lambda_i \phi_i(x)\phi_i(y)$, where $\lambda_i > 0$ and $\{\phi_i\}_{i=1}^{d}$ are continuous and form an orthonormal basis in $L^2(\mathcal{X})$. λ_i, ϕ_i are the eigenvalues and eigenfunctions of the linear operator $Tf = \int_{\mathcal{X}} k(x, y) f(y) dy$, $T : L^2(\mathcal{X}) \to L^2(\mathcal{X})$.

We may arrange the data vectors $\phi(x_1),\ \phi(x_2),\ ...,\ \phi(x_n)$ in $\mathcal{H}$, taken to be

column vectors, along the rows of a data matrix Φ_n. For notational convenience we may write $\phi(x_i) := \phi_i$ and we occasionally denote the data vectors by $\Phi(x_i)$. The matrix $(k(x_i, x_j)) \in \mathbb{R}^{n \times n} = \Phi\Phi^T := K$ is called the kernel matrix. The kernel matrix has dimension $n \times n$ and many algorithms that involve it, such as solving a linear system, finding the eigendecomposition, or quadratic programming, scale as $\mathcal{O}(n^3)$.

The flexibility in the choice and construction of kernel functions contributes to the appeal of kernel methods. Sums and products of kernel functions are also kernels, giving a flexible framework for construction of kernel functions to suit a particular task.

Many kernel functions have the property that $k(x, x) = C, \quad \forall x$ and for some $C \in \mathbb{R}$. One important class of kernel functions for which this holds are the translation-invariant (or shift-invariant) ones, for which $k(x - c, y - c) = k(x, y) \;\; \forall x, y, \; \forall c \in \mathcal{X}$.

2.4.1 Gaussian processes

Essentially the same results can be obtained through a Bayesian formalism by specifying a Gaussian process prior on the hypothesis space of functions [Rasmussen and Williams, 2006]. Whilst the RKHS derivation relies on e.g. the Moore-Aronszajn theorem to choose a positive definite kernel function that corresponds to some transformation of the data points, for Gaussian processes one specifies a covariance function between two points. This essentially leads to the same set of kernel functions, since the covariance function needs to be positive definite to describe a valid covariance matrix. Often stationarity of the covariance is assumed, which corresponds to using a translation-invariant kernel.

2.4.2 Examples of kernel methods

Many linear methods from statistics and machine learning have been adapted to be applied in the reproducing kernel Hilbert space. In this section we provide some examples of common kernel methods, to illustrate the concepts introduced above.

Kernel SVM

The maximum-margin classifiers were one of the first methods to be adapted to using kernels [Bishop, 2006, Vapnik, 1998]. The kernel support vectors machine (SVM) finds the two parallel hyperplanes that maximally separates data examples from different classes in feature space, i.e. two linear subspaces of the form

$$\{x \mid f(x) = \alpha\}$$

where f is a linear functional and α a constant. Similarly, a hyperplane in feature space is given by $\{x \mid \sum \alpha_i k(x_i, x) = \alpha\}$. The dual of the optimization problem for finding the maximum margin has the following form

$$\min_{\alpha_i} \left\{ \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right\}$$

which is expressed entirely in terms of the kernel function and other known quantities, where the indices run over all n data examples, subject to the constraints $\alpha_i \geq 0$ and $\sum \alpha_i y_i = 0$.

Kernel SVR

Kernel support vector regression (SVR) uses a hyperplane for function estimation in regression tasks [Smola and Schölkopf, 2004]. It attempts to find a hyperplane $\langle w, x \rangle + b = 0$ with margins ε on either side such that all training points are within the margins, under the constraint that the hyperplane be as flat as possible.

As such it solves the following optimization problem

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \|w\| \\ \text{subject to} \quad & |y_i - w^T x_i - b| \leq \varepsilon \end{aligned}$$

To deal with datasets that don't fit with a margin ε one may intrpoduce slack variables ξ_i and solve the alternative optimization problem

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \|w\| + \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & |y_i - w^T x_i - b| \leq \varepsilon + \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

Kernel SVR with zero margin is equivalent to kernel ridge regression, described below, but with a mean absolute deviation loss function. We have to rewrite the above optimization problem in its dual form, which can be expressed entirely in terms of inner products of the data.

Kernel ridge regression

Kernel ridge regression is the application of linear regression in feature space, including a ridge penalty [Saunders et al., 1998, Campbell, 2002]. Consider the data matrix of transformed data examples Φ, where each data example occupies a row in the matrix. The regression coefficients in feature space are given by the well-known formula

$$\beta = (\gamma I + \Phi^T \Phi)^{-1} \Phi^T y$$

where γ is the regularization parameter. Applying a well-known matrix identity, we can rewrite the above expression as

$$\beta = \Phi^T (\gamma I + \Phi \Phi^T)^{-1} y$$

The predicted value for a new data example x_* is then given by

$$\hat{y} = \beta^T \Phi(x^*) = y^T(\gamma I + \Phi\Phi^T)^{-1}\Phi^T\Phi(x_*)$$
$$:= \beta^T \Phi(x^*) = y^T(\gamma I + \Phi\Phi^T)^{-1}\kappa(x^*) := \sum \alpha_i k(x_i, x_*)$$

where $\Phi(x_*)$ is the data example transformed into feature space as a row vector, and

$$\alpha := y^T(\gamma I + \Phi\Phi^T)^{-1} = y^T(\gamma I + K)^{-1}$$

Note that we rewrote the required prediction in terms of the data examples without reference to the representer theorem. The result can also be derived as an optimization problem using the dual formulation.

Kernel PCA

Kernel PCA is obtained through the application of linear PCA in the reproducing kernel Hilbert space. It has been demonstrated to outperform linear PCA in a number of applications [Chin and Suter, 2007]. Assuming centered data, kernel PCA performs the eigendecomposition of the covariance matrix in feature space through [Schölkopf et al., 1998]

$$\frac{1}{n}\Phi^T\Phi\mathbf{v} = \lambda^*\mathbf{v} \tag{2.7}$$

resulting in the decomposition $\frac{1}{n}\Phi^T\Phi = V\Sigma V^T$. Henceforth we will set $\lambda := n\lambda^*$ and only be concerned with the eigendecomposition of $\Phi^T\Phi$. Noting that $\text{span}\{\Phi^T\} = \text{span}\{V\}$, we can write $\mathbf{v}$ in terms of an n-dimensinal vector $\mathbf{u}$ as $\mathbf{v} = \Phi^T\mathbf{u}$. Left-multiplying the eigenvalue equation by Φ we obtain $K\mathbf{u} = \lambda\mathbf{u}$ and the decomposition $K = U\Lambda U^T$.

The eigenvalue equation $Ax = \lambda x$ has potentially an infinite number of solutions for x, if the additional constraint of x having unit norm is not imposed. The relationship $\mathbf{v} = \Phi^T\mathbf{u}$ does not impose unit norm, so $\mathbf{u}$ and $\mathbf{v}$ could be any

solutions of $\Phi\Phi^T\mathbf{u} = \lambda\mathbf{u}$ and $\Phi^T\Phi\mathbf{v} = \lambda\mathbf{v}$, respectively.

If only a subset of k principal components are to be retained, we truncate the eigenvector and eigenvalue matrices such that

$$K \approx U_k \Lambda_k U_k^T$$

where U_k contains the first k columns of U, and Λ_k contains the first k rows and columns of Λ. Since the principal components $\mathbf{v}$ are related to the eigenvectors of K through $\mathbf{v} = \Phi^T\mathbf{u}$, truncating the eigendecomposition of K also corresponds to truncating the eigendecomposition in the feature space.

If the data vectors in feature space are not assumed to be centred, we need to subtract the mean of each variable from Φ and instead calculate the eigendecomposition of

$$K' = (\Phi - \mathbb{1}_n\Phi)(\Phi - \mathbb{1}_n\Phi)^T = K - \mathbb{1}_n K - K\mathbb{1}_n + \mathbb{1}_n K \mathbb{1}_n \tag{2.8}$$

where $\mathbb{1}_n$ is a matrix for which $(\mathbb{1}_n)_{i,j} = \frac{1}{n}$.

Kernel PCR

Kernel principal components regression (PCR) is the kernel equivalent of principal components regression, in other words principal components regression carried out in the reproducing kernel Hilbert space [Rosipal et al., 2001]. Principal components regression replaces the individual regressors in a linear regression by a subset of their principal components [Jolliffe, 1982]. In this way, a bias in the estimates for the response variable is introduced, but regularization may be achieved and the variance of the estimates may be reduced. For example, principal components regression is known to ameliorate collinearity of the regressors, which leads to high variance of the coefficient estimates. Principal components regression is equivalent to the *errors-in-variables* model, where the independent

and dependent variables are assumed to contain measurement noise, under some assumptions on the covariance of the measurement noise [Fuller, 1980].

Kernel PCR adapts principal compentents regression to be carried out in the high-dimensional RKHS, by projecting the data examples Φ onto the top k principal components V_k, i.e. the top eigenvectors of $\Phi^T\Phi$. The projection is given by ΦV_k.

The estimated PCR model is given by [Rosipal et al., 2001]

$$f(x^*) = w^T U_k^T \kappa(x^*)$$

where U_k consists of the first k columns of the eigenvector matrix U, and where

$$w = \Lambda_k^{-1} V_k^T \Phi^T y$$

The projection ΦV_k is not directly known, since we do not have access to the explicit feature maps. Instead Rosipal et al. [2001] suggest to estimate the weights w through the expectation-maximization (EM) algorithm [Moon, 1996]. A closed form expression for kernel PCR can also be calculated, but existing approaches do not account for non-zero mean in feature space [Rosipal et al., 2000, 2001, Wibowo and Yamamoto, 2012]. In Chapter 6 we present a novel specification for kernel PCR that removes the need to assume that data in feature space has zero mean.

Kernel CCA

Instead of performing canonical correlation analysis in Euclidean space, one can attempt to find vectors f, g in a reproducing kernel Hilbert space such the correlation between $f(X)$ and $g(Y)$ is maximized, for random variables X, Y [Fukumizu et al., 2007].

Kernel FDA

Linear discriminant analysis, a generalization of Fisher discriminant analysis, is a popular method for classification. Fisher discriminant analysis was adapted to the reproducing kernel Hilbert space in Mika et al. [1999].

Others

A host of other methods have been adapted to be used with kernels, including independent component analysis (ICA) [Bach and Jordan, 2002], instrumental variable (IV) regression [Singh et al., 2019], partial least squares regression [Rosipal and Trejo, 2001], clustering [Filippone et al., 2008] and blind source separation [Harmeling et al., 2003].

2.4.3 Scalable kernel methods

Due to the large computational complexity of kernel methods, they are often not practically usable in their original form, or competitive with other methods, such as neural networks. Many techniques have been proposed to make kernel methods more scalable, many of which are based on the introduction of approximations to the exact solution. The introduction of an approximation can also lead to statistical regularization, as elaborated on below.

We point out that since we are working with data that is generated from an unknown probability distribution, any quantity calculated from the data is an approximation to the true value of the quantity as determined by its underlying probability distribution. An approximation to a standard estimation method can also be seen as just another estimate of the true value.

Low-rank matrix approximation

Approximations are often based on the introduction of a low-rank approximation to the kernel matrix K through $K \approx AB^T$, where A and B are thin matrices,

i.e. if K is $n \times n$, then A, B are $n \times r$ with $r < n$. While no longer exact, this leads to more efficient computation, since each column of K is approximated by a linear combination of a smaller number r of columns. This is the case for the Nyström method and the incomplete Cholesky factorization, which has been applied to kernel methods [Fine and Scheinberg, 2001]. The latter method works by setting certain columns of the Cholesky factor L to zero, where $A = LL^T$, thus resulting in a low-rank approximation. Low-rank matrix approximation is also known as matrix sketching [Liberty, 2013].

Early stopping

Since an iterative algorithm produces a sequence of solutions that converges towards the exact solution, an approximate algorithm can be obtained by limiting the number of iterations prematurely. Prematurely interrupting iterative algorithms is a way to achieve statistical regularization, termed *early stopping*, which is often applied in the training of neural networks.

Optimization-based methods

The success of neural networks on massive datasets has been founded partly on stochastic gradient descent (SGD) [Bottou, 2010]. SGD has also been applied to kernel methods, based on reformulations of the particular kernel method as an optimization problem [Dai et al., 2014].

Parallel computation

Another way to improve the computational performance of kernel methods is to employ parallel computation, which often results in an approximate solution. This is the case for the procedure in Zhang et al. [2013], where the dataset is divided into a number of subsets and a kernel ridge regression is calculated for each dataset in parallel, before the solutions are averaged.

Gaussian processes

In the Gaussian process literature similar approximate techniques are known as inducing points methods or sparse Gaussian processes. See for example Quiñonero-Candela and Rasmussen [2005], Snelson and Ghahramani [2007].

Numerical approximations

One method that has been used to speed up numerical algorithms is to round the decimal representation or otherwise employ lower-memory representations of floating-point numbers [Achlioptas and McSherry, 2007, Dettmers, 2015].

Random Fourier features

Random Fourier features [Rahimi and Recht, 2007] relies on Bochner's theorem, which states that any normalized shift-invariant kernel is the Fourier transform of a probability measure, giving

$$k(x-y) = \int_{[0,2\pi]^d} e^{i\omega^T(x-y)} d\mathbb{P}(\omega) = \int_{[0,2\pi]^d} p(\omega) e^{i\omega^T(x-y)} d\omega$$

$$= \mathbb{E}^{\mathbb{P}}\left[e^{i\omega^T x} e^{-i\omega^T y}\right] := \mathbb{E}^{\mathbb{P}}\left[\zeta_\omega(x)\zeta_\omega(y)^*\right]$$

If ω is sampled from p, then $\zeta_\omega(x)\zeta_\omega(y)^*$ is an unbiased estimate of $k(x-y)$. One can replace $e^{i\omega^T(x-y)}$ by $\cos(\omega^T(x-y))$ if the kernel is real-valued. To reduce variance one can repeat the sampling to create a vector $z(x)$ of sampled cosines.

2.4.4 The Nyström method

The Nyström method initially appeared in the 1930s in the discretization of integral equations [Nyström, 1930] and was extended to kernel methods in Williams and Seeger [2001]. In the latter setting, it randomly samples m data examples from the full dataset to produce a low-rank approximation $\tilde{K}$ to the

full kernel matrix K.

$$\tilde{K} = K_{n,m} K_{m,m}^{-1} K_{m,n}$$

where $K_{n,m}$ is an $n \times m$ matrix obtained by choosing m columns from the original matrix K, $K_{m,n}$ is its transpose and $K_{m,m}$ contains the intersection of the same m columns and rows. The approximation can be derived as follows. Each eigenfunction of $T_k f = \int_{\mathcal{X}} k(x,y) f(y) dy$ solves the spectral equation

$$\int_{\mathcal{X}} k(y,x)\phi_i(x)dx = \lambda_i \phi_i(y) \qquad i = 1, 2, ..., d$$

where d is the number of eigenfunctions. If we approximate this integral by a discretization at $m < n$ points $x_{\gamma_1}, x_{\gamma_2}, ..., x_{\gamma_m}$, where $\gamma_k \in \Gamma$ for an index set $\Gamma \subset \{1, 2, ..., n\}$ we obtain

$$\frac{1}{m} \sum_{j=1}^{m} k(y, x_{\gamma_j}) \phi_i(x_{\gamma_j}) \approx \lambda_i \phi_i(y) \qquad i = 1, 2, ..., d \tag{2.9}$$

This should hold approximately for each value of $y \in \{x_1, x_2, ..., x_n\}$. First let $y = x_{\gamma_1}, x_{\gamma_2}, ..., x_{\gamma_m}$, giving m equations for each $i = 1, 2, ..., d$

$$\frac{1}{m} \sum_{j=1}^{m} k(x_{\gamma_k}, x_{\gamma_j}) \phi_i(x_{\gamma_j}) \approx \lambda_i \phi_i(x_{\gamma_k}) \qquad k = 1, 2, ..., m \tag{2.10}$$

If we set $u_i := [\ \phi_i(x_{\gamma_1}) \quad \phi_i(x_{\gamma_2}) \quad ... \quad \phi_i(x_{\gamma_m})\ ]^T$, we can write the above equations in matrix form

$$\frac{1}{m} K_{m,m} u_i \approx \lambda_i u_i \qquad i = 1, 2, ..., d \tag{2.11}$$

Now if we solve the eigenvalue equation $\frac{1}{m} K_{m,m} v = \sigma v_i$ we will obtain m eigenpairs (σ_i, v_i) for which have $\sigma_i \approx \lambda_j, v_i \approx u_j$ for some (λ_j, u_j), i.e that approximate the eigenvalues of T_k and its eigenfunctions at points $x_{\gamma_1}, x_{\gamma_2}, ..., x_{\gamma_m}$. If we instead solve the eigenvalue equation equation $K_{m,m} u^{(m)} = \lambda^{(m)} u^{(m)}$,

and match up the indices, we obtain the approximations $\lambda_i \approx \lambda_i^{(m)}/m$ and $u_i \approx \sqrt{m}\, u_i^{(m)}$. Inserting these approximations into the discretization 2.10 we obtain

$$\phi_i(y) \approx \frac{\sqrt{m}}{\lambda_i^{(m)}} \mathbf{k}_y^T u_i^{(m)} \qquad i = 1, 2, ..., d$$

where $\mathbf{k}_y = [\ k(y, x_1) \quad k(y, x_2) \quad ... \quad k(y, x_m)\ ]^T$. If we instead discretize the integral using all n points, and write n equations for $y = x_1, x_2, ..., x_n$, we similarly arrive at approximations $\lambda_i \approx \lambda_i^{(n)}/n$ and $\phi_i(x_j) \approx \sqrt{n}(u_i^{(n)})_j$, where $(\lambda_i^{(n)}, u_i^{(n)})$ are the eigenpairs of K.

Now since both $\lambda_i \approx \lambda_i^{(n)}/n$ and $\lambda_i \approx \lambda_i^{(m)}/m$, and for each $j = 1, 2, ..., n$, both $\phi_i(x_j) \approx \sqrt{n}(u_i^{(n)})_j$ and $\phi_i(x_j) \approx \frac{\sqrt{m}}{\lambda_i^{(m)}} \mathbf{k}_{x_j}^T u_i^{(m)}$, we can approximate the eigenpairs of K by those of $K_{m,m}$ by equating the approximations at all points $j = 1, 2, ..., n$. We obtain for each $i = 1, 2, ..., n$

$$\lambda_i^{(n)} \approx \frac{n}{m} \lambda_i^{(m)}$$

$$(u_i^{(n)})_j \approx \sqrt{\frac{m}{n}} \frac{1}{\lambda_i^{(m)}} \mathbf{k}_{x_j}^T u_i^{(m)} \qquad j = 1, 2, ..., n$$

or in matrix form

$$\lambda_i^{(n)} \approx \frac{n}{m} \lambda_i^{(m)}$$

$$u_i^{(n)} \approx \sqrt{\frac{m}{n}} \frac{1}{\lambda_i^{(m)}} K_{n,m} u_i^{(m)}$$

Multiplying together the approximate eigendecomposition results in the approximation $\tilde{K} = K_{n,m} K_{m,m}^{-1} K_{m,n}$.

Often the m columns are sampled uniformly, but many other sampling methods have been proposed [Kumar et al., 2009], such as using a sampling distribution approximately proportional to the statistical leverage scores of the data examples

[Drineas et al., 2012].

Drineas and Mahoney [2005] suggest a variation of the above approximation obtained by replacing $K_{m,m}$ by an approximation of rank $r \leq m$ obtained by means of the truncated singular value decomposition (or eigendecomposition), thus introducing regularization. It is well known that the best rank r approximation in Frobenius and spectral norm to a matrix is given by the rank r truncated singular value decomposition. The expression for the approximate kernel matrix is given by

$$\tilde{K}_r = K_{n,m} W_r^{-1} K_{m,n}$$

where W_r is the rank r truncated singular value decomposition of $K_{m,m}$. They also derive a bound on the accuracy of the approximation when sampling columns proportionally to the diagonal elements of K.

Kernel ridge regression

To apply the Nyström method in kernel ridge regression one replaces the full kernel matrix K by its approximation $\tilde{K}$ in the regression function

$$\hat{f}(x) = y^T(\gamma I + \tilde{K})^{-1}\kappa(x) = y^T(\gamma I + K_{n,m}K_{m,m}^{-1}K_{m,n})^{-1}\kappa(x)$$

To be able to calculate the inverse of an $m \times m$ matrix we can apply the Woodbury matrix identity [Williams and Seeger, 2001], to obtain

$$(\gamma I + \tilde{K})^{-1} = \frac{1}{\gamma} I - \left(\frac{1}{\gamma}\right)^2 K_{n,m} \left(K_{m,m} + \frac{1}{\gamma} K_{m,n} K_{n,m}\right)^{-1} K_{m,n}$$

2.4.5 Multiple kernel learning

Multiple kernel learning (MKL) is the principled combination of individual kernel functions into more expressive kernel functions, for improved inference

properties. It relies on the fact that sums and products of kernel functions are still kernel functions, i.e. they are symmetric and positive definite.

One way to obtain a new kernel is as a sum of a number of basis kernels $k(x, y) = \sum_i k_i(x, y)$, which corresponds to the product of the individual feature spaces $\prod_i \mathcal{H}_i$, i.e. the feature space is a product space of smaller feature spaces [Bach, 2009]. The learning problem in this setting is to select a number of kernels $k_i(x, y)$ among a larger set of possible kernels.

Another way to combine different kernel functions is to create a linear combination of a fixed set of kernels [Gönen and Alpaydın, 2011]

$$k(x, y) = \sum_i \eta_i k_i(x, y)$$

The learning problem in this instance consists of estimating the weight parameters $\{\eta_i\}$. Other functions of a fixed set of basis kernels kan also be used to create a new kernel, including multiplications and exponentiations.

2.4.6 Invariance

An appealing property of kernel methods that contributes to their popularity is that invariance of the result to transformations of the data can be achieved by an appropriate choice of kernel function, regardless of which specific kernel method is used. Kernel methods could also therefore be particularly suited to estimation through the invariance principle, which we outlined in Section 2.1.2. Shift-invariant kernels, which are on the form $k(x, y) = \varphi(x - y)$ are invariant to any translation T_a of the input data, where

$$T_a x = x + a$$

since $\varphi(T_a x - T_a y) = \varphi(x - y)$. So called isotropic kernels, which are on the form $k(x, y) = \varphi(\|x - y\|)$, are invariant to both translations, rotations and reflections of the input data. This holds for example for the radial basis functions kernel.

A new kernel can be obtained from an existing one through the transformation

$$\tilde{k}(x, y) = \frac{k(x, y)}{(k(x, x)k(y, y))^{1/2}}$$

This kernel is scale invariant since the kernel function is bilinear. It also has $\sup_x \tilde{k}(x, x) = 1$ for any kernel k, which is taken advantage of in Chapter 6. Note however that the resulting transformed kernel and its associated reproducing kernel Hilbert space have different properties than the original kernel function and its RKHS.

Kernels can also be constructed to be invariant to arbitrary transformations, see e.g. Haasdonk and Burkhardt [2007].

2.5 INCREMENTAL LEARNING

Incremental algorithms update exactly an existing solution to a computational problem for one or several additional data examples. The goal is that specialized algorithms will achieve greater time or memory performance than repeated application of batch procedures. There are many use cases for incremental versions of batch algorithms, for example when memory capacity is constrained, since an incremental algorithm often does not need to keep the full dataset in memory, or when data examples arrive sequentially in time, termed streaming data, and a solution is desired for each additional data example.

Iterative algorithms produce a sequence of improving approximate solutions

that converge to the exact solution as the number of steps increases [Golub and Van Loan, 2013]. An iterative algorithm can often be made to operate efficiently in an incremental fashion, by expanding the data set with additional data examples and restarting the iterative procedure with the previous solution as the new initial guess for the iterative algorithm. Since a small number of additional data examples can be assumed to only slightly affect the existing solution, convergence to the new solution will be fast.

One typical example of an incremental algorithm is the rank one perturbation of the matrix inverse, or equivalently the rank one perturbation of the solution to a linear system of equations. When the matrix in question is a covariance matrix used in linear regression it is also known as recursive least squares. Given a matrix A with inverse A^{-1}, the inverse of $A + uv^T$ is given by [Heath, 2002]

$$(A + uv^T)^{-1} = A^{-1}\left(I - \frac{uv^T A^{-1}}{1 + v^T A^{-1} u}\right) \tag{2.12}$$

The above equation is known as the Sherman-Morrison formula. It does not require access to the original matrix A and its time complexity is $\mathcal{O}(n^2)$. This formula is used in the next chapter in an algorithm to select which data points to include in the Nyström subset.

2.5.1 Complexity of incremental learning

There is always a naïve online algorithm, obtained through sequential application of a batch algorithm. A naïve online algorithm thus provides an upper bound on the time complexity (whether worst-case or average or something else) of the best online algorithm. Likewise, the best batch algorithm provides a lower bound on the time complexity of any online algorithm, since a batch algorithm can be constructed from an online algorithm by ignoring the intermediate solutions. The best batch algorithm is often known or conjectured.

Let the best batch algorithm have time complexity $\mathcal{O}(f(n))$ with constant C_f and the best online algorithm $\mathcal{O}(g(n))$ with constant C_g. Then we have the following trivial bound

$$\mathcal{O}(nf(n)) \geq \mathcal{O}(g(n)) \geq \mathcal{O}(f(n))$$

with constants $\frac{1}{2}C_f, C_g, C_f$, where $\mathcal{O}(\cdot)$ refers to Big-O notation [Cormen, 2009]. It can be illustrative to compare an average empirical running time of an online algorithm to these bounds.

2.5.2 Online learning

In online learning the data also arrives one data point at a time, but the goal is not to reproduce exactly a batch algorithm, as is the case for incremental learning, but to minimize the loss between the solution at each additional data point and a true unknown solution [Foster et al., 2018].

If T data points $\{x_t\}_{t=1}^{T}$ arrive sequentially in time, and a statistical model produces output $\hat{y}$ at each time step t, then the goal is to minimize

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T} L(\hat{y}_t, y_t) \tag{2.13}$$

where y_t is the true output at each time step and L is a loss function. The cumulative loss $\mathcal{L}(\hat{y}, y)$ is also termed the *regret*.

2.6 RESAMPLING METHODS

Resampling methods create new data sets by repeatedly sampling data from the original data set [James et al., 2013]. The most well-known ones are *Cross-Validation* and the *Bootstrap*. We employ cross-validation to select the number of principal components to retain for kernel PCA in Chapter 5. The Bootstrap

has been used to similar ends as our confidence bound in Chapter 6, to measure the accuracy of functional PCA.

2.6.1 Cross-Validation

Cross-validation is a popular technique for estimating the out-of-sample error or prediction error of a statistical model, in other words the error of a model for a new unseen data point [Friedman et al., 2001, Arlot et al., 2010]. It works by splitting the full dataset into a number of disjoint test sets, then for each test set it estimates the statistical model on the other data points and calculates the error of a model based on the test set. The mean of the errors on the different test sets are calculated to obtain an estimate of the prediction error. In *leave-one-out cross-validation* the test sets contain just one point each; in *k-fold cross-validation* there are k different test sets. The focus in the following work will be on leave-one-out cross-validation.

One issue with cross-validation is that the out-of-sample errors are not independent, since the training data overlaps. The variance of the estimate of the prediction error therefore increases and is more difficult to estimate or apply, for example in a statistical test.

Cross-validation is time consuming due to repeated re-estimations of the model, especially in the case of leave-one-out cross-validation, where the time complexity increases by a factor n, the number of data points. To make cross-validation a more tractable method, many authors have proposed approximations to the cross-validation estimates in various situations. One of the first such approximatiuon was an approximate closed-form expression for the cross-validation estimate of the ridge parameter in linear ridge regression [Golub et al., 1979].

Cross-validation has been shown to be asymptotically equivalent to the AIC

information criterion [Stone, 1977].

2.6.2 Bootstrap

The bootstrap method repeatedly draws n data points from a data set of size n uniformly with replacement, and then estimates one parameter or model for each sampled dataset [Davison and Hinkley, 1997]. The final estimate is then usually the arithmetic mean of the individual estimates.

The bootstrap is a popular method for example for creating approximate confidence intervals or for model averaging in *bagging* of decision trees [Friedman et al., 2001].

Like cross-validation the bootstrap method can be time-consuming to apply. However, both cross-validation and the boostrap are *embarrasingly parallel* and so modern parallel computing architectures can be used, for example according to the *map-reduce* pattern [McCool et al., 2012].

Chapter 3

Incremental kernel PCA

In this chapter we present a novel exact algorithm for incremental kernel PCA which compares favourably to existing methods in computational efficiency[1]. A number of previous incremental algorithms for kernel PCA have been proposed. These are either exact, in that they exactly reproduce the kernel PCA solution for an additional data example, apart from possible numerical inaccuracies, or approximations. They can either be general, in that they supply the full kernel PCA solution, or they are adapted to a specific application of kernel PCA, such as classification.

An example of an iterative method for kernel PCA that can be made to operate incrementally is the kernel Hebbian algorithm [Kim et al., 2005], based on the generalized Hebbian algorithm [Oja, 1982] applied in feature space. It is linear in memory. Time complexity is not stated and is not entirely straightforward to evaluate due to the iterative nature of the algorithm.

Another example of an exact algorithm is Hoegaerts et al. [2007]. They write the

[1]Parts of this chapter have previously been made available in Hallgren and Northrop [2018]

kernel matrix expanded with an additional data example in terms of two rank one updates, without adjusting for a change in mean, and hence propose an algorithm to update the eigenvalues and corresponding eigenvectors. The complexities in time and memory are $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ at each iteration, respectively.

The algorithm that is most comparable to ours is the one in Chin and Suter [2007], which also accounts for a change in mean. It is based on the incremental linear PCA algorithm from Lim et al. [2004]. The procedure needs to store all seen data, and is cubic and quadratic in time and memory, respectively. If one additional data example is added incrementally, and all eigenpairs are retained, it requires the eigendecomposition of an $(m+2)\times(m+2)$ matrix, the eigendecomposition of the $m \times m$ unadjusted kernel matrix, and a multiplication of two $m \times m$ matrices at each step. Since a multiplication of two $m \times m$ matrices requires $2m^3$ flops, and the state-of-the-art QR algorithm for the symmetric eigenproblem about $9m^3$ flops [Golub and Van Loan, 2013], the algorithm thus requires $20m^3$ flops to the $\mathcal{O}(m^3)$ factor. Our proposed algorithm requires $8m^3$ flops to the $\mathcal{O}(m^3)$ factor if the mean is adjusted, and $4m^3$ flops otherwise, from one multiplication of two $(m + 1) \times (m + 1)$ matrices for each rank one update. Our algorithm is thus more than twice as efficient.

We will not describe further any incremental algorithms for kernel PCA adapted to a specific task, such as classification. The interested reader can see for example Takeuchi et al. [2007]. Two examples of approximations to incremental kernel PCA are Tokumoto and Ozawa [2011] or Sheikholeslami et al. [2015]. In the former, only a subset of incoming data examples are chosen and then the full eigendecomposition is recalculated. The contribution is thus not so much an algorithm for incremental kernel PCA, but rather a procedure for selecting representative data examples to include in an updated kernel PCA solution.

Our algorithm for incremental kernel PCA is based on rank one updates to the

eigendecomposition of the kernel matrix K or the mean-adjusted kernel matrix K'. In contrast to the covariance matrix in linear PCA, the kernel matrix expands in size for each additional data point, which needs to be taken into account and the effect on the eigensystem determined.

We write the kernel matrix $K'_{m+1,m+1}$ created with $m+1$ data examples in terms of an expansion and a sequence of symmetric rank one updates to the kernel matrix $K'_{m,m}$, and apply a rank one update algorithm to the eigendecomposition of $K'_{m,m}$ to obtain the eigendecomposition for $K'_{m+1,m+1}$.

A number of algorithms have been suggested to perform rank one modification to the symmetric eigenproblem. Golub [1973] initially presented a procedure to determine the eigenvalues of a diagonal matrix updated through a rank one perturbation. Bunch et al. [1978] extended the results to the determination of both eigenvalues and eigenvectors of an arbitrary perturbed matrix, including an improved procedure to determine the eigenvalues. Stability issues in the calculation of the eigenvectors, including loss of numerical orthogonality, later motivated several improvements [Dongarra and Sorensen, 1987, Sorensen and Tang, 1991, Gu and Eisenstat, 1994].

Alternatively, one could employ update algorithms for the singular value decomposition, such as the algorithm suggested in Brand [2006] for the thin singular value decomposition. In this case one would only retain, or indeed calculate, the left-singular vectors. It is not immediately clear whether this approach would be better than applying an update algorithm for the eigendecomposition. When the eigendecomposition is desired, as is our case, it seems generally preferable to apply algorithms constructed explicitly for this purpose [Golub and Van Loan, 2013]. Sometimes algorithms for the SVD can be more numerically stable [Trefethen and Bau, 1997], which could argue in favour of algorithms for the SVD.

We use the rank one update algorithm for eigenvalues from Golub [1973] and determine the eigenvectors according to Bunch et al. [1978]. If improved stability is desired, one can employ the algorithm in Gu and Eisenstat [1994]. Numerical orthogonality can be verified in $\mathcal{O}(n^2)$ time through seeing whether UU^T is close to I. Indeed exact orthogonality is not expected when eigenvectors are calculated numerically, and one has to contend with numerical orthogonality, where the off-diagonal elements of I are zero to within machine precision. In the experiments our approach seems to be sufficiently stable and accurate for most use cases. We assume throughout that the kernel matrix remains non-singular after each update.

Since one of our aims is to apply the incremental algorithm to the Nyström approximation, which already maintains a small subset of the data points, we are mainly interested in the full eigendecomposition. Various other applications of PCA also require the full eigendecomposition, such as outlier detection [Jolliffe, 2002].

As commented on above, any incremental algorithm for the eigendecomposition of the kernel matrix can be applied where the explicit or implicit inverse of the same is required, such as kernel regression and kernel SVM, since the latter can easily be recovered from the former. Even when more efficient solution methods are available, access to the eigendecomposition can be highly useful for regularization or controlling numerical stability.

3.1 RANK ONE UPDATE ALGORITHM

If we know the eigendecomposition of $K'_{m,m} = U_m \Lambda_m U_m^T$ and write $K'_{m+1,m+1}$ in terms of an expansion and number of symmetric rank one updates to $K'_{m,m}$, we can then apply a rank one update algorithm to obtain the eigendecomposition of $K'_{m+1,m+1} = U_{m+1} \Lambda_{m+1} U_{m+1}^T$. We here describe two procedures to update

$K'_{m+1,m+1}$ from $K'_{m,m}$, with and without assuming data points to have zero mean.

3.1.1 Update procedure with centred data

We first assume the feature vectors have zero mean and present the method to update $K_{m,m}$ to $K_{m+1,m+1}$. In this case, the mean does not need to be updated for an additional data point and $K_{m,m}$ only needs to be expanded with an additional row and column. We can then devise a rank one update procedure from $K_{m,m}$ to $K_{m+1,m+1}$ in two steps. If we denote $k_{i,j} := k(x_i, x_j)$ and $a = [\ k_{1,m+1}\ \ k_{2,m+1}\ \ \cdots\ \ k_{m,m+1}\ ]^T$ and let

$$k_1 = [\ a^T\ \ \frac{1}{2}k_{m+1,m+1}\ ]^T$$

$$k_0 = [\ a^T\ \ \frac{1}{4}k_{m+1,m+1}\ ]^T$$

$$\sigma = 4/k_{m+1,m+1}$$

we have

$$\begin{aligned} K_{m+1,m+1} &= \begin{bmatrix} K_{m,m} & \mathbf{0}_m \\ \mathbf{0}_m^T & \frac{1}{4}k_{m+1,m+1} \end{bmatrix} + \sigma k_1 k_1^T - \sigma k_0 k_0^T \\ &:= K^0_{m,m} + \sigma k_1 k_1^T - \sigma k_0 k_0^T \end{aligned} \tag{3.1}$$

corresponding to an expansion of $K_{m,m}$ to $K^0_{m,m}$ and two rank one updates. Compared to the eigensystem of $K_{m,m}$, the matrix $K^0_{m,m}$ will have an additional eigenvalue $\lambda_{m+1} = \frac{1}{4}k(x_{m+1}, x_{m+1})$ and corresponding eigenvector $u_{m+1} = [\ 0\ \ 0\ \ \cdots\ \ 0\ \ 1\ ]^T$. The matrix $K^0_{m,m}$ is symmetric positive definite, since all eigenvalues are positive. It will remain symmetric positive definite after the first update, since a sum of two symmetric positive definite matrices is symmetric positive definite and $k_1 k_1^T$ is a Gram matrix, since $k_1 = A^T$ for a matrix A, and $A^T A$ is a Gram matrix by definition. The resulting matrix after the second

update will be symmetric positive definite since this holds for $K_{m+1,m+1}$. The full procedure is described in Algorithm 1.

Algorithm 1 Incremental eigendecomposition of uncentred kernel matrix

Input: Dataset $\{x_i\}_{i=1}^{m+1}$; row vector of eigenvalues L and matrix of eigenvectors U of $K_{m,m}$; kernel function $k(\cdot,\cdot)$
Output: Eigenvalues L and eigenvectors U of $K_{m+1,m+1}$

$L \leftarrow [L \quad k(x_{m+1}, x_{m+1})/4]$
$U \leftarrow \begin{bmatrix} U & 0 \\ 0 & 1 \end{bmatrix}$
sigma $\leftarrow 4/k(x_{m+1}, x_{m+1})$
$k1 \leftarrow [k(x_1, x_{m+1}) \quad k(x_2, x_{m+1}) \quad ... \quad k(x_{m+1}, x_{m+1})/2]$
$k0 \leftarrow [k(x_1, x_{m+1}) \quad k(x_2, x_{m+1}) \quad ... \quad k(x_{m+1}, x_{m+1})/4]$
$L, U \leftarrow \text{rankoneupdate}(\text{sigma}, k1, L, U)$
$L, U \leftarrow \text{rankoneupdate}(-\text{sigma}, k0, L, U)$

If we limit ourselves to kernel functions for which $k(x, x)$ is constant, the above expression simplifies. Without loss of generality, we can set $k(x, x) = 1$, to obtain

$$k_1 = [\ a^T \quad 1/2\ ]^T$$
$$k_0 = [\ a^T \quad 1/4\ ]^T$$
$$\sigma = 4$$

$$K_{m+1,m+1} = \begin{bmatrix} K_{m,m} & \mathbf{0}_m \\ \mathbf{0}_m^T & 1/4 \end{bmatrix} + \sigma k_1 k_1^T - \sigma k_0 k_0^T \tag{3.2}$$

3.1.2 Update procedure with mean adjustment

We now present a procedure for symmetric rank one updates of $K'_{m,m}$ to $K'_{m+1,m+1}$ that reflects a change in mean from an added data example. In this case, all the elements of $K'_{m,m}$ need to be adjusted in addition to the expansion with another row and column.

Our procedure works as follows. We first devise two rank one updates that adjust

the mean of $K'_{m,m}$ to the one with the additonal data example included. We then expand the resulting matrix and perform symmetric updates to set the last row and column to the required values, similarly to (3.1).

Recall from Chapter 2 that when taking the mean into account, one performs an eigendecomposition on the adjusted kernel matrix $K'_{m,m} = K_{m,m} - \mathbb{1}_m K_{m,m} - K_{m,m}\mathbb{1}_m + \mathbb{1}_m K_{m,m}\mathbb{1}_m$. The elements of $K'_{m,m}$ therefore need to be adjusted through the following formula, which reverts the subtraction of the mean of $K_{m,m}$, and subtracts the mean of $K_{m+1,m+1}$ instead

$$K''_{m,m} := (K'_{m+1,m+1})_{1:m,1:m} = K'_{m,m} + \mathbb{1}_m K_{m,m} + K_{m,m}\mathbb{1}_m - \mathbb{1}_m K_{m,m}\mathbb{1}_m$$

$$+ (-\mathbb{1}_{m+1} K_{m+1,m+1} - K_{m+1,m+1}\mathbb{1}_{m+1} + \mathbb{1}_{m+1} K_{m+1,m+1}\mathbb{1}_{m+1})_{1:m,1:m}$$

The latter six terms are all rank one matrices. The matrices

$$\mathbb{1}_m K_{m,m}$$
$$-(\mathbb{1}_{m+1} K_{m+1,m+1})_{1:m,1:m}$$

are constant along the columns, and hence their sum, and similarly for the rows of

$$K_{m,m}\mathbb{1}_m - (K_{m+1,m+1}\mathbb{1}_{m+1})_{1:m,1:m}$$

The matrices

$$\mathbb{1}_m K_{m,m}\mathbb{1}_m$$
$$(\mathbb{1}_{m+1} K_{m+1,m+1}\mathbb{1}_{m+1,m+1})_{1:m,1:m}$$

are constant along both rows and columns, where each element of $\mathbb{1}_m K_{m,m}\mathbb{1}_m$ equals the sum of all elements of $K_{m,m}$ multiplied by a factor $1/m^2$, and similarly for

$$(\mathbb{1}_{m+1} K_{m+1,m+1}\mathbb{1}_{m+1})_{1:m,1:m}$$

Consequently, all terms can be written as two rank one updates. We have

$$\begin{aligned} \mathbb{1}_m K_{m,m} - (\mathbb{1}_{m+1} K_{m+1,m+1})_{1:m,1:m} &= \mathbb{1}^{(m\times m)}_{m(m+1)} K_{m,m} - \frac{1}{m+1}\mathbf{1}_m a^T \\ K_{m,m}\mathbb{1}_m - (K_{m+1,m+1}\mathbb{1}_{m+1})_{1:m,1:m} &= K_{m,m}\mathbb{1}^{(m\times m)}_{m(m+1)} - \frac{1}{m+1} a\mathbf{1}_m^T \end{aligned} \tag{3.3}$$

where, again, $k_{i,j} = k(x_i, x_j)$ and $a = [\ k_{1,m+1} \quad k_{2,m+1} \quad \cdots \quad k_{m,m+1}\ ]^T$. Since $K_{m,m}$ is symmetric for all m, we have $\mathbb{1}_m K_{m,m} = (K_{m,m}\mathbb{1}_m)^T$ and $(\mathbb{1}_{m+1}K_{m+1,m+1})_{1:m,1:m} = (K_{m+1,m+1}\mathbb{1}_{m+1})^T_{1:m,1:m}$, and can set

$$u = \frac{1}{m(m+1)} K_{m,m}\mathbf{1}_m - \frac{1}{m+1}a + \frac{1}{2}C\mathbf{1}_m$$

$$C = -\frac{1}{m^2}\Sigma_m + \frac{1}{(m+1)^2}\Sigma_{m+1}$$

where we have denoted $\Sigma_m = \mathbf{1}_m^T K_{m,m}\mathbf{1}_m$, the sum of all elements of $K_{m,m}$, to obtain

$$\begin{aligned} K''_{m,m} &= K'_{m,m} + \mathbf{1}_m u^T + u\mathbf{1}_m^T \\ &= K'_{m,m} + \frac{1}{2}(\mathbf{1}_m + u)(\mathbf{1}_m + u)^T - \frac{1}{2}(\mathbf{1}_m - u)(\mathbf{1}_m - u)^T \end{aligned}$$

which is two symmetric rank one updates. The scalar Σ_m and the vector $K_{m,m}\mathbf{1}_m$ can easily be updated between iterations like so

$$\Sigma_{m+1} = \Sigma_m + 2a^T\mathbf{1}_m + k_{m+1,m+1}$$

$$K_{m+1,m+1}\mathbf{1}_{m+1} = [\ K_{m,m}\mathbf{1}_m + a\ ;\ a^T\mathbf{1}_m + k_{m+1,m+1}\ ]$$

We now expand $K''_{m,m}$ to $K'_{m+1,m+1}$, analogously to equation (3.1), but while taking the adjusted mean into account. The required last row and column is $(K'_{m+1,m+1})_{1:m+1,m+1}$ which is given by, with $k = (K'_{m+1,m+1})_{m+1,1:m+1} =$

$[\ a\ ;\ k_{m+1,m+1}\ ]$

$$v := k - \frac{1}{1+m}(\mathbf{1}_{m+1}\mathbf{1}_{m+1}^T k + K_{m+1,m+1}\mathbf{1}_{m+1} - \frac{1}{m+1}\Sigma_{m+1})$$

Now let

$$\begin{aligned} k_0 &= [(v)_{1:m}\ ;\ \frac{1}{2}v_{m+1}] \\ k_1 &= [(v)_{1:m}\ ;\ \frac{1}{4}v_{m+1}] \\ \sigma &= 4/v_{m+1} \end{aligned}$$

obtaining

$$K'_{m+1,m+1} = \begin{bmatrix} K''_{m,m} & \mathbf{0}_m \\ \mathbf{0}_m^T & \frac{1}{4}v_{m+1} \end{bmatrix} + \sigma k_1 k_1^T - \sigma k_0 k_0^T := K^0_{m,m} + \sigma k_1 k_1^T - \sigma k_0 k_0^T$$

We have thus devised a procedure to update $K'_{m,m}$ to $K'_{m+1,m+1}$ using four symmetric rank one updates, for which a rank one eigendecomposition update algorithm can be applied. The full procedure is described in Algorithm 2. Note that the matrix $K'_{m,m}$ or its expansion do not need to be kept in memory. The memory complexity to calculate the update terms is linear in m.

3.1.3 Rank one eigendecomposition updates

Here we describe an algorithm for updating the eigendecomposition after a rank one perturbation. Suppose we know the eigendecomposition of a symmetric matrix $A = U\Lambda U^T$. Let

$$B = U\Lambda U^T + \sigma v v^T = U(\Lambda + \sigma z z^T)U^T$$

where $z = U^T v$, and look for the eigendecomposition of $\tilde{B} = \Lambda + \sigma z z^T := \tilde{U}\tilde{\Lambda}\tilde{U}^T$ [Bunch et al., 1978]. Then the eigendecomposition of B is given by

Algorithm 2 Incremental eigendecomposition of mean-adjusted kernel matrix

Input: Dataset $\{x_i\}_{i=1}^{m+1}$; row vector of eigenvalues L and matrix of eigenvectors U of $K_{m,m}$; kernel function $k(\cdot,\cdot)$; sum of all elements of $K_{m,m}$, denoted S; $K1 \leftarrow K_{m,m}\mathbf{1}_m$

Output: Eigenvalues L and eigenvectors U of $K_{m+1,m+1}$

$a \leftarrow [k(x_1, x_{m+1}) \quad k(x_2, x_{m+1}) \quad \ldots \quad k(x_m, x_{m+1})]$
$k \leftarrow [a; k(x_{m+1}, x_{m+1})]$
$S2 \leftarrow S + 2 * \mathrm{sum}(a) + k$
$C \leftarrow -S/m^2 + S2/(m+1)^2$
$u \leftarrow K1/(m*(m+1)) - a/(m+1) + 0.5 * C * \mathrm{ones}(m)$
$L, U \leftarrow \mathrm{rankoneupdate}(0.5, 1+u, L, U)$
$L, U \leftarrow \mathrm{rankoneupdate}(-0.5, 1-u, L, U)$
$K1 \leftarrow [K1 + a \quad \mathrm{sum}(a) + k]$
$S \leftarrow S2$
$m \leftarrow m + 1$
$v \leftarrow k - (\mathrm{ones}(m) * (\mathrm{sum}(a) + k) + K1 - S/m)/m$
$v0 \leftarrow v[m]$
$v \leftarrow v[1 : m-1]$
$L \leftarrow [L \quad v0/4]$
$U \leftarrow \begin{bmatrix} U & 0 \\ 0 & 1 \end{bmatrix}$
$\mathrm{sigma} \leftarrow 4/v0$
$k1 \leftarrow [v \quad v0/2]$
$k0 \leftarrow [v \quad v0/4]$
$L, U \leftarrow \mathrm{rankoneupdate}(\mathrm{sigma}, k1, L, U)$
$L, U \leftarrow \mathrm{rankoneupdate}(-\mathrm{sigma}, k0, L, U)$

$U\tilde{U}\tilde{\Lambda}\tilde{U}^T U^T$ with unchanged eigenvalues and eigenvectors $U^B := U\tilde{U}$, since the product of two orthogonal matrices is orthogonal and since the eigendecomposition is unique, provided all eigenvalues are distinct.

The eigenvalues of $\tilde{B}$ can be calculated in $\mathcal{O}(m^2)$ time by finding the roots of the secular equation Golub [1973]

$$\omega(\tilde{\lambda}) := 1 + \sigma \sum_{i=1}^{m} \frac{z_i^2}{\lambda_i - \tilde{\lambda}} \tag{3.4}$$

The eigenvalues of the modified system are subject to the following bounds

$$\begin{aligned}
&\lambda_i \leq \tilde{\lambda}_i \leq \lambda_{i+1} && i = 1, 2, ..., m-1, && \sigma > 0 \\
&\lambda_m \leq \tilde{\lambda}_m \leq \lambda_m + \sigma z^T z && && \sigma > 0 \\
&\lambda_{i-1} \leq \tilde{\lambda}_i \leq \lambda_i && i = 2, 3, ..., m, && \sigma < 0 \\
&\lambda_1 + \sigma z^T z \leq \tilde{\lambda}_1 \leq \lambda_1 && && \sigma < 0
\end{aligned} \tag{3.5}$$

which can be used to supply initial guesses for the root finding algorithm. Note that after expanding the eigensystem, as described above, the eigenpairs need to be reordered for the bounds to be valid.

Once the updated eigenvalues are calculated the eigenvectors of the perturbed matrix B are given by, with $D_i := \Lambda - \tilde{\lambda}_i I$ [Bunch et al., 1978]

$$u_i^B = \frac{U D_i^{-1} z}{\|D_i^{-1} z\|} \tag{3.6}$$

Since U and D_i^{-1} are $m \times m$ and D_i is diagonal the denominator is $\mathcal{O}(m)$ and the numerator is $\mathcal{O}(m^2)$, leading to $\mathcal{O}(m^3)$ time complexity to update all eigenvectors. This is the same complexity as an efficient batch eigenvalue algorithms for both eigenvalues and eigenvectors, but with a lower constant. The above equation can also be rewritten to update all eigenvectors at once

using one matrix multiplication of two $n \times n$ matrices, which is how it is done in the experiments in section 3.5. Such a matrix multiplication is known to require $2m^3 + \mathcal{O}(m^2)$ flops using the standard method of calculating matrix multiplications.

There exist asymptotically faster algorithms for matrix multiplication that obtain better worst case time complexity than $\mathcal{O}(n^3)$. Notable examples are the Strassen algorithm [Strassen, 1969], which is $\mathcal{O}(n^{2.8})$, or the Coppersmith-Winograd algorithm [Coppersmith and Winograd, 1987], which is $\mathcal{O}(n^{2.375})$. However, these are only useful for very large matrices and seem not to be applied often in practice.

3.2 DECREMENTAL LEARNING

Based on the above algorithms, in this section we describe an efficient procedure for removing data points from an existing kernel PCA solution, or equivalently from an existing eigendecomposition for the kernel matrix. One use case of note for a decremental procedure is leave-one-out (LOO) cross-validation [Friedman et al., 2001], where each data point is in turn removed from the training dataset, the statistical method in question is reestimated, and then the excluded data point is used for validation. An incremental algorithm for kernel SVM was proposed in Cauwenberghs and Poggio [2001], not dissimilar to the method we have proposed above, but for kernel SVM instead of kernel PCA. The method was extended to a decremental procedure to remove the effect of individual data points, in order that leave-one-out cross-validation could efficiently be performed. Rank "downdates" for the eigendecomposition of the covariance matrix in linear PCA, for purposes of leave-one-out cross-validation, was studied in Mertens et al. [1995], with applications to principal components regression. Rank downdates have not previously been applied to the cross-validation of kernel PCA.

3.2.1 Downdating procedure

We present an efficient procedure for the removal of effects of individual data points on the solution of kernel PCA, in other words a procedure for arriving at the eigendecomposition of the kernel matrix $K_{m,m}$ from the eigendecomposition of the kernel matrix $K_{m+1,m+1}$, without having to recalculate the eigendecomposition of $K_{m,m}$. Our procedure consists of a sequence of rank one modifications to the eigendecomposition of $K_{m+1,m+1}$ and a deletion of a row and a column. The deletion and modifications proceed in exact reverse to the expansion and modifications of the incremental procedures presented above. The same algorithm for the rank one modification of the symmetric eigenproblem described above can be applied.

Downdate procedure with centred data

Recall from the previous section that

$$K_{m+1,m+1} = K^0_{m,m} + \sigma k_1 k_1^T - \sigma k_0 k_0^T$$

where

$$K^0_{m,m} = \begin{bmatrix} K_{m,m} & \mathbf{0}_m \\ \mathbf{0}_m^T & \frac{1}{4}k_{m+1,m+1} \end{bmatrix} \tag{3.7}$$

and

$$k_1 = [\ a^T \ \ \frac{1}{2}k_{m+1,m+1}\ ]^T$$

$$k_0 = [\ a^T \ \ \frac{1}{4}k_{m+1,m+1}\ ]^T$$

$$\sigma = 4/k_{m+1,m+1}$$

Applying the updates in reverse, we obtain

$$K^0_{m,m} = K_{m+1,m+1} - \sigma k_1 k_1^T + \sigma k_0 k_0^T$$

Applying the rank one update algorithm described in section 3.1.3 to these rank one modifications of $K_{m+1,m+1}$ we obtain the eigenvalues and eigenvectors of the matrix $K^0_{m,m}$. To obtain the eigenvalues and eigenvectors of the matrix $K_{m,m}$ we remove the eigenvalue equal to $k_{m+1,m+1}/4$ and the row and column of the eigenvector matrix of $K^0_{m,m}$ which contains all zeros but for the element $k_{m+1,m+1}/4$. The procedure is described in Algorithm 3, where the notation $L[1:m]$ denotes selecting the first m elements of the vector L, and $U[1:m, 1:m]$ denotes selecting the first m rows and columns of the matrix U.

Above we have removed the effect of the data point x_{m+1} from the eigendecomposition. We may remove any data point x_i from the kernel matrix and eigendecomposition by instead setting the factors k_0 and k_1 to

$$k_1 = [\ k_{i,1}\ \ ...\ k_{i,i-1}\ \ \frac{1}{2}k_{i,i}\ \ k_{i,i+1}\ ...\ \ k_{i,m+1}\ ]^T$$

$$k_0 = [\ k_{i,1}\ \ ...\ k_{i,i-1}\ \ \frac{1}{4}k_{i,i}\ \ k_{i,i+1}\ ...\ \ k_{i,m+1}\ ]^T$$

Algorithm 3 Decremental eigendecomposition of uncentred kernel matrix

Input: Dataset $\{x_i\}_{i=1}^{m+1}$; row vector of eigenvalues L and matrix of eigenvectors U of $K_{m+1,m+1}$; kernel function $k(\cdot,\cdot)$
Output: Eigenvalues L and eigenvectors U of $K_{m,m}$

sigma $\leftarrow 4/k(x_{m+1}, x_{m+1})$
$k1 \leftarrow [k(x_1, x_{m+1})\ \ k(x_2, x_{m+1})\ \ ...\ \ k(x_{m+1}, x_{m+1})/2]$
$k0 \leftarrow [k(x_1, x_{m+1})\ \ k(x_2, x_{m+1})\ \ ...\ \ k(x_{m+1}, x_{m+1})/4]$
$L, U \leftarrow \text{rankoneupdate}(-\text{sigma}, k1, L, U)$
$L, U \leftarrow \text{rankoneupdate}(\text{sigma}, k0, L, U)$
$L \leftarrow L[1:m]$
$U \leftarrow U[1:m, 1:m]$

Downdate procedure with mean adjustment

As was the case for centred data, the downdating procedure with mean-adjusted data follow exactly in reverse the updating procedure with mean adjustment presented above. We describe the procedure to go from the adjusted kernel

matrix $K'_{m+1,m+1}$, obtained from $m+1$ data points, to the adjusted kernel matrix calcuated from m data points $K'_{m,m}$. The procedure consists as before of a series of rank one modifications to which a rank one update algorithm can be applied, as well as a contraction.

First we apply two rank one updates to obtain the eigendecomposition of the matrix $K^0_{m,m}$ defined in section 3.1.2 from the matrix $K'_{m+1,m+1}$ through

$$K^0_{m,m} = K'_{m+1,m+1} - \sigma k_1 k_1^T + \sigma k_0 k_0^T$$

where σ, k_1 and k_2 are as in section 3.1.2. We now remove the rows and columns of the the eigendecompostion of $K^0_{m,m}$ corresponding to the eigenvalue $v_{m+1}/4$ to obtain the the eigendecomposition of $K''_{m,m}$. Then we again apply two rank one modifications to arrive at the desired eigendecomposition of the matrix $K'_{m,m}$, through

$$K'_{m,m} = K''_{m+1,m+1} - \frac{1}{2}(\mathbf{1}_m + u)(\mathbf{1}_m + u)^T + \frac{1}{2}(\mathbf{1}_m - u)(\mathbf{1}_m - u)^T$$

where again the vector u is given in section 3.1.2.

If several iterations are being run, the quantity Σ_m can be updated as follows

$$\Sigma_m = \Sigma_{m+1} - 2a^T \mathbf{1}_m - k_{m+1,m+1}$$

and the vector $K_{m,m}\mathbf{1}_m$ may be obtained from $K_{m+1,m+1}\mathbf{1}_{m+1}$ by removing the last element and subtracting the vector a, defined above.

We describe one iteration of the procedure in Algorithm 4 below.

Algorithm 4 Decremental eigendecomposition of adjusted kernel matrix

Input: Dataset $\{x_i\}_{i=1}^{m+1}$; row vector of eigenvalues L and matrix of eigenvectors U of $K_{m+1,m+1}$; kernel function $k(\cdot,\cdot)$; sum of all elements of $K_{m+1,m+1}$, denoted S; $K1 \leftarrow K_{m+1,m+1}\mathbf{1}_{m+1}$

Output: Eigenvalues L and eigenvectors U of $K_{m,m}$

$a \leftarrow [k(x_1, x_{m+1}) \quad k(x_2, x_{m+1}) \quad \ldots \quad k(x_m, x_{m+1})]$
$k \leftarrow [a; k(x_{m+1}, x_{m+1})]$
$v \leftarrow k - (\text{ones}(m+1) * (\text{sum}(a) + k) + K1 - S/(m+1))/(m+1)$
$v0 \leftarrow v[m+1]$
$\text{sigma} \leftarrow 4/v0$
$k1 \leftarrow [v[1:m] \quad v0/2]$
$k0 \leftarrow [v[1:m] \quad v0/4]$
$L, U \leftarrow \text{rankoneupdate}(-\text{sigma}, k1, L, U)$
$L, U \leftarrow \text{rankoneupdate}(\text{sigma}, k0, L, U)$
$L \leftarrow L[1:m]$
$U \leftarrow U[1:m, 1:m]$
$K1 \leftarrow K1[1:m]$
$a \leftarrow k[1:m]$
$S2 \leftarrow S - 2 * \text{sum}(a) - k$
$C \leftarrow -S/(m+1)^2 + S2/m^2$
$K1 \leftarrow K1 - a$
$u \leftarrow K1/(m * (m+1))^2 - a/(m+1) + 0.5 * C * \text{ones}(m)$
$L, U \leftarrow \text{rankoneupdate}(-0.5, 1 + u, L, U)$
$L, U \leftarrow \text{rankoneupdate}(0.5, 1 - u, L, U)$

3.3 INCREMENTAL NYSTRÖM METHOD

In this section we describe an algorithm for incremental calculation of the Nyström approximation, when the mean is not adjusted, and for the Nyström approximation to kernel PCA with adjustment of the mean of the feature vectors. To the best of our knowledge, this is the first incremental algorithm for the full Nyström approximation to the kernel matrix. Furthermore, we believe this is the first time that the Nyström method has been suggested to provide approximations to kernel PCA with mean-adjustment of the kernel matrix, regardless of whether it is calculated incrementally or in a batch fashion.

Having access to an incremental procedure for the Nyström approximation or the Nyström solution to kernel PCA can be highly useful. Different sizes of subsets used in the approximation can efficiently be evaluated, to determine a suitable size for the problem at hand or for empirical investigation of the characteristics of the Nyström method for subsets of different sizes. For very large datasets, the combination of the Nyström method with incremental calculation results in further gains in memory efficiency.

Rudi et al. [2015] previously proposed an incremental algorithm for the Nyström approximation when applied to kernel ridge regression, based on rank one updates to the Cholesky decomposition. The time complexity of a Cholesky update is $\mathcal{O}(m^2)$, and their incremental algorithm could be extended to also compute the inverse in the usual way, through a number m of forward and backwards substitutions, leading to $\mathcal{O}(m^3)$ complexity for each update, before then creating the Nyström approximation. However, in this case it is more efficient to use the Sherman-Morrison formula (2.12).

The Nyström method provides approximations to the m largest eigenvalues and corresponding eigenvectors of the kernel matrix K or K'. In the latter case, one

obtains an approximate solution for kernel PCA. If the mean is not adjusted, one obtains an approximation of the eigendecomposition of K, which can be applied more widely to other kernel methods.

From an incremental algorithm for the eigendecomposition of the kernel matrix $K_{m,m}$, an incremental procedure for the Nyström method can be obtained by calculating the Nyström approximation to the eigenpairs at each iteration. Recall that given the eigenvalues λ_i and eigenvectors u_i of the matrix $K_{m,m}$, the corresponding approximate eigenvalues and eigenvectors of K are given by

$$\begin{aligned} \lambda_i^{nys} &:= \frac{n}{m}\lambda_i \\ u_i^{nys} &:= \sqrt{\frac{m}{n}}\frac{1}{\lambda_i}K_{n,m}u_i \end{aligned} \tag{3.8}$$

Rewriting these expressions in matrix form we obtain

$$\begin{aligned} \Lambda^{nys} &:= \frac{n}{m}\Lambda \\ U^{nys} &:= \sqrt{\frac{m}{n}}K_{n,m}U\Lambda^{-1} \end{aligned} \tag{3.9}$$

The Nyström method approximates the full kernel matrix K through $\tilde{K} = U^{nys}\Lambda^{nys}(U^{nys})^T$. Note that these are not the eigenpairs of $\tilde{K}$, and the approximate eigenvectors are not orthogonal, but they are only approximations to the eigenpairs of the full kernel matrix K.

If one is instead looking to approximate the eigendecomposition of the adjusted

kernel matrix K' the equations become

$$\begin{aligned} \lambda_i^{nys} &:= \frac{n}{m}\lambda_i \\ u_i^{nys} &:= \sqrt{\frac{m}{n}}\frac{1}{\lambda_i}K'_{n,m}u_i \end{aligned} \tag{3.10}$$

for the individual eigenvalues and eigenvectors, or in matrix form

$$\begin{aligned} \Lambda^{nys} &:= \frac{n}{m}\Lambda \\ U^{nys} &:= \sqrt{\frac{m}{n}}K'_{n,m}U\Lambda^{-1} \end{aligned} \tag{3.11}$$

where Λ, U are now the eigenvalues and eigenvectors of the matrix $K'_{m,m}$, and

$$K'_{n,m} = K_{n,m} - K\mathbb{1}_n^{(n\times m)} - \mathbb{1}_n K_{n,m} + \mathbb{1}_n K\mathbb{1}_n^{(n\times m)}$$

obtained by truncating K', where $\mathbb{1}_n^{(n\times m)}$ is an $n \times m$ matrix with each element equal to $\frac{1}{n}$. The second and last term involve summing all elements of K, hence are $\mathcal{O}(n^2)$ and we suggest instead to use the approximations

$$\mathbb{1}_n K\mathbb{1}_n^{(n\times m)} \approx \mathbb{1}_m^{(n\times m)} K_{m,n}\mathbb{1}_n^{(n\times m)}$$

$$K\mathbb{1}_n^{(n\times m)} \approx K_{n,m}\mathbb{1}_m^{(m\times m)}$$

which only involves summing the elements of $K_{n,m}$ and is $\mathcal{O}(nm)$, where similarly $\mathbb{1}_m^{(n\times m)}$ is an $n \times m$ matrix with each element equal to $\frac{1}{m}$ and $\mathbb{1}_m^{(m\times m)}$ is an $m \times m$ matrix with each element equal to $\frac{1}{m}$. This approximation is also used in Chapter 6 when calculating the PCA reconstruction error.

Our incremental algorithm for kernel PCA with the Nyström method thus corresponds to first a series of rank one updates to the eigendecomposition of $K'_{m,m}$

or $K_{m,m}$ followed by a rescaling at each step to obtain the approximation. The rescaling has $\mathcal{O}(m^2 n)$ time complexity from the matrix product in (3.9). The entire incremental rescaling procedure has complexity $\mathcal{O}(m^3 n)$, where m now denotes the maximum number of data examples in the approximation.

3.4 INCREMENTAL HYPERPARAMETER SELECTION

Approximate kernel methods introduce an additional hyperparameter that governs the accuracy of the approximation, where improved accuracy is obtained at the expense of increasing time and memory requirements. The direct determination of this hyperparameter does not seem to have been extensively studied in the literature, one of the few such studies being Rudi et al. [2015].

For subset methods this additional parameter is the number of basis elements from which to construct the approximation. In the case of the Nyström method, it corresponds to the number of randomly sampled data examples m.

One way to guide the choice of the hyperparameter is to use a bound on the statistical accuracy of the approximation, and set its value based on a desired minimum level of accuracy. For the Nyström method various such bounds have been derived under varying assumptions on the data-generating process [Drineas and Mahoney, 2005, Drineas et al., 2012, Rudi et al., 2015, El Alaoui and Mahoney, 2014, Bach, 2013, Gittens and Mahoney, 2016, Wang and Zhang, 2013, 2014, Cohen et al., 2015, Kumar et al., 2012, Cortes et al., 2010, Jin et al., 2013, Yang et al., 2012, Alaoui and Mahoney, 2014].

Occasionally, the assumptions required by the available statistical bounds are deemed too strict for the problem at hand, or other considerations make the application of such bounds impractical. Instead, one may incrementally update the approximate solution and empirically evaluate when sufficient accuracy

appears to have been achieved. In the case of randomized methods, an empirical method of evaluation can also naturally adapt to a specific subsample obtained.

We here explain one way to do this for the Nyström method, which relies on incrementally expanding the Nyström subset until the method is deemed sufficiently accurate. Measuring the accuracy by comparing the original matrix K directly to the approximate reconstruction $\tilde{K}$, through some matrix norm, is not computationally efficient since it requires $\mathcal{O}(n^2)$ comparisons at each iteration. Even reconstructing the approximation at every step can become intractable if the size of the full data set n is large. A more efficient way is therefore required.

We use the reconstruction error of an additional data point with respect to the eigenspace of the existing subset to measure how each additional observation affects the existing solution, and expand the subset while this is large. By the iid assumption, when successive observations start having limited effect on the solution, it is likely that the same holds for the remaining observations as well. Since the Nyström kernel matrix can be constructed by first projecting all data points onto the subset and then calculating the outer product, using only the subset is equivalent to using the eigenspace of the full Nyström matrix.

The use of the reconstruction error in this setting can be motivated from the perspective of robust statistics, which studies the influence of observations on the solution of a statistical model, with a view to limiting outsize influence of individual observations [Huber, 2011]. Here, we apply a measure of influence in the reverse sense, by including data examples that have a high influence on the solution. In robust PCA one wishes to determine which observations are influential with respect to either a set of eigenvalues, eigenvectors, or both [Jolliffe, 2002]. Often the reconstruction error is used, sometimes passed through an influence function [Li, 2004, De La Torre and Black, 2003].

Robust statistics can also be described as the study of a statistical model's sensitivity to departures from its assumptions, including distributional assumptions. In our case, at each iteration the model consists of the m data examples already observed but departures from the model indicate instead that the model does not accurately represent the full data and that it should be expanded.

In regression, the influence of observations is often measured through their statistical leverage scores. The leverage score of observation i is $\ell_i = \|U_{(i)}\|_2^2$, the squared 2-norm of the ith row of the matrix of left singular vectors [Drineas et al., 2012, Mahoney, 2011]. This corresponds to the ith row of the matrix U of eigenvectors of K. The statistical leverage scores, or approximations thereof, have been used successfully to guide the choice of basis points in the Nyström approximation, by adjusting the sampling distribution in accordance with their magnitude [El Alaoui and Mahoney, 2014]. There is thus some precedent for using influence measures to guide the choice of basis elements.

The reconstruction error has also been used as a novelty measure in novelty detection for kernel methods, also known as anomaly detection or one-class classification [Hoffmann, 2007]. For us, a *novelty* or *anomaly* means that the data point should be included in the subset, since the existing solution does not adequately describe all the data.

The procedure above to select the size of the subset by incrementally calculating the reconstruction error is not limited to kernel PCA or the Nyström method, but can be applied to any approximate method that uses a random subset of data points, analogous to PCA being a useful technique to determine the influence of observations regardless of the ultimate goal of the statistical analysis.

One question that arises when applying the above procedure is at what level of the reconstruction error to conclude that the subset accurately captures the

full dataset. Incremental calculation of the reconstruction error may provide additional and better guidance to a practitioner in deciding an appropriate value for this parameter rather than only relying on existing approaches. It may be more interpretable than only relying on the parameter m itself, and its statistical properties could be more easily analyzed. In this section we are not delving further into at which exact level of the reconstruction error that the subset is of a sufficient size, but in the next chapter we develop a statistical hypothesis test based on the reconstruction error that gives a conclusive answer to when additional data points should be added to the Nyström subset.

3.4.1 Residual error in feature space

To apply the incremental procedure described above we wish to calculate the residual error of a data example in feature space, when that data example is excluded from the sample. The residual error equals the projection of a data example on the orthogonal complement of an existing eigenspace. Since $\text{span}\{\Phi^T\} = \text{span}\{V\}$, with Φ and V as above, we can calculate the squared residual error in feature space by calculating the squared 2-norm of the projection of a data example ϕ_{m+1} on the orthogonal complement of the span of previous data examples $\phi(x_1), \phi(x_2), ..., \phi(x_m)$.

We write the expanded kernel matrix as

$$K_{m+1,m+1} = \begin{bmatrix} K_{m,m} & \mathbf{u} \\ \mathbf{u}^T & k \end{bmatrix} \tag{3.12}$$

The projection of ϕ_{m+1} on the orthogonal complement of $\text{span}\{\Phi_m^T\}$ is given by

$$P_{(\Phi_m^T)^\perp}\phi_{m+1} = (I - \Phi_m^+\Phi_m)\phi_{m+1} = \phi_{m+1} - \Phi_m^+\Phi_m\phi_{m+1} \tag{3.13}$$

where $+$ denotes the Moore-Penrose pseudo-inverse and P_A the projection matrix

onto the column span of a matrix A. The squared 2-norm is given by

$$(P_{(\Phi_m^T)^\perp}\phi_{m+1})^T P_{(\Phi_m^T)^\perp}\phi_{m+1} = (\phi_{m+1} - \Phi_m^+\Phi_m\phi_{m+1})^T(\phi_{m+1} - \Phi_m^+\Phi_m\phi_{m+1})$$

$$= \phi_{m+1}^T\phi_{m+1} - \phi_{m+1}^T\Phi_m^T(\Phi_m^+)^T\phi_{m+1}$$

$$- \phi_m^T\Phi_m^+\Phi_m\phi_{m+1} + \phi_{m+1}^T\Phi_m^T(\Phi_m^+)^T\Phi_m^+\Phi_m\phi_{m+1}$$

$$= \phi_{m+1}^T\phi_{m+1} - 2\phi_m^T\Phi_m^+\Phi_m\phi_{m+1} + \phi_{m+1}^T\Phi_m^T(\Phi_m^+)^T\Phi_m^+\Phi_m\phi_{m+1}$$

since $\Phi_m^+\Phi_m$ is symmetric. Furthermore, $\Phi_m^+\Phi_m$ is a projection matrix, hence idempotent, and can be replicated in the second term, yielding

$$\begin{aligned}\|P_{(\Phi_m^T)^\perp}\phi_{m+1}\|_2^2 &= \phi_{m+1}^T\phi_{m+1} - \phi_{m+1}^T\Phi_m^T(\Phi_m^+)^T\Phi_m^+\Phi_m\phi_{m+1}\\ &= \phi_{m+1}^T\phi_{m+1} - \phi_{m+1}^T\Phi_m^T(\Phi_m\Phi_m^T)^+\Phi_m\phi_{m+1}\\ &= k - \mathbf{u}^T K_{m,m}^+\mathbf{u} = k - \mathbf{u}^T K_{m,m}^{-1}\mathbf{u}\end{aligned}$$

The quantity $k - \mathbf{u}^T K_{m,m}^{-1}\mathbf{u}$ is the Schur complement of the block $K_{m,m}$ of $K_{m+1,m+1}$. If we invert $K_{m+1,m+1}$ blockwise, the lower-right 1×1 block of the inverse is $(k - \mathbf{u}^T K_{m,m}^{-1}\mathbf{u})^{-1}$ (assuming it is non-singular). Inverting the eigendecomposition of $K_{m+1,m+1}$ and equating we obtain

$$k - \mathbf{u}^T K_{m,m}^{-1}\mathbf{u} = \left(\left(K_{m+1,m+1}^{-1}\right)_{m+1,m+1}\right)^{-1} = \left(\left(\sum_{i=1}^{m+1}\frac{u_i u_i^T}{\lambda_i}\right)_{m+1,m+1}\right)^{-1}$$

$$= \left(\sum_{i=1}^{m+1}\frac{((u_i)_{m+1})^2}{\lambda_i}\right)^{-1}$$

This allows us to calculate the residual error without recourse to the full eigende-

composition. Calculation of the final quantity involves $\mathcal{O}(m)$ operations, given that one has calculated the last row of the eigenvector matrix U.

Note the ressemblance of the residual error to the last statistical leverage score $\|U_{(m+1)}\|_2^2$. The residual error is the inverse statistical leverage score, after weighting by the eigenvalues, or equivalently the inverse statistical leverage score in the weighted norm defined by $\|z\|_\cdot^2 = \sum_{i=1}^{m+1} z_i^2/\lambda_i$. Through weighting each element of the row $U_{(m+1)}$ by the inverse of the corresponding eigenvalue, the residual error becomes larger for observations that have a larger effect on dimensions of the eigenspace corresponding to larger eigenvalues. The statistical leverage scores are agnostic to which dimensions of the eigenspace are affected more, thus treating different observations similar even if they have very different effects on the reconstruction.

3.4.2 Choice of data examples

The residual error can also be applied to guide the choice of which specific data examples to include in the Nyström approximation, similar to how statistical leverage scores have been applied for regression, and to kernel matching pursuit algorithms [Hussain et al., 2011]. If calculating the residual error incrementally, observations with a higher residual error, and thus a higher influence on the existing eigendecomposition, should be included more often than those with a near-zero residual error to achieve a better approximation. A (partly) deterministic version of the Nyström approximation can thus be obtained as follows. For a specific permutation of all data examples, one may then calculate incrementally the residual error of each data example with respect to those already observed, and add it to the eigendecomposition only if it is above a certain judiciously chosen level. Whilst randomness is inherent in the specific permutation used for the incremental calculation of the residual error, affecting which data examples are included, the span of the final eigendecomposition will be similar for all

permutations, since any data example not included in the final subset will have a small residual error with respect to any chosen examples.

The residual error of each candidate data point can be calculated efficiently using the formula $\|P_{(\Phi_m^T)^\perp}\phi_{m+1}\|_2^2 = k - \mathbf{u}^T K_{m,m}^{-1}\mathbf{u}$ from above, which is $\mathcal{O}(m^2)$. The inverse or eigendecomposition only needs to be updated if a data point is to be included in the subset. If only the full Nyström approximation is desired, and not the approximate eigenpairs, then the incremental procedure to select the subset can be carried out with only $\mathcal{O}(m^2)$ complexity at each iteration, by leveraging the Sherman-Morrison formula to update the inverse of $K_{m,m}$ when the Nyström subset is to be expanded. Running the whole procedure for all n data points then becomes $\mathcal{O}(nm^2)$, where m is the final size of the Nyström subset, which is the same as when the Nyström approximation is applied directly with a fixed subset.

The data examples chosen from the specific permutation of the full dataset that is being iterated over can be seen as an approximation to the subset chosen if the residual error were to be calculated for each data example with respect to all the other possible subsets. This is somewhat analogous to the approximate leverage scores calculated in Drineas et al. [2012] which are used to create a sampling distribution for the Nyström subset.

The statistical leverage scores can also be calculated in the same incremental fashion, simply by taking the 2-norm of the last row of the incrementally calculated eigenvector matrix. This thus corresponds to calculating the statistical leverage scores with respect to the data examples already seen, and much of the above discussion carries over to this setting.

3.5 NUMERICAL EXPERIMENTS

To examine how the methods presented behave on actual data we illustrate them through computer experiments. We implement and evaluate our algorithm for incremental kernel PCA, with and without adjustment of the mean of the feature vectors. We implement our proposed procedure for incrementally updating the Nyström approximation, both when used to approximate the original kernel matrix, and when one is interested in the approximate eigenpairs from the mean-adjusted kernel matrix. We also incrementally calculate the residual error, both with a view to its application to choose the number of data examples to include in the Nyström approximation, and to using it in the selection of the specific data examples to include.

In this section we perform the experiments on the `magic` and `yeast` datasets and use the radial basis functions kernel. For each dataset we set the bandwidth parameter σ to be the median of the distance between pairs of data examples, which is a commonly used heuristic. Calculating the distances between all pairs of data points is computationally expensive, so we only use a subset of data points. If the bandwidth is chosen to be too large, then the resultant kernel matrix is likely to be rank-deficient; indeed, in the limit as $\sigma \to \infty$ the kernel matrix has rank 1. Very small parameter values causes the kernel matrix to approach the identity, regardless of what dataset is used.

Source code in Python for the incremental algorithm, with or without the Nyström method, is available at `https://github.com/fredhallgren/inkpca`.

3.5.1 Incremental kernel PCA

We have previously made two assumptions about the data. First, we assume that the kernel matrix remains of full rank after each added data example. This will

always be the case in theory if data contains noise, however near numerical rank deficiency can cause issues in practice. The secular equation 3.4 may then lack the required number of roots. In this instance one can deflate the matrix (see e.g. Bunch et al. [1978] for details), but for the purposes of our experiments we have contended with excluding the specific data example from the algorithm.

In theory the matrices will remain positive definite after each expansion and update, as expounded on previously, and thus have all positive eigenvalues, however this can fail in practice. No great effort is made to mitigate the effect of this.

Numerical accuracy is generally good, whether adjusting the mean or not. A slight loss of orthogonality is discovered in the eigenvectors, as measured by how close UU^T is to the identity, particularly for mean-adjusted data that requires four updates at each step.

Every numerical operation leads to a small loss in accuracy, due to the finite representation of floating-point numbers, which is propagated, with varying severity, over subsequent operations. An incremental procedure involves substantionally more operations than a batch procedure, which leads to worse accuracy in comparison, often termed *drift*. We illustrate this by plotting the Frobenius norm of the difference between the adjusted kernel matrix calculated using the batch and incremental procedure respectively, for different number of data points. In other words, we plot $\|K'_{m,m} - U'_m \Lambda'_m U'^T_m\|_F$, where

$$\|A\|_F = \sqrt{\sum_{i=1}^{n} a_{i,j}^2}$$

Please see Figure 3.1 on page 104. The drift for reconstruction of the unadjusted matrix is smaller and is not plotted. Our results seem to indicate that the drift is small.

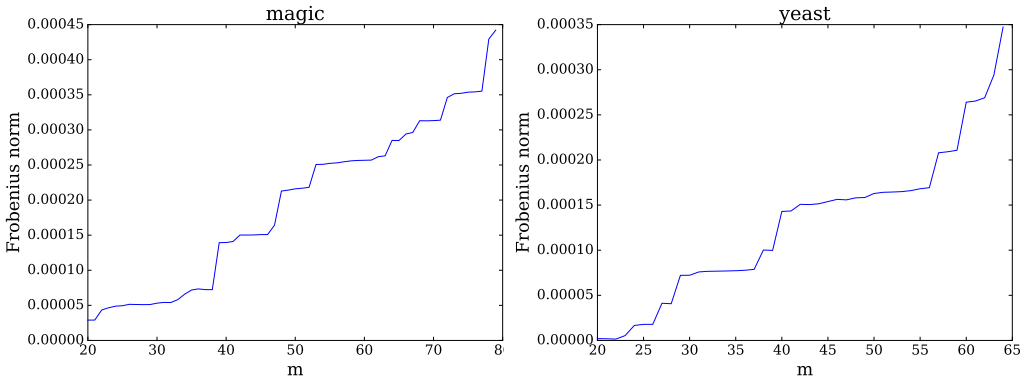

Figure 3.1: Frobenius norm of the difference between $K'_{m,m}$ and the incremental reconstruction for different values of m

There is no definitive answer to the question of when a certain algorithm is or is not numerically stable and evaluation of numerical stability is often subjective, depending on the requirements of the problem at hand [Fine and Scheinberg, 2001].

3.5.2 Incremental Nyström approximation

We run our algorithm for incremental kernel PCA with the Nyström method, both with and without mean adjustment. We run the experiments on the first 1000 observations from each dataset, for a more accurate comparison and to limit computational requirements.

We first consider the incremental Nyström approximation without adjusting the mean, i.e. for incremental updating of the Nyström approximation to the matrix K. As mentioned, this can be applied to get an approximate kernel PCA when the data examples are assumed to be centred in feature space, or when the Nyström method is to be applied more generally to kernel methods other than kernel PCA.

Having access to an incremental algorithm for calculating the Nyström approximation lets us investigate explicitly how the approximation improves with each

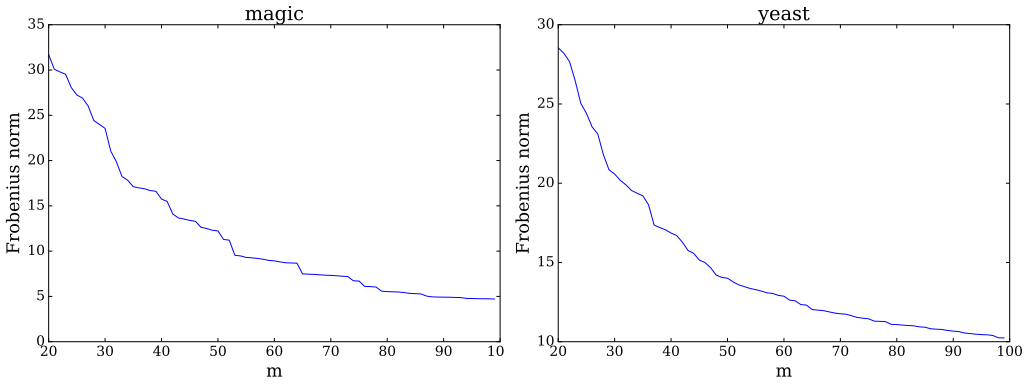

Figure 3.2: Frobenius norm of the difference between K and $\tilde{K}$ for different values of m

additional data point for a specific data set. We thus calculate the Frobenius norm of the difference between the kernel matrix and the Nyström approximation $\|K - \tilde{K}\|_F$ at each step of the algorithm. Please see Figure 3.2 on page 105.

Calculation of the Nyström method can exacerbate numerical inaccuracy in the eigenvalues when calculating the approximate eigenvectors, due to the factor Λ^{-1} in $U^{nys} \propto K_{n,m} U \Lambda^{-1}$. This leads to instability when calculating the Nyström approximation for the adjusted kernel matrix, since the adjusted covariance matrix is less well conditioned, as explained above, and the eigenvalues are smaller in general. Furthermore, the approximation of $K'_{n,m}$ leads to further inaccuracies. For the adjusted Nyström approximation we therefore employ the method suggested in Drineas and Mahoney [2005], which introduces regularization by masking of the smallest eigenvalues. We set $r = 10$, i.e. keeping the top 10 eigenvalues and corresponding eigenvectors. We plot the accuracy of the approximation below for one sample from the dataset.

We also implement the incremental Nyström procedure for an approximate kernel PCA with mean adjustment, using the approximations introduced in section 4, and again calculate the Frobenius norm of the difference between K and its

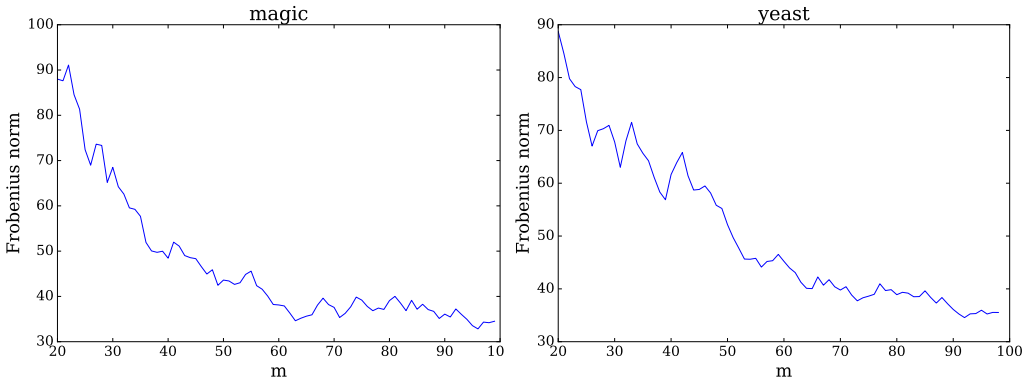

Figure 3.3: Frobenius norm of the difference between K' and $\tilde{K}'$ for different values of m

approximate reconstruction. Please see Figure 3.3 on page 106.

As seen in the plots, the Nyström approximation seems to provide a high degree of accuracy in approximating the matrices K and K' for the two datasets, even for a fairly small number of basis points. When the sample size is 100, the Frobenius error is 5 and 10 respectively, or 5×10^{-6} and 10^{-5} on average per matrix element.

Improved accuracy for the mean-adjusted approximation could also be obtained by increasing the rank parameter r when the size of the subset increases.

3.5.3 Incremental residual error

Given access to the last row of the eigenvector matrix for each additional data example as well as the updated eigenvalues, the incremental residual error is straight forward to calculate. In this section we assume that all variables have zero mean in feature space and so we don't perform a mean-adjustment.

For an individual sample, the residual error exhibits significant noise as seen in Figure 3.4 on page 107. To get a better idea of the behaviour of the residual

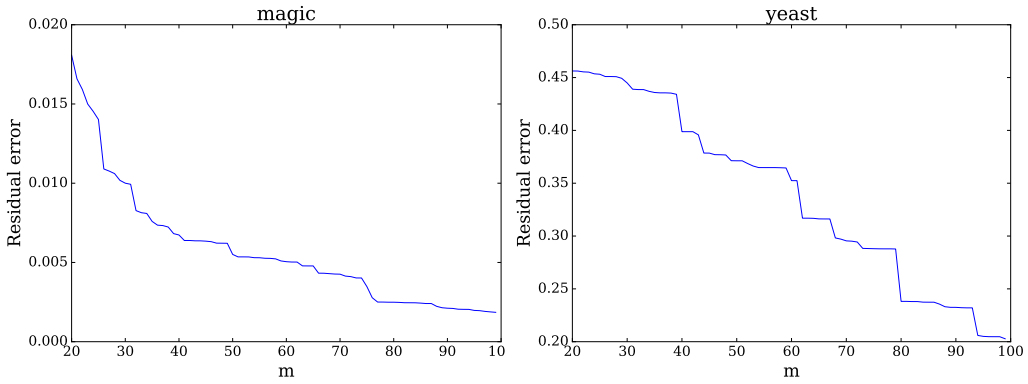

Figure 3.4: Incremental residual error over one sample

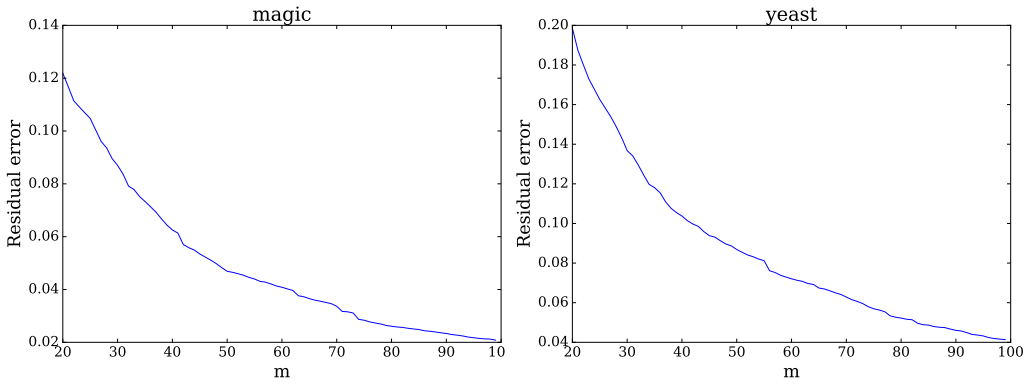

Figure 3.5: Incremental residual error over 100 samples

error in expectation, we plot the average residual error over many samples of the datasets. Please see Figure 3.5 on page 107.

As seen in the plot, the average residual error levels out, at which point additional data examples can be assumed not to meaningfully influence the solution. The exact level at which the incremental reconstruction error is small enough is perhaps best left to the discretion of a practitioner, but could for example be phrased in terms of the full variance of the existing data in the subset, which is equal to 1 in the above examples. The statistical test in the following chapter could also be applied, which recasts the question in terms of a statistical significance level.

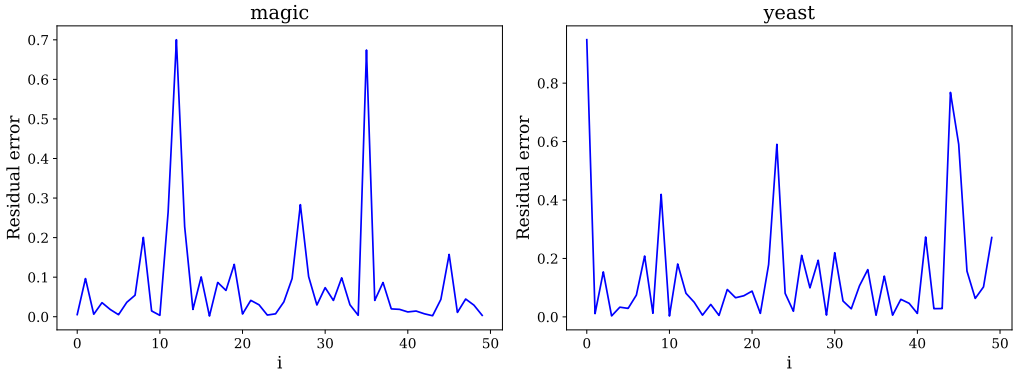

Figure 3.6: Residual error for each data point

3.5.4 Decremental learning

Experiments were also performed on the decremental algorithm, to be able to inspect its behaviour and verify its correctness. We applied it to calculate efficiently the residual error of each data point with respect to all the others. For a kernel matrix of size 50, we applied the decremental algorithm to each data point in turn and hence calculated the residual error of each data point with respect to the rest of the data points. The residual errors for one random sample of data points for each of the two datasets is plotted in Figure 3.6. The experiments are without mean-adjustment.

The residual error is slightly larger on average for the `yeast` dataset compared to the `magic` dataset, which concurs with the experimental results in the previous section.

3.6 SUMMARY

In this chapter we developed an algorithm for incremental calculation of kernel PCA, with or without allowing for a change mean, which appears more efficient than comparable algorithms already in existence. We extended the algorithm to incremental calculation of the Nyström method and reversed it to present

a decremental algorithm, which can find uses for example to perform leave-one-out cross-validation for kernel PCA. We applied the incremental algorithm to empirically determine an appropriate size of the Nyström subset, using the reconstruction error of an additional data point with respect to the subset as a measure of fidelity for the Nyström method. The proposed algorithms appear computationally efficient and numerically stable in computer experiments.

Chapter 4

Statistical testing of the Nyström method

The Nyström method is most often considered an *approximation* and is not expected to be as accurate as the corresponding full method. But can the Nyström method ever be considered the true mathematical model underlying the data? In this chapter we present assumptions on the data which lead to the Nyström method being closer to the true model for reality and hence develop a statistical hypothesis test to select the number of data points to include in the Nyström subset [Blom et al., 2005, Ramachandran and Tsokos, 2020]. As far as we know, it is the first statistical test developed to select the hyperparameter that governs the accuracy of the Nyström approximation and for the first time provides a procedure for formal decision-making in the selection of the size of the Nyström subset.

If the Nyström method captures all the important structure in the data, then the reconstruction error of all the data onto the subset will solely be due to noise. Otherwise, the reconstruction error will be larger than otherwise expected, which indicates that more data points should be added to the subset.

4.1 DATA MODEL

Every random process $Z \in L^2(\mathbb{R})$ can be decomposed in terms of uncorrelated real-valued random variables ξ_k and orthogonal basis functions ϕ_k through the *Karhunen-Loève Transform* (KLT) [Paulsen and Raghupathi, 2016]

$$Z = \sum_{k=1}^{+\infty} \xi_k \phi_k$$

The basis functions are the eigenvectors of the kernel integral operator on a compact subset of $\mathcal{X}$. As an example, a Gaussian random variable in $L^2(\mathbb{R})$ has coefficients ξ_k that are independent univariate Gaussian.

A low-rank kernel matrix corresponds to a Karhunen-Loève transform with fewer terms than the number of data points, $d < n$

$$Z = \sum_{k=1}^{d} \xi_k \phi_k$$

The observations of Z are given by, for $i = 1, 2, ..., n$

$$z_i = \sum_{k=1}^{d} \xi_{k,i} \phi_k$$

The $\xi_{k,i}$ and ϕ_k are unobserved and need to be estimated from data, and then correspond to the principal scores and principal components (before normalization). The corresponding kernel matrix can be written

$$K = U_d \Sigma_d U_d^T$$

where Σ_d contains the variances σ_k^2 of the ξ_k in the diagonal, and

$$U_d = \begin{pmatrix} \xi_{1,1} & \xi_{2,1} & \dots & \xi_{d,1} \\ \xi_{1,2} & \xi_{2,2} & \dots & \xi_{d,2} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{1,n} & \xi_{2,n} & \dots & \xi_{d,n} \end{pmatrix} \Sigma_d^{-1/2}$$

Kernel evaluation is given by

$$k(x_i, x_j) = \left\langle \sum_{k=1}^{d} \xi_{k,i}\phi_k, \sum_{k=1}^{d} \xi_{k,j}\phi_k \right\rangle_{\mathcal{H}} = \sum_{k=1}^{d} \xi_{k,i}\xi_{k,j}$$

The kernel matrix has rank d and if $m \geq d$ and the ξ_k have continuous distributions then the Nyström method is exact[1] – applying the Nyström approximation will recover the original kernel matrix. This is the case since then any z_i will lie in the span of $\{z_j\}_{j=1}^m$. Then also the empirical risk with the Nyström method will equal the empirical risk of the standard method. This holds regardless of which subset of data points is chosen and regardless of if the Nyström method is specified as in Williams and Seeger [2001] or Rudi et al. [2015].

However, the Nyström method should not need to exactly recover the original method for it to be considered the true representation of reality. If data is corrupted by unpredictable, uninformative noise, that the original method has no greater prospect of discerning than the approximation, then the approximation should still be considered true and the full model superfluous and overspecified. This leads us to the following data model, with an underlying low-rank structure and additive noise

[1] modulo numerical instabilities

$$Z = \sum_{k=1}^{d} \xi_k \phi_k + \varepsilon = \sum_{k=1}^{d} \xi_k \phi_k + \sum_{k=d+1}^{\infty} \epsilon_k \psi_k \tag{4.1}$$

where the noise processes ε_i have zero mean and are independent, and are also independent of the ξ_j or any other random variables under consideration, and the real-valued noise variables ϵ_j also have zero mean and are independent, and for which $\epsilon_k \to 0$ almost surely as $k \to \infty$, since ε is in $L^2(\mathbb{R})$.

By the assumption of a continuous data distribution all the variances σ_k^2 will be different and also different from the error variance σ_ϵ^2. This is not an unreasonable request – processes occurring in nature are unlikely to have the same variability in the error as in underlying explanatory dimensions.

Since the data points are corrupted by noise, we have no prospect of determining the basis functions exactly, with or without the Nyström method. Exact solutions to problems involving data are the purview of numerical analysis – the existence of the noise term is what separates statistics from other disciplines.

Without loss of generality we may assume that the noise variances $\{\sigma_{\epsilon_k}^2\}_{k=d+1}^{\infty}$ are monotonically decreasing – if not we may rearrange the order of the basis for the noise such that this holds. We assume that the noise variances are slowly varying, or equivalently that they are constant over a large enough number of consecutive error dimensions. This is akin to the common *homoscedasticity* assumption or white noise [Grégoire, 2010]. The noise should be fairly evenly spread out over many dimensions, or one may discern important information from the noise, unlike the characteristics of many sources of noise in nature. We want a model where the noise conveys as little information as possible and is of no use for decision-making. For the purposes of this chapter we may take

the noise variance to be constant across the first $n - m$ error dimensions, and zero otherwise. Assuming that the noise has $n - d$ dimensions ensures that the kernel matrix has full rank, a fact that will not be changed whether or not errors in the subseqeuent dimensions are zero. We only have n data points at hand, and any quantity we create from them will have at most n dimensions, so it does not really matter if the actual error dimension is larger than n, or if it's not.

We may write $L^2(\mathbb{R})$ as the direct sum of the explanatory dimensions and the nuisance dimensions through

$$L^2(\mathbb{R}) = \langle \{\phi\}_{k=1}^{d} \rangle \oplus \langle \{\psi\}_{k=d+1}^{\infty} \rangle =: \mathcal{H}_\phi \oplus \mathcal{H}_\varepsilon$$

and then the sum of the projections onto these subspaces is the identity map, $I = P_\phi + P_\varepsilon$.

Note that the above model (4.1) becomes the standard Karhunen-Loève transform if $d = n$, and this always exists. Therefore it is rather a question of the true value for the maximum number of explantory dimensions d than a question of which model is the true one. The unrestricted decomposition can be considered a special case of the above low-rank representation. It is perhaps unlikely that any natural process has an infinite number of informative underlying dimensions.

4.2 AN F-TEST

If the Nyström method with the current size of the subset is the correct model for the data, then the reconstruction error of the remaining data points is solely due to noise, and will be smaller than if important dimensions are missing from the representation. This can be used to construct a test for determining when new data points should be added to the Nyström subset.

Given the data model with a true underlying low-rank structure, we develop an F-test for testing whether the Nyström subset captures all the structure in the data, which is the case when $m \geq d$, or if the subset should be expanded. The F-test well-known and is used widely to compare model errors and variances. It is easy to apply and available in most statistical or numerical software packages.

We have the following null and alternative hypotheses

$$H_0 : \quad m = d$$

$$H_1 : \quad m < d$$

When devising a statistical test, strong conclusions can only be drawn when the null hypothesis is rejected, whereas if it is not then the verdict is only that there is insufficient evidence for rejecting the null [Sundberg, 1981]. Therefore, we have constructed the hypotheses such that when the null is rejected the conclusion will be that data points should be added to the Nyström subset. We do not want to take action and add new members to the subset unless there is strong evidence for this course of action. The null hypothesis should represent the *status quo*, the prevailing state of affairs – which is the current size of the subset.

The residual for a data point x_i with $i > m$ under H_0 is

$$z_i - P_{\mathcal{H}_m} z_i = \sum_{k=1}^{d} \xi_{k,i}\phi_k + \varepsilon_i - \sum_{k=1}^{d} \xi_{k,i} P_{\mathcal{H}_m}\phi_k + P_{\mathcal{H}_m}\varepsilon_i = \varepsilon_i - P_{\mathcal{H}_m}\varepsilon_i$$

since $\langle \{\phi_k\}_{k=1}^{d} \rangle \subset \langle \{\phi_k\}_{k=1}^{d} \rangle \oplus \langle \{\varepsilon_j\}_{j=1}^{m} \rangle = \mathcal{H}_m$.

By the assumption of constant variance of ε for a very large number of dimensions, each realization of ε will be orthogonal to the span of any small number of other dimensions, which implies $P_{\mathcal{H}_m}\varepsilon_i = \bar{0}$. Therefore, under H_0, we have $z_i - P_{\mathcal{H}_m} z_i = \varepsilon_i$, and the reconstruction error is solely due to noise. The norm

becomes

$$\|z_i - P_{\mathcal{H}_m} z_i\|^2 = \|\varepsilon_i\|^2 = \sum_{k=m+1}^{n} \epsilon_{k,i}^2$$

If we add an arbitrary data point $x_{m+1} \neq x_i$ to the subset, we still obtain

$$z_i - P_{\mathcal{H}_{m+1}} z_i = \varepsilon_i$$

This will *not* be the case if H_0 is false, a fact that can be used to create a test. If H_1 were true then adding another data point would decrease the reconstruction error. Under H_1, the reconstruction error for the current subset becomes

$$z_i - P_{\mathcal{H}_m} z_i = \varepsilon_i + (I - P_{\mathcal{H}_m}) \sum_{k=1}^{m+1} \xi_{k,i} \phi_k$$

Now the subset no longer spans the d important dimensions, but some subspace of dimension $m < d$. Adding an arbitrary point to the subset then gives either, if $m + 1 = d$

$$z_i - P_{\mathcal{H}_{m+1}} z_i = \varepsilon_i$$

or, if $m + 1 < d$

$$z_i - P_{\mathcal{H}_m} z_i = \varepsilon_i + (I - P_{\mathcal{H}_{m+1}}) \sum_{k=1}^{m+1} \xi_{k,i} \phi_k$$

both of which are less in magnitude than without the additional data example.

This suggests testing whether the error remains the same if we add an arbitrary data point to the subset. The magnitude under H_0 for a data point x_i with $i > m$ is, again

$$\|z_i - P_{\mathcal{H}_m} z_i\|_{\mathcal{H}}^2 = \|\varepsilon_i\|_{\mathcal{H}}^2 = \sum_{k=d+1}^{n} \epsilon_{k,i}^2$$

where we assumed the ϵ_k have constant variance σ_ε^2 for $k = m + 1, m + 2, ..., n$ and zero variance for $k = n + 1, n + 2, ...$ For the purposes of the test we now

assume that they are normally distributed, and since they have zero mean, their squared sum, scaled by the variances, will have a chi-squared distribution.

For a different data point $x_j, j > m$, and a different subset of size $m+1$, whose span we denote $\mathcal{H}'_{m+1}$, the corresponding quantity will be independent and also have a chi-squared distribution under H_0

$$\|z_j - P_{\mathcal{H}'_{m+1}} z_j\|^2_{\mathcal{H}} = \|\varepsilon_j\|^2_{\mathcal{H}} = \sum_{k=d+1}^{n} \epsilon^2_{k,j} \sim \sigma^2_\varepsilon \chi^2_{n-d}$$

As a test statistic we use the ratio of the reconstruction errors scaled by the error variances

$$F = \frac{\|z_i - P_{\mathcal{H}_m} z_i\|^2_{\mathcal{H}} / \sigma^2_\varepsilon}{\|z_j - P_{\mathcal{H}'_{m+1}} z_j\|^2_{\mathcal{H}} / \sigma^2_\varepsilon} = \frac{\|z_i - P_{\mathcal{H}_m} z_i\|^2_{\mathcal{H}}}{\|z_j - P_{\mathcal{H}'_{m+1}} z_j\|^2_{\mathcal{H}}}$$

which has an F-distribution with $n-m$ and $n-m$ degrees of freedom under the null hypothesis, where z_i and z_j are any two distinct data points that don't belong to either subset.

To get a better estimate of the reconstruction errors we may use the mean reconstruction error of several data points in both the numerator and denominator above, and the distribution of the test statistic will remain the same since the sum of random variables with chi-squared distributions also has a chi-squared distribution. Let I and I' be index sets over data points that belong to neither subset with $|I| = |I'| = N$ and $I \cap I' = \emptyset$. Then the test statistic becomes

$$F = \frac{\sum_{i \in I} \|z_i - P_{\mathcal{H}_m} z_i\|^2_{\mathcal{H}}}{\sum_{j \in I'} \|z_j - P_{\mathcal{H}'_{m+1}} z_j\|^2_{\mathcal{H}}}$$

which has an F-distribution with $N \cdot (n-m)$ and $N \cdot (n-m)$ degrees of freedom under the null hypothesis.

If the actual value we obtain for the test statistic is very unlikely given the assumption of the Nyström method capturing the important structure of the data, we can reject the null hypothesis and conclude that more data points should be added to the subset. If it is not, then there is insufficient evidence to support adding more data points, and to keep the model as parsimonious as possible we should keep the current number of data points.

The assumption of Gaussian errors is perhaps less restrictive in this setting than usual. The Nyström method is only ever applied when n is very large, indeed by assumption $n \gg m$, and by the Central Limit Theorem the sum of squared errors will asymptotically follow a normal distribution when $n \to \infty$, which is also the law that the chi-squared distribution approaches for a large number of degrees of freedom.

The distribution of the reconstruction error under the alternative hypothesis is impossible to determine without additional assumptions on the distribution of the data. Since the number of important dimensions may be small, a Gaussian assumption could perhaps be considered contrived.

It is possible that some of the assumptions above could be relaxed, or the test could be constructed differently, for increased generality and closer fidelity to the real world, but we contend with the present analysis for now. We also leave experimental analysis for future work.

If data points are added to the Nyström subset incrementally and the test is applied repeatedly until the null hypothesis is rejected then the p-value needs to be adjusted in line with sequential analysis [Lai, 2001]. Again, we spare further analysis of this setting for future enquiry.

The test statistic can be calculated through the following lemma. It is somewhat similar to Theorem 3 in Chapter 6.

Lemma 1. *The reconstruction error for a data point x_i with respect to the Nyström subset is given by*

$$\|z_j - P_{\mathcal{H}_m} z_j\|^2_{\mathcal{H}} = k(x_j, x_j) - \kappa(x_j)^T K^{-1}_{m,m} \kappa(x_j)$$

where $K_{m,m}$ is the kernel matrix from the subset of data points and $\kappa(x) = (k(x_1, x),\ k(x_2, x)\ ...,\ k(x_m, x))^T$.

This quantity will be zero for low-rank matrices when the Nyström method is exact.

Proof of Lemma 1. First we have

$$\|z_j - P_{\mathcal{H}_m} z_j\|^2_{\mathcal{H}} = \langle z_j - P_{\mathcal{H}_m} z_j, z_j - P_{\mathcal{H}_m} z_j \rangle_{\mathcal{H}}$$

$$= \langle z_j, z_j \rangle_{\mathcal{H}} - 2\langle z_j, P_{\mathcal{H}_m} z_j \rangle_{\mathcal{H}} + \langle P_{\mathcal{H}_m} z_j, P_{\mathcal{H}_m} z_j \rangle_{\mathcal{H}}$$

$$= \langle z_j, z_j \rangle_{\mathcal{H}} - \langle P_{\mathcal{H}_m} z_j, P_{\mathcal{H}_m} z_j \rangle_{\mathcal{H}}$$

We note that $\langle z_j, z_j \rangle_{\mathcal{H}} = k(x_j, x_j)$. Let $\{\phi_i\}_{i=1}^m$ be the eigenvectors of C_m, the empirical covariance operator using the m subset points. Then $\{\phi_i\}_{i=1}^m$ span $\mathcal{H}_m$ and

$$P_{\mathcal{H}_m} z_j = \sum_{i=1}^{m} \langle z_j, \phi_i \rangle_{\mathcal{H}} \phi_i$$

and so

$$\langle P_{\mathcal{H}_m} z_j, P_{\mathcal{H}_m} z_j \rangle_{\mathcal{H}} = \left\langle \sum_{i=1}^{m} \langle z_j, \phi_i \rangle_{\mathcal{H}} \phi_i, \sum_{k=1}^{m} \langle z_j, \phi_k \rangle_{\mathcal{H}} \phi_k \right\rangle_{\mathcal{H}}$$

$$= \sum_{i=1}^{m} \sum_{k=1}^{m} \langle z_j, \phi_i \rangle_{\mathcal{H}} \langle \phi_i, \phi_k \rangle \langle z_j, \phi_k \rangle_{\mathcal{H}} = \sum_{i=1}^{m} \langle z_j, \phi_i \rangle_{\mathcal{H}}^2$$

The eigenvectors of the empirical covariance operator C_m are given by [Bengio et al., 2004]

$$\phi_i = \frac{1}{\sqrt{\lambda_i}} u_i^T \kappa(x)$$

where $\{(\lambda_i, u_i)\}_{i=1}^m$ are the eigenpairs of $K_{m,m}$ and $\kappa(x) = (k(x_1, x),\ k(x_2, x)\ ...,\ k(x_m, x))^T$.

Then by the reproducing property

$$\left\langle z_j, \frac{1}{\sqrt{\lambda_i}} u_i^T \kappa(x) \right\rangle_{\mathcal{H}} = \frac{1}{\sqrt{\lambda_i}} u_i^T \kappa(x_j)$$

And so

$$\sum_{i=1}^{m} \langle z_j, \phi_i \rangle_{\mathcal{H}}^2 = \sum_{i=1}^{m} \kappa(x_j)^T \frac{1}{\lambda_i} u_i u_i^T \kappa(x_j) = \kappa(x_j)^T K_{m,m}^{-1} \kappa(x_j)$$

Finally we arrive at

$$\|z_j - P_{\mathcal{H}_m} z_j\|_{\mathcal{H}}^2 = k(x_j, x_j) - \kappa(x_j)^T K_{m,m}^{-1} \kappa(x_j)$$

□

4.3 SUMMARY

In this chapter we have developed a statistical hypothesis test to determine if the size of the Nyström subset is sufficiently large, based on assumptions where the Nyström method will be the true model for the data if this holds true. Similar to other tests applied to model errors or variances, we obtained an F-test, which is well-known to practitioners within statistics and data analysis and easy to apply with standard software tools.

Chapter 5

Cross-validation for kernel PCA

In this chapter we develop a cross-validation procedure which can be used to estimate the accuracy of kernel PCA, based on an existing method for cross-validation of linear PCA which we adapt to be carried out in the feature space. We calculate in the feature space the residual error of a data point with respect to a subset of principal components calculated from the rest of the data points. The residual error can also be described as the 2-norm of the projection of the excluded data point on the orthogonal complement of the subset of principal components. A cross-validation estimate of the residual error can thus be obtained by leaving out each data point in turns and calculating the principal components using the rest of the data points.

The cross-validation estimate of the residual error can be used to determine the number of principal components to retain in a low-dimensional representation of the data. Such a procedure works by calculating the average residual error across data points and selecting the number of principal components corresponding to the lowest residual error, adjusted for parsimony in the statistical model by preferring a smaller number of principal components.

5.1 CROSS-VALIDATION FOR PCA

Cross-validation is commonly applied to PCA in order to determine the number of components to retain for further analysis. It can be carried out in a number of different ways, corresponding to different sets of entries in the data matrix X that are left out at each iteration [Bro et al., 2008]. We will consider *row-wise cross-validation*, which is closest to the standard definition of cross-validation. For each number of principal components, the residual variation or residual error of one data point is calculated with respect to the top k prinicpal components calculated from the remaining data points. The squared residual variation or residual error for data point x_i with respect to the top k principal components is given by

$$e_i^2 = \|x_i - V_k^{(-i)} V_k^{(-i)T} x_i\|_2^2$$

where $V_k^{(-i)}$ are the top k eigenvectors of $X^{(-i)T} X^{(-i)}$, with $X^{(-i)}$ the data matrix excluding data point x_i. $V_k^{(-i)} V_k^{(-i)T}$ is the projection matrix onto these top k principal components.

For each number of retained principal components, the residual errors are summed to calculate the *mean predicted residual sum of squares*

$$MPRESS(k) = \frac{1}{n(d-k)} \sum_{i=1}^{n} e_i^2 = \frac{1}{n(d-k)} \sum_{i=1}^{n} \|x_i - V_k^{(-i)} V_k^{(-i)T} x_i\|_2^2$$

where, again, n is the number of data points and d is the dimension of the variables. The mean predicted sum of squares is calculated for all the principal components, and the number of total principal components corresponding to the lowest value is selected.

The mean of the errors is adjusted by the factor $1/(d-k)$, dividing by the dimension of the variables d minus the number of retained principal components k. This then increases the calculated error if more principal components are retained.

If an adjustment is not made to penalize subspaces with higher dimension then adding more principal components will always lead to a smaller error, and the maximum number of principal components would always be selected.

5.2 CROSS-VALIDATION FOR KERNEL PCA

In this section we describe how to perform row-wise cross-validation for kernel PCA, with the goal to determine the number of principal components to retain. To this end we derive the residual error in feature space with respect to the top k principal components in feature space. This residual error is similar to the one derived in section 3.4.1, but there we calculated the residual error of a data point with respect to all the other data points, not with respect to the principal components.

To the best of our knowledge, this is the first time that cross-validation of PCA is adapted to the setting of kernel PCA through the residual error in the feature space. In Alam and Fukumizu [2014] cross-validation was applied to the choice between different kernel functions, but it was based on an error in the input space, not in the feature space.

Consider the data matrix in feature space Φ which has dimension $n \times p$, where p is potentially equal to infinity. To perform row-wise cross-validation we proceed as follows. For each data point x_i we remove the data point $\Phi(x_i) =: \phi_i$ from the sample, forming the matrix $\Phi^{(-i)}$, and estimate the principal components, i.e. the eigenvectors of $\Phi^{(-i)T}\Phi^{(-i)}$. We then calculate the squared residual error e_i^2 for each data point ϕ_i

$$e_i^2 = \|\phi_i - V_k^{(-i)} V_k^{(-i)T} \phi_i\|_2^2$$

where now $V_k^{(-i)}$ is the matrix of the top k eigenvectors of $\Phi^{(-i)T}\Phi^{(-i)}$.

We now derive the residual error in feature space. Since $\mathbf{v} = \Phi^T \mathbf{u}$, where $\mathbf{v}$ is an eigenvector of C and $\mathbf{u}$ is an eigenvector of $\frac{1}{n}K$, and where $C = \frac{1}{n}\Phi^T\Phi$, we have $V_k = \Phi^T U_k$, and also $V_k^{(-i)} = \Phi^{(-i)T} U_k^{(-i)}$, where $U_k^{(-i)}$ are the top k eigenvectors of $\Phi^{(-i)}\Phi^{(-i)T}$. However, if the eigenvectors U_k have unit norm, then this does not necessarily hold for the eigenvectors V_k. Therefore, we normalize $\mathbf{v}$ through

$$\mathbf{v} = \frac{\Phi^T \mathbf{u}}{\|\Phi^T \mathbf{u}\|_2} = \frac{\Phi^T \mathbf{u}}{\sqrt{\mathbf{u}^T \Phi \Phi^T \mathbf{u}}} = \frac{\Phi^T \mathbf{u}}{\sqrt{\mathbf{u}^T K \mathbf{u}}}$$

The above holds for any of the eigenvectors $\mathbf{v}$. For a specific eigenvector $\mathbf{v}_i$ we have that

$$\mathbf{v}_i = \frac{\Phi^T \mathbf{u}_i}{\sqrt{\mathbf{u}_i^T K \mathbf{u}_i}} = \frac{\Phi^T \mathbf{u}_i}{\sqrt{\lambda_i}}$$

In matrix form we obtain the expression

$$V_k = \Phi^T U_k \Lambda_k^{-1/2}$$

In the following we dispense with the superscript $(-i)$ for clarity – for the error e_i the quantities Φ, U_k, K are calculated based on the dataset with data point i excluded. Inserting the formula for V_k into the expression for the residual error we obtain

$$e_i^2 = \left\| \phi_i - \Phi^T U_k \Lambda_k^{-1} U_k^T \Phi \phi_i \right\|_2^2$$

Expanding the norm we obtain

$$\begin{aligned} e_i^2 &= \left\| \phi_i - \Phi^T U_k \Lambda_k^{-1} U_k^T \Phi \phi_i \right\|_2^2 \\ &= \left(\phi_i - \Phi^T U_k \Lambda_k^{-1} U_k^T \Phi \phi_i \right)^T \left(\phi_i - \Phi^T U_k \Lambda_k^{-1} U_k^T \Phi \phi_i \right) \\ &= \phi_i^T \phi_i - 2\phi_i^T \Phi^T U_k \Lambda_k^{-1} U_k^T \Phi \phi_i + \phi_i^T \Phi^T U_k \Lambda_k^{-1} U_k^T \Phi \Phi^T U_k \Lambda_k^{-1} U_k^T \Phi \phi_i \end{aligned}$$

We can rewrite this expression entirely in terms of known quantities, by noting that $\phi_i^T \phi_i = k(x_i, x_i)$, that $\Phi\Phi^T$ equals the kernel matrix with the ith row and column deleted (denoted K below) and that $\phi^T \Phi^T$ is a row of the same kernel matrix. We then obtain

$$e_i^2 = k(x_i, x_i) - 2\mathbf{k}_{-x_i}^T U_k \Lambda_k^{-1} U_k^T \mathbf{k}_{-x_i} + \mathbf{k}_{-x_i}^T U_k \Lambda_k^{-1} U_k^T K U_k \Lambda_k^{-1} U_k^T \mathbf{k}_{-x_i}$$

where we have denoted K the kernel matrix with the ith row and column deleted, and where a row of this kernel matrix is given by

$$\mathbf{k}_{-x_i} = [\ k(x_i, x_1) \quad k(x_i, x_2) \quad \ldots \quad k(x_i, x_{i-1}) \quad k(x_i, x_{i+1}) \quad \ldots \quad k(x_i, x_n)\ ]^T$$

We have thus derived the residual error of data point $\Phi(x_i)$ with respect to the top k principal components of the other data points in the feature space, expressed entirely in terms of the kernel function, the kernel matrix, and the eigenvectors of the kernel matrix.

We then wish to calculate the mean predicted residual sum of squares

$$MPRESS(k) = \frac{1}{n(d-k)} \sum_{i=1}^{n} e_i^2$$

However, the dimension d of the variables in feature space is often not known, or indeed infinite. This means that we can not directly apply the correction $1/(n(d-k))$ used for linear PCA. Instead, we note that the data points Φ lie entirely in a subspace of dimension at most n since $\Phi\Phi^T$ has at most n non-zero eigenvalues [Shawe-Taylor et al., 2005]. If we represent the data points instead in the coordinate system given by the eigenbasis, the dimension will be n and the residual error will be the same. Using this representation we obtain

$$MPRESS(k) = \frac{1}{n(n-k)} \sum_{i=1}^{n} e_i^2$$

We describe the procedure in Algorithm 5. The algorithm has time complexity $\mathcal{O}(n^4)$ if the matrix multiplications for $U_3 = U_k \Lambda_k^{-1} U_k^T$ and $U_3 K U_3^T$ (corresponding to $U3$ and UK in Algorithm 5) are calculated incrementally instead of from scratch at every iteration. Please see the next section for a detailed description of the faster implementation. See section 5.3 for experimental analysis.

Algorithm 5 Cross-validation for kernel PCA

Input: Dataset $\{x_i\}_{i=1}^{m+1}$
Output: The selected number of principal components

$K \leftarrow (k(x_i, x_j))_{i,j}/(n-1)$
$errors = \texttt{zeros}(n)$
for $i = 1, 2, ..., n$ **do**
  $K2 \leftarrow \texttt{deleterow}(i, K)$
  $kx \leftarrow K2[1 : (n-1), i]$
  $K2 \leftarrow \texttt{deletecolumn}(i, K2)$
  $L, U \leftarrow \texttt{eigendecomposition}(K2)$
  for $k = 1, 2, ..., n-1$ **do**
    $U2 \leftarrow U[1 : (n-1), 1 : k]$
    $L2 \leftarrow L[1 : k, 1 : k]$
    $U3 \leftarrow \texttt{dotproduct}(U2, \texttt{dotproduct}(\texttt{inverse}(L), \texttt{transpose}(U2)))$
    $UK \leftarrow \texttt{dotproduct}(U3, \texttt{dotproduct}(K, U3))$
    $a \leftarrow \texttt{dotproduct}(\texttt{transpose}(kx), \texttt{dotproduct}(U3, kx))$
    $b \leftarrow \texttt{dotproduct}(\texttt{transpose}(kx), \texttt{dotproduct}(UK, kx))$
    $err \leftarrow (K[i,i] - 2a + b)/(n(n-k))$
    $errors[k] = errors[k] + err$
  end for
end for
return $\texttt{argmin}(errors)$

Instead of re-calculating the eigendecomposition with a single data point excluded in the algorithm above, the decremental procedure presented in Section 3.2 can be applied to remove the effect of single data points on the principal components.

5.2.1 Fast implementation

A direct implementation of Algorithm 5 will have time complexity $\mathcal{O}(n^5)$, due to $(n \times n) \times (n \times n)$ matrix multiplications in the inner loop. It's possible to implement the algorithm with time complexity $\mathcal{O}(n^4)$ instead, which is the usual

complexity when implementing leave-one-out cross-validation for algorithms with cubic time complexity. We describe the fast implementation in Algorithm 6.

Algorithm 6 Fast cross-validation for kernel PCA

Input: Dataset $\{x_i\}_{i=1}^{m+1}$
Output: The selected number of principal components

$K \leftarrow (k(x_i, x_j))_{i,j}/(n-1)$
$errors = \texttt{zeros}(n)$
for $i = 1, 2, ..., n$ **do**
  $K2 \leftarrow \texttt{deleterow}(i, K)$
  $kx \leftarrow K2[1 : (n-1), i]$
  $K2 \leftarrow \texttt{deletecolumn}(i, K2)$
  $L, U \leftarrow \texttt{eigendecomposition}(K2)$
  $a = 0$
  $b = 0$
  for $k = 1, 2, ..., n-1$ **do**
    $U2 \leftarrow U[1 : (n-1), k]$
    $L2 \leftarrow L[k, k]$
    $kU \leftarrow \texttt{dotproduct}(\texttt{transpose}(kx), U2) * \texttt{transpose}(U2)/L2$
    $a = a + \texttt{dotproduct}(kU, kx)$
    $b = b + \texttt{dotproduct}(kU, \texttt{dotproduct}(K2, \texttt{transpose}(kx)))$
    $err \leftarrow (K[i, i] - 2a + b)/(n(n-k))$
    $errors[k] = errors[k] + err$
  end for
end for
return $\texttt{argmin}(errors)$

5.3 NUMERICAL EXPERIMENTS

In this section we detail experiments on our methodology for cross-validation of kernel PCA, where we apply our cross-validation methodology to determine the number of principal components. We implement the algorithm for the `magic`, `yeast`, `cardiotocography` and `segmentation` datasets from the UCI machine learning repository.

For these experiments we use the radial basis functions (RBF) kernel and the polynomial kernel (see Table 1.2 on page 33). For the RBF kernel we determine

the bandwidth using the heuristic described in Section 1.8.2. For the polynomial kernel we set the parameters to $d = 5$ and $R = 1$. We plot the average error for each number of retained principal components for datasets containing 10, 50 and 100 data points. Please see Figures 5.1 - 5.12 below.

We also table the chosen number of principal components, i.e. the number of principal components corresponding to the lowest values of the mean predicted sum of squares. Please see Table 5.1

In general, the more data points that are in the sample, the more principal components are needed to capture accurately the information in all data points. For some datasets, one of the two kernels requires fewer principal components to capture the whole dataset. For example, for the `segmentation` dataset, the polynomial kernel requires only 14 principal components, whilst the RBF kernel requires 41.

If the kernel matrix does not have full rank, then it is sufficient to calculate the residual error with respect to at most the eigenvectors corresponding to non-zero eigenvalues. The error when including additional eigenvectors will not be decreased. Since the error is calculated by dividing by the eigenvalues, eigenvalues very close to zero can lead to instabilities in the form of very large negative or positive values for the error, and should be excluded. This has not been an issue in the implementation of our algorithm, since the selected number of principal components to retain was always well below the point where small eigenvalues could lead to instabilities.

Source code in Python implementing the algorithm for cross-validation of kernel PCA is available at `https://github.com/fredhallgren/KPCA-CV`.

Dataset	Kernel	Data size	Selected PCs
magic	radial	10	2
magic	polynomial	10	3
magic	radial	50	21
magic	polynomial	50	26
magic	radial	100	45
magic	polynomial	100	46
yeast	radial	10	4
yeast	polynomial	10	4
yeast	radial	50	21
yeast	polynomial	50	21
yeast	radial	100	39
yeast	polynomial	100	22
cardiotocography	radial	10	3
cardiotocography	polynomial	10	6
cardiotocography	radial	50	20
cardiotocography	polynomial	50	24
cardiotocography	radial	100	40
cardiotocography	polynomial	100	56
segmentation	radial	10	2
segmentation	polynomial	10	3
segmentation	radial	50	18
segmentation	polynomial	50	8
segmentation	radial	100	41
segmentation	polynomial	100	14

Table 5.1: The selected number of principal components

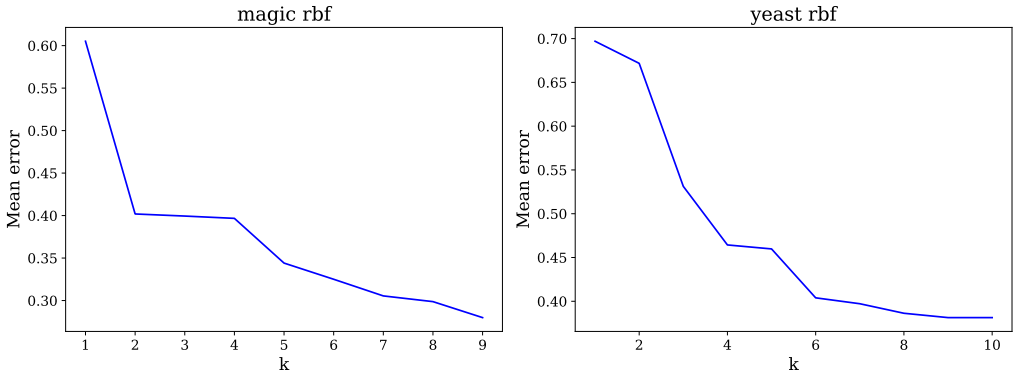

Figure 5.1: Average residual error for the top k principal components with data size 10

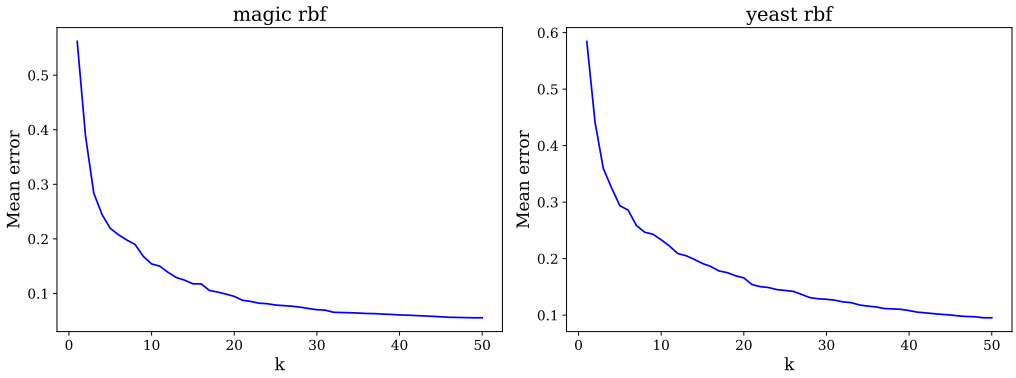

Figure 5.2: Average residual error for the top k principal components with data size 50

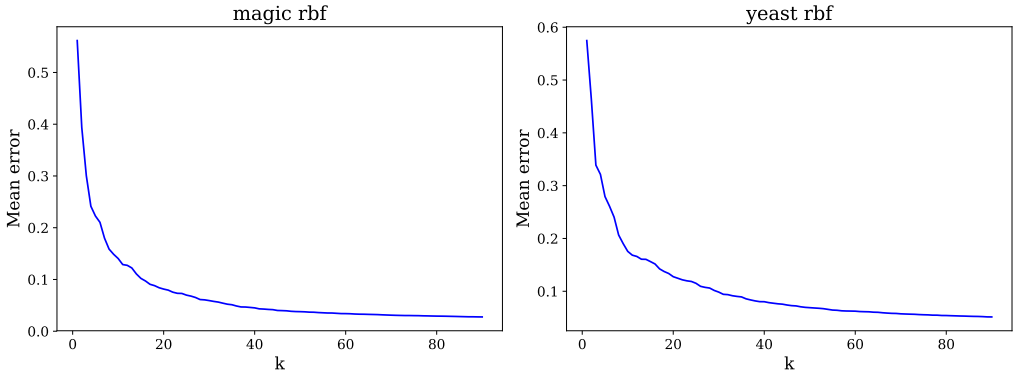

Figure 5.3: Average residual error for the top k principal components with data size 100

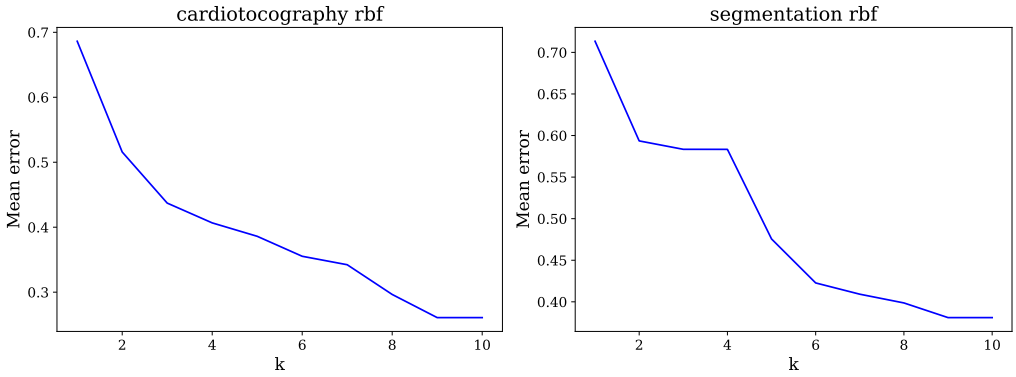

Figure 5.4: Average residual error for the top k principal components with data size 10

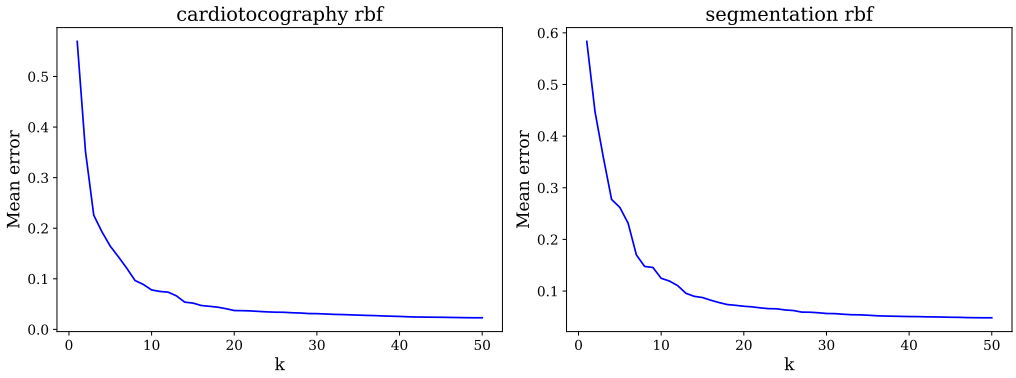

Figure 5.5: Average residual error for the top k principal components with data size 50

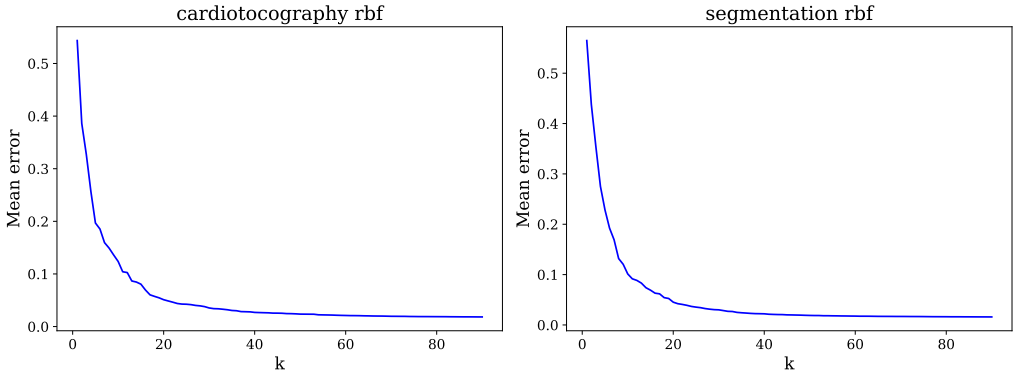

Figure 5.6: Average residual error for the top k principal components with data size 100

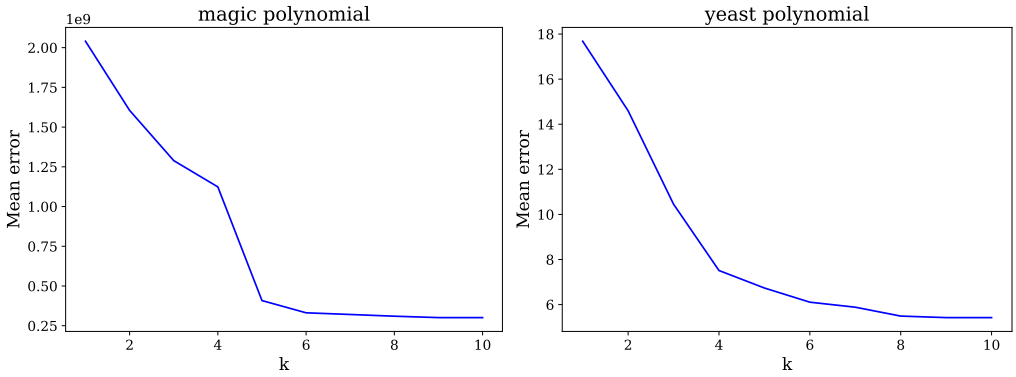

Figure 5.7: Average residual error for the top k principal components with data size 10

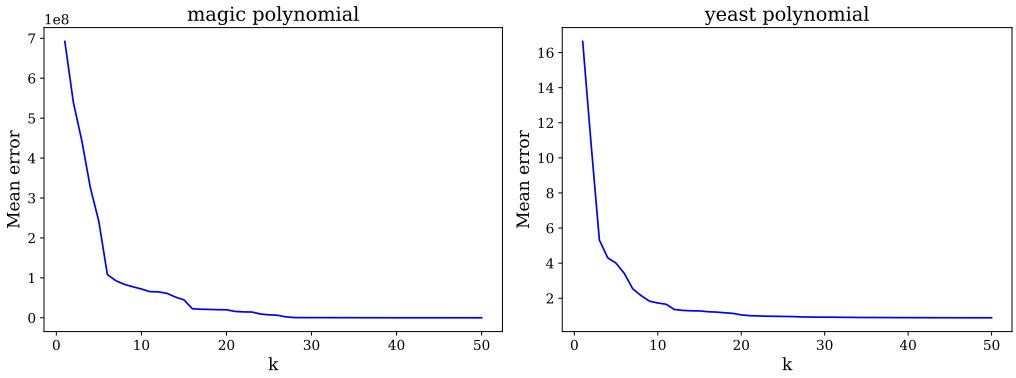

Figure 5.8: Average residual error for the top k principal components with data size 50

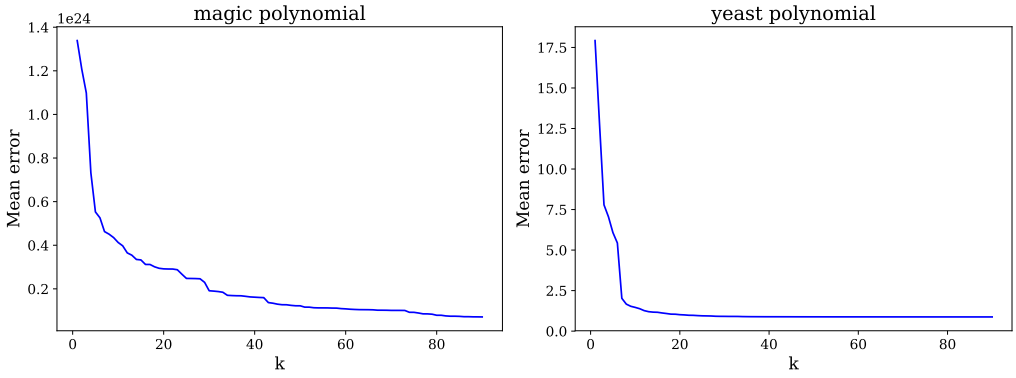

Figure 5.9: Average residual error for the top k principal components with data size 100

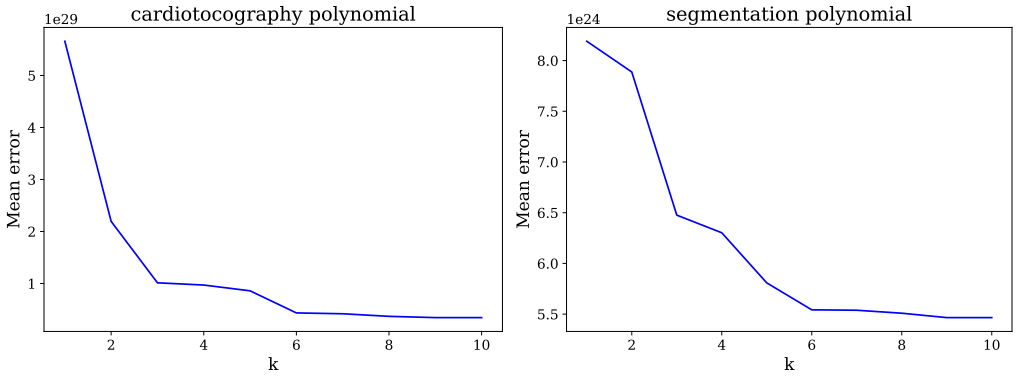

Figure 5.10: Average residual error for the top k principal components with data size 10

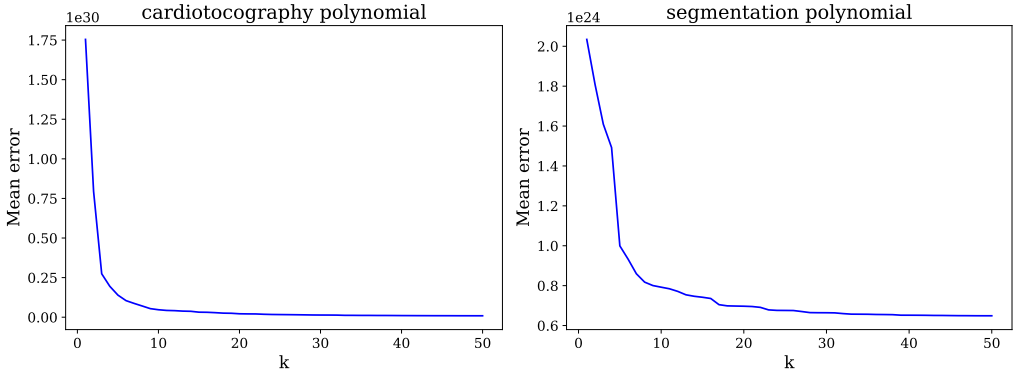

Figure 5.11: Average residual error for the top k principal components with data size 50

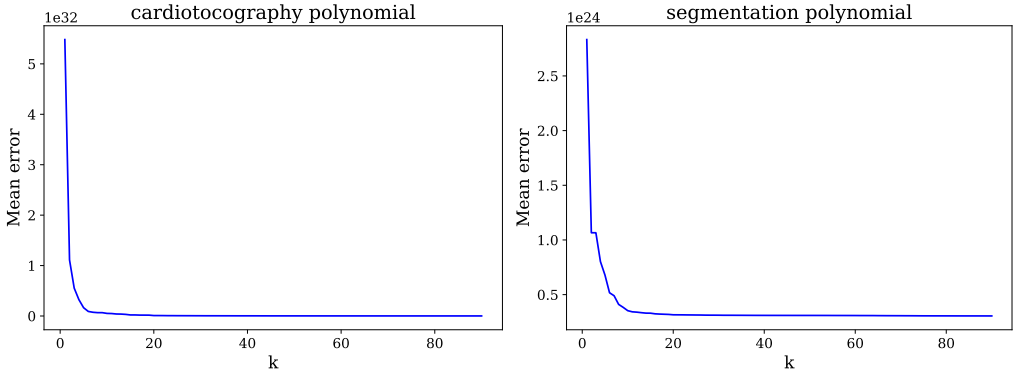

Figure 5.12: Average residual error for the top k principal components with data size 100

5.4 SUMMARY

In this chapter we have presented a leave-one-out cross-validation procedure specifically for kernel PCA, based on an equivalent method for linear PCA, and demonstrated its behaviour through numerical experiments. It appears to be the first algorithm specifically developed for this purpose.

Chapter 6

Kernel PCA with the Nyström method

The Nyström method can substantially reduce the computational requirements of kernel methods and enable them to be applied to problems where otherwise they would be intractable. Sometimes this can be achieved simply by replacing the original full kernel matrix by the approximate kernel matrix obtained from the Nyström method, but for certain methods it is not this straight-forward. Kernel PCA is one of those methods.

In this chapter we derive kernel PCA with the Nyström method[1]. We provide orthonormal principal components, uncorrelated principal scores, the variances of data along the principal components, known as the principal values, and the reconstruction error of the full data with respect to the principal components. The principal scores are perhaps of particular interest, since they allow for the method to be used as a preprocessing step before applying supervised learning methods, by virtue of providing a new representation of data points in the new coordinate system defined by the principal components.

[1]Parts of this chapter have previously been made available in Hallgren [2021]

The method centres the data in the feature space, as is the case for linear PCA [Jolliffe and Cadima, 2016]. Without this adjustment, the perpendicular lines defined by the principal components are forced to go through the origin, no longer minimizing the reconstruction error in an unconstrained fashion and requiring an assumption of zero-mean data in feature space. If the assumption does not hold then the principal components may be very different from their true values. For any positive kernel function, like the popular radial basis functions or Cauchy kernels, the assumption will never hold, since the corresponding functions $k(x_i, \cdot)$ in the reproducing kernel Hilbert space are positive everywhere. Subtracting the mean from the input data will not give zero-mean data in feature space.

Often the top eigenvalue of the uncentred kernel matrix accounts for the mean being different from zero, and the subsequent eigenvalues approximately correspond to the full set of eigenvalues of the centred kernel matrix. However, this is not always the case, and the correspondence is not exact. A comparison of the eigenvalues of the centred and uncentred kernel matrices with some real data would demonstrate this.

We further study the statistical accuracy of the proposed method. We show that in the special case when the number of subsampled data points for the Nyström method equals the PCA dimension, then both the empirical and true reconstruction errors of the Nyström method equal the corresponding reconstruction errors for kernel PCA constructed using only the subset of data points. For the general case we provide a finite-sample confidence bound with $\mathcal{O}(m^3)$ complexity that doesn't require that we have observed the entire dataset, only the subset of data. In line with essentially all results on the accuracy of kernel PCA we here assume that data has zero mean[2].

We illustrate and evaluate the proposed method and derived confidence bound

[2]Please see the conclusion for further explanation

through experimental analysis using several different datasets and kernel functions. We first compare the accuracy of the proposed method with a number of other unsupervised learning methods, and then illustrate the behaviour of the bound across different PCA dimensions. The source code for all the experiments is publicly available at `https://github.com/fredhallgren/nystrompca`.

From the proof of the confidence bound one can deduce sharper versions of some concentration results from Rosasco et al. [2010] on the empirical covariance operator and its eigenvalues. Please see Section 6.5.1 for details.

To demonstrate the use of Nyström kernel PCA with supervised learning methods we apply it to the regression problem to present kernel principal components regression with the Nyström method. Principal components regression (PCR) performs a linear regression on the principal scores instead of the original data and introduces regularization and improved generalization. We illustrate the method through experimental analysis and compare it to kernel ridge regression with the Nyström method.

In the next section we give an overview of previous work (Section 6.1), then go through relevant background (Section 6.2), present the main method (Section 6.3), study the special case when $d = m$ (Section 6.4), provide the confidence bound on the accuracy of the method (Section 6.5), conduct experimental analysis of the method and bound (Section 6.6), present kernel principal component regression with the Nyström method (Section 6.7) and finally conclude with a summary and outlook (Section 6.8). Proofs are in the appendix.

6.1 PREVIOUS WORK

The study of the statistical accuracy of kernel PCA, or of the related problems of functional PCA [Besse and Ramsay, 1986, Hall et al., 2006] and PCA of a Hilbert space-valued random variable [Besse, 1991], was initiated in Dauxois et al. [1982]. They demonstrated the consistency of the reconstruction error and asymptotic normality of the empirical reconstruction error and principal components about the true quantities. The asymptotics of kernel PCA was also studied in Koltchinskii and Giné [2000]. A concentration inequality for the empirical reconstruction error versus its expectation was provided in Shawe-Taylor et al. [2002] and the same authors later presented a confidence bound on the expected empirical reconstruction error versus the true error [Shawe-Taylor et al., 2005], using Rademacher complexities [Bartlett and Mendelson, 2002]. In this bound the expectation is with respect to the data point to be projected and the confidence with respect to different training datasets. A sharper version of the latter, as well as a version for centred kernel PCA, was presented in Blanchard et al. [2007]. The centred version is more conservative compared to the uncentred one. Approximate confidence bounds for both the principal values and components were given in Hall and Hosseini-Nasab [2006] based on the bootstrap method [Davison and Hinkley, 1997]. However, these results are not immediately applicable to kernel PCA since the kernel is defined on a compact subset of $\mathbb{R} \times \mathbb{R}$. The first PAC-Bayes bounds for kernel PCA were recently presented in Haddouche et al. [2020].

A recent paper [Sterge et al., 2020] presented a partly similar method for kernel PCA with the Nyström method, but only explicitly derived the top principal component. Other important differences are that we centre the data points in feature space, like linear PCA, present the principal scores, which allows for the method to be used for preprocessing before supervised learning techniques, and

perform an experimental analysis to evaluate our method. They also presented a probabilistic inequality for the true reconstruction error with respect to the empirical subspace, which depends on the maximum value of the kernel function $\sup_x k(x, x)$, the total number of data points n and the covariance operator C from the true distribution $\mathbb{P}$. As a corollary they also presented an asymptotic rate of convergence under the assumption of polynomial decay of the eigenvalues of C. The main difference between our confidence bound and their inequality is that ours bounds the empirical reconstruction error by a quantity that is calculated from data, whereas their inequality is for the expected reconstruction error and includes quantities that depend on the unknown true data distribution on the other side of the inequality.

Sterge and Sriperumbudur [2021] later presented a similar analysis to Sterge et al. [2020], which included a way of centring the data. However, their centring is different from the one considered here and the presented method does not minimize the reconstruction error or recover standard kernel PCA when $m = n$.

The Nyström method has been widely studied for different settings and assumptions. Originally developed for the discretization of integral equations [Nyström, 1930, Banach, 1932], it was adapted to kernel methods in Williams and Seeger [2001] and applied to regression. The accuracy of the approximate kernel matrix versus the full kernel matrix, considering the full dataset as fixed, has been studied in a number of papers, please see Gittens and Mahoney [2016] and references therein. The study of the accuracy of the Nyström method as applied to regression culminated in the seminal work by Rudi et al. [2015] as a probabilistic bound on the expected regression error with general assumptions.

6.2 PROBLEM SETUP

We have a reproducing kernel Hilbert space $\mathcal{H}$ (RKHS) of functions from a set $\mathcal{X}$ to the real numbers. We assume $\mathcal{H}$ is separable, which will be the case for example if k is continuous and $\mathcal{X}$ is compact [Paulsen and Raghupathi, 2016]. We have observations $\{x_i\}_{i=1}^n$ of an $\mathcal{X}$-valued random variable $X : (\Omega, \mathcal{A}, \mathbb{P}) \to (\mathcal{X}, \mathcal{A}_\mathcal{X}, \mathbb{P}_X)$ where $\mathbb{P}_X(A) = \mathbb{P}(X^{-1}(A))$ [Cohn, 1980, Graham and Talay, 2011]. We assume X is absolutely continuous and that it has a continuous density and so all x_i will be distinct. We obtain a random variable $Z = \phi(X) \in \mathcal{H}$ with observations $z_i = \phi(x_i)$, assuming that ϕ is measurable. Its expectation in $\mathcal{H}$ is given by $\mathbb{E}[Z] = \int Z d\mathbb{P}$. We assume Z is integrable and so is an element in $L^1(\Omega, \mathcal{A}, \mathbb{P}; \mathcal{H})$ with norm $\mathbb{E}[\|Z\|_\mathcal{H}] = \int \|Z\|_\mathcal{H} d\mathbb{P}$ [Ledoux and Talagrand, 2013].

Observation of an integrable random variable Y with values in some Banach space $\mathcal{B}$ (such as $\mathcal{H}$, or $\mathbb{R}$ with norm $\|\cdot\|_\mathcal{B} = |\cdot|$) and with observations $y_1, y_2, ..., y_M$ corresponds to application of the evaluation operator

$$E_\omega : L^1(\Omega, \mathcal{A}, \mathbb{P}_\mathcal{B}; \mathcal{B}) \to \mathcal{B}$$

which is given by

$$E_\omega(Y) = Y(\omega)$$

$$E_{\omega_i}(Y) = Y(\omega_i) = y_i$$

and which is linear, since $E_\omega(aY_1 + bY_2) = (aY_1 + bY_2)(\omega) = aY_1(\omega) + bY_2(\omega)$ for $a, b \in \mathbb{R}$ with norm

$$\|E_\omega\| = \sup_{\|Y\|_1=1} \|E_\omega(Y)\|_\mathcal{B} = \sup_{\|Y\|_1=1} \|Y(\omega)\|_\mathcal{B} = \sup_{y \in \mathcal{B}} \|y\|_\mathcal{B}$$

Principal component analysis (PCA) of a zero-mean random variable Z with values in a Hilbert space $\mathcal{H}$ constructs an optimal subspace $V_d \subset \mathcal{H}$, of dimension d, such that the so-called reconstruction error

$$R(V) = \mathbb{E}\left[\|P_V Z - Z\|_{\mathcal{H}}^2\right]$$

is minimized, where $P_V : \mathcal{H} \to \mathcal{H}$ is the projection of (a realization of) Z on a subspace V [Besse, 1991]. This is termed the *true* reconstruction error [Blanchard et al., 2007]. Since Z is integrable, the reconstruction error always exists and is finite.

We denote the optimal d-dimensional subspace by V_d

$$V_d = \underset{\dim(V)=d}{\arg\min}\, \mathbb{E}\left[\|P_V Z - Z\|_{\mathcal{H}}^2\right]$$

An estimate of the optimal subspace V_d is obtained from the data $\{z_i\}_{i=1}^n$ by minimizing the *empirical* reconstruction error

$$R_n(V) = \frac{1}{n}\sum_{i=1}^{n} \|P_V z_i - z_i\|_{\mathcal{H}}^2$$

which has a unique minimum since all eigenvalues are distinct [Blanchard et al., 2007]. We denote the estimated subspace by $\hat{V}_d$.

One may also consider the true reconstruction error with respect to the empirical subspace, given by $R(\hat{V}_d) = \mathbb{E}\left[\|P_{\hat{V}_d} Z - Z\|_{\mathcal{H}}^2\right]$, where the expectation may be taken both with respect to Z and $\hat{V}_d$, or treating the subspace as fixed; as well as the expected value of the empirical reconstruction error, given by $\mathbb{E}\left[R_n(\hat{V}_d)\right] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n \|P_{\hat{V}_d} z_i - z_i\|_{\mathcal{H}}^2\right]$.

When the random variable Z is not assumed to have zero mean, the smallest

reconstruction error is obtained from the centred random variable $Z' = Z - \mathbb{E}[Z]$

$$R(V_d) = \min_{\dim(V)=d} \mathbb{E}[\|P_V Z' - Z'\|_{\mathcal{H}}^2]$$

and similarly for the empirical reconstruction error replacing z_i by $z_i' = z_i - \frac{1}{n}\sum_{\ell=1}^n z_\ell$.

Alternatively, instead of minimizing the reconstruction error over d-dimensional subspaces V using the centred random variable, one may minimize over affine subspaces with respect to the original random variable, and also optimize with respect to the term used for centring

$$R(V_d) = \min_{\substack{a\in\mathcal{H}\\ \dim(V)=d}} \mathbb{E}[\|P_{a+V} Z - Z\|_{\mathcal{H}}^2] = \min_{\substack{a\in\mathcal{H}\\ \dim(V)=d}} \mathbb{E}[\|P_V(Z-a) - (Z-a)\|_{\mathcal{H}}^2]$$

where a is the translation of the vector space V, and whose optimal value is known to equal $\mathbb{E}[Z]$, and $P_{a+V}Z = a + P_V(Z-a)$ is the affine projection.

The covariance operator is an element $C(u,v) \in \mathcal{H}\otimes\mathcal{H}$ in the tensor product of bilinear functionals on $\mathcal{H}$, given by $C(u,v) = \mathbb{E}[Z\otimes Z]$. The centred covariance operator is given by

$$C'(u,v) = \mathbb{E}[(Z - \mathbb{E}[Z]) \otimes (Z - \mathbb{E}[Z])] =: \mathbb{E}[Z'\otimes Z']$$

Identifying $\mathcal{H}\otimes\mathcal{H}$ with the space $\mathrm{HS}(\mathcal{H})$ of Hilbert-Schmidt operators on $\mathcal{H}$ by way of the mapping of elementary tensors $u\otimes v \mapsto \langle\cdot, u\rangle_{\mathcal{H}} v$ we obtain $C' = \mathbb{E}[\langle\,\cdot, Z'\rangle_{\mathcal{H}} Z']$. When we refer to the covariance operator we may either refer to the tensor in $\mathcal{H}\otimes\mathcal{H}$ or the operator in $\mathrm{HS}(\mathcal{H})$.

A Hilbert-Schmidt operator L is an operator on a Hilbert space $\mathcal{H}$ with finite Hilbert-Schmidt norm, given by $\|L\|_{\mathrm{HS}(\mathcal{H})} = \sum_i \|Le_i\|_{\mathcal{H}}$ for any orthonormal basis $\{e_i\}_i$ in $\mathcal{H}$ [Davies, 2007, Chapter 5]. It is a Hilbert space, with inner

product $\langle L_1, L_2 \rangle_{\mathrm{HS}(\mathcal{H})} = \sum_i \langle L_1 e_i, L_2 e_i \rangle_{\mathcal{H}}$. The Hilbert-Schmidt norm is always larger than or equal to the operator norm, $\|L\| \leq \|L\|_{\mathrm{HS}(\mathcal{H})}$, and if $\mathcal{H}$ is finite it coincides with the Frobenius norm, $\|L\|_{\mathrm{HS}(\mathcal{H})} = \|M\|_F$ where M is a matrix representation of L [Kreyszig, 1989].

The covariance operator C' is compact, since it is Hilbert-Schmidt, and so its spectrum is countable and all spectral values are eigenvalues apart from possibly 0, and it is in the closure of the finite rank operators with respect to the topology induced by the operator norm. Since C' is infinite-dimensional, by assumption, the value 0 is always a spectral value. Furthermore, the covariance operator is self-adjoint, and so the spectrum is real and the resolvent spectrum is empty; this means that 0 is either an eigenvalue or belongs to the continuous spectrum. Finally, it is positive and so the spectrum is positive.

The sum of the smallest eigenvalues of the centred operator C' equal the minimum true reconstruction error. The eigenvectors form a countable orthonormal basis of the image of C' in $\mathcal{H}$, which can be extended to an orthonormal basis for the entire space, since $\mathcal{H}$ is separable. Denoting the eigenvalues by $\{\lambda_i\}_{i=1}^{\infty}$ in decreasing order the minimum reconstruction error can be written $R(V_d) = \sum_{i=d+1}^{\infty} \lambda_i$. The optimal subspace itself is given by the shifted linear span of the eigenvectors of C' corresponding to the top d eigenvalues, $\mathbb{E}[Z] + \mathrm{span}\{\{\phi_i\}_{i=1}^{d}\}$.

Replacing the measure $\mathbb{P}_Z$ on $\mathcal{H}$ by the empirical measure $\mathbb{P}_n = \frac{1}{n}\sum_{i=1}^{n} \delta_{z_i}$, where δ_x is the Dirac delta function, we obtain the empirical covariance operator $C'_n : \mathcal{H} \to \mathcal{H}$

$$C'_n = \frac{1}{n} \sum_{i=1}^{n} \langle \cdot, z_i'^n \rangle_{\mathcal{H}} z_i'^n$$

where $z_i'^n = z_i - \frac{1}{n}\sum_{k=1}^{n} z_k$. We denote its eigenvalues $\hat{\lambda}_1^n, \hat{\lambda}_2^n, ..., \hat{\lambda}_n^n$ in decreasing order and corresponding eigenvectors $\hat{\phi}_1^n, \hat{\phi}_2^n, ..., \hat{\phi}_n^n$. It has finite rank, and so the spectrum only contains eigenvalues, and may or may not include 0.

The minimum empirical reconstruction error is given by its smallest eigenvalues, $R_n(\hat{V}_d) = \sum_{i=d+1}^{n} \hat{\lambda}_i^n$, and it can be decomposed as $C_n' = \sum_{i=1}^{n} \hat{\lambda}_i^n \langle \cdot, \hat{\phi}_i^n \rangle_{\mathcal{H}} \hat{\phi}_i^n$.

If $s : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is square-integrable in the second variable, then the operator given by

$$T_s f = \int_{\mathcal{X}} s(x, y) f(y) d\mathbb{P}_X(y)$$

is an isometry of $L^2(\mathcal{X}, \mathcal{A}_{\mathcal{X}}, \mathbb{P}_X; \mathbb{R})$ into the RKHS with kernel $k(x, y) = \int_{\mathcal{X}} s(x, z) s(z, y) d\mathbb{P}_X(z)$ [Paulsen and Raghupathi, 2016]. One may also consider the integral operator

$$T_k f = \int_{\mathcal{X}} k(x, y) f(y) d\mathbb{P}_X(y)$$

which is equal to $T_k = T_s^2$ and whose eigenvalues equal those of the covariance operator C [Bach, 2017, Shawe-Taylor et al., 2005].

If one replaces the probability measure $\mathbb{P}_X$ by its empirical equivalent $\mathbb{P}_n = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}$with respect to the data points $\{x_i\}_{i=1}^n$ one obtains an empirical operator T_n

$$T_n f = \int_{\mathcal{X}} k(x, y) f(y) d\mathbb{P}_n(y) = \frac{1}{n} \sum_{i=1}^{n} k(x, x_i) f(x_i)$$

The sampling operator G_n, $f \mapsto \frac{1}{\sqrt{n}}(f(x_1), f(x_2), ..., f(x_n))$ defines an isometry of $L^2(\mathcal{X}, \mathcal{A}_{\mathcal{X}}, \mathbb{P}_X; \mathbb{R})$ into $\mathbb{R}^n$ which identifies T_n with K [Koltchinskii and Giné, 2000]. Its adjoint G^* is given by $\alpha \mapsto \sum_{k=1}^{n} \alpha_k k(x_k, x)$ [Rudi et al., 2015]. Furthermore, $C_n = G^* G$ and $K = G G^*$.

And so the eigenvalues of the empirical kernel integral operator T_n are the same as the eigenvalues of the kernel matrix, and its eigenvectors are given by [Bengio et al., 2004]

$$\hat{\psi}_j^n = \frac{\sqrt{n}}{\hat{\lambda}_j^n} \sum_{i=1}^{n} u_{j,i} k(x_i, x) = \frac{\sqrt{n}}{\hat{\lambda}_j^n} u_j^T \kappa(x)$$

The values of $\hat{\psi}_j^n(x)$ at the points $x_1, x_2, ..., x_n$ equal the corresponding eigenvector of the kernel matrix K, $\hat{\psi}_j^n(x_i) = (u_j)_i$.

If we randomly sample $m < n$ data points $\{x_j\}_{j \in S}$ from the full dataset and then take the values of $\hat{\psi}_j^m,\ j = 1, 2, ..., m$ at all the points $x_1, x_2, ..., x_n$, and normalize by $\frac{1}{\sqrt{n}}$, we obtain the Nyström approximation [Williams and Seeger, 2001]

$$\begin{aligned} \widetilde{\lambda}_i &= \frac{n}{m} \hat{\lambda}_i^m \\ \widetilde{u}_i &= \sqrt{\frac{m}{n}} \frac{1}{\hat{\lambda}_i^m} K_{nm} u_i \end{aligned} \tag{6.1}$$

Multiplying together the approximate eigenvectors and eigenvalues one so obtains an approximate kernel matrix $\widetilde{K} = K_{nm} K_{mm}^{-1} K_{mn}$ where K_{mm} contains the m subsampled rows and columns of K, K_{nm} contains the m subsampled columns, and K_{mn} is its transpose. The approximate kernel matrix can serve as a replacement of the original kernel matrix for improved computational efficiency for different kernel methods.

Kernel methods in machine learning look for functions in the reproducing kernel Hilbert space to be adapted to data

$$f(x) = \sum_{j=1}^{n} \alpha_j \langle \phi(x_j), \phi(x) \rangle_{\mathcal{H}} = \sum_{j=1}^{n} \alpha_j k(x_j, x)$$

where $\{\alpha_j\}$ are parameters. The Nyström method may equivalently be defined by restricting these functions to lie in the linear span of the m subsampled data points $\{\phi(x_i)\}_{i \in S}$, while using the full dataset of n points for estimation of the unknown parameters [Rudi et al., 2015]. For fixed S the linear span of $\{\phi(x_i)\}_{i \in S}$ is a closed subspace of $\mathcal{H}$ and so is a Hilbert space, which we will denote by $\mathcal{H}_S$

[Bollobás, 1999]. In other words, one looks for functions of the form

$$f(x) = \sum_{j \in S} \alpha_j \langle \phi(x_j), \phi(x) \rangle_{\mathcal{H}} = \sum_{j \in S} \alpha_j k(x_j, x)$$

that solve an empirical risk minimization problem based on all data points $\{x_i\}_{i=1}^n$.

After drawing the n observations $\{x_i\}_{i=1}^n$ independently from $\mathbb{P}_X$, the subset of m data points $\{x_i\}_{i \in S} = \{x_{i_1}, x_{i_2}, ..., x_{i_m}\}$ is randomly selected according to a specified distribution $p(S|\{x_i\}_{i=1}^n)$. Before the data points are observed the elements in the subset are random variables $\{X_{i_1}, X_{i_2}, ..., X_{i_m}\}$. For notational convenience we will assume that the data points are reordered after the subampling so that $\{x_i\}_{i \in S} = \{x_1, x_2, ..., x_m\}$.

Kernel PCA may be obtained by appealing to the $\ell^2(\mathbb{R})$ representation of a separable real Hilbert space and arranging the data points in $\mathcal{H}$ in a data matrix Φ with one data point occupying a row, which may then have an infinite number of columns. The principal components are then the eigenvectors of $\frac{1}{n}\Phi^T\Phi$ and the kernel matrix can be written as $K = \Phi\Phi^T$. The mean can be subtracted in the RKHS (the feature space) through Schölkopf et al. [1998]

$$K' = (\Phi - \mathbb{1}_n\Phi)(\Phi - \mathbb{1}_n\Phi)^T = K - \mathbb{1}_n K - K\mathbb{1}_n + \mathbb{1}_n K \mathbb{1}_n$$

where $\mathbb{1}_n$ is a matrix for which $(\mathbb{1}_n)_{i,j} = \frac{1}{n}$. The eigenvalues of $K' = Q\Lambda Q^T$ scaled by $\frac{1}{n}$ then measure the variance of the data projected onto each individual principal component. Its eigenvectors Q are proportional to the principal scores – the principal scores are given by $S = Q\Lambda^{1/2}$. By the singular value decomposition $\Phi - \mathbb{1}_n\Phi = Q\Sigma E^T$, where $\Lambda = \Sigma^2$, the principal scores of a *new* data point

x^* which is centred in feature space is given by

$$\begin{aligned} w^* &= ((\phi(x^*) - \mathbf{1}_n\Phi)E)^T = ((\phi(x^*) - \mathbf{1}_n\Phi)(\Phi - \mathbb{1}_n\Phi)^T Q\Lambda^{-1/2})^T \\ &= ((\kappa(x^*)^T - \kappa(x^*)^T\mathbb{1}_n - \mathbf{1}_n K + \mathbf{1}_n K\mathbb{1}_n)Q\Lambda^{-1/2})^T \\ &= \Lambda^{-1/2}Q^T(\kappa(x^*) - \mathbb{1}_n\kappa(x^*) - K\mathbf{1}_n + \mathbb{1}_n K\mathbf{1}_n) \\ &=: \Lambda^{-1/2}Q^T\kappa'(x^*) = S_d^{-1}\kappa'(x^*) \end{aligned} \tag{6.2}$$

where $\phi(x_i)$ is an element in $\ell^2(\mathbb{R})$ as a row vector, $\mathbf{1}_n$ is a length-n column vector with each element equal to $\frac{1}{n}$ and $\kappa(x) = (k(x_1, x), k(x_2, x), ..., k(x_n, x))^T$.

Using the above formula in Equation (6.2) to calculate the scores for the original data points we get that $\kappa'(x^*)$ becomes K' and obtain $w^{*T} = K'Q\Lambda^{-1/2} = Q\Lambda Q^T Q\Lambda^{-1/2} = Q\Lambda^{1/2}$ and so as expected we recover the previous expression for the principal scores.

When applying PCA it is often appropriate to normalize the input variables to have variance 1, so as to make the analysis independent of arbitrary changes of units in the data. Otherwise the variables with higher variance will dominate the principal components and comparisons between variables become difficult. This normalization will often also be appropriate for kernel PCA and we do this for the experimental analysis (Section 6.6). The centring of variables in the feature space does not guarantee that the input variables become centred.

The approximate eigenvalues and eigenvectors from the Nyström method in Equation (6.1) applied to the centred kernel matrix may be used to define an approximate kernel PCA. However, these approximate principal scores are not orthogonal (i.e. uncorrelated), so they do not define true PCA, and the eigenvalues do not describe the variance captured by these vectors. There is a need for another way to derive kernel PCA with the Nyström method.

6.3 KERNEL PCA WITH THE NYSTRÖM METHOD

In this section we present kernel PCA with the Nyström method, which provides an efficient and flexible technique for non-linear PCA. We present the corresponding quantities that are defined for linear PCA and are useful for data exploration and application of the method in downstream tasks

1. a set of orthogonal principal components with unit length in the linear span of the subsampled data points in $\mathcal{H}$ (denoted $\mathcal{H}_S$),

2. the variance of the data along each of these directions, termed the explained variance,

3. the reconstruction error of the data onto the principal components,

4. a set of uncorrelated principal scores with the weightings of the data points on the principal components, and,

5. the principal scores of a new data point with respect to the existing principal components

For standard kernel PCA (2) and (3) are the same, but with the Nyström method they are different, since the principal components will not span the entire data.

We first present the principal components, explained variance and scores for a dataset in the following theorem

Theorem 2 (Nyström kernel PCA)**.** *Let $(\widetilde{\lambda}_j, v_j)$ be the eigenpairs and $V\widetilde{\Lambda}V^T$ be the eigendecomposition of*

$$\frac{1}{n}\widetilde{K}' = \frac{1}{n}K'^{-1/2}_{mm}K'_{mn}K'_{nm}K'^{-1/2}_{mm} \tag{6.3}$$

where

$$K'_{mn} = K_{mn} - K_{mn}\mathbb{1}_n - \mathbb{1}_n^{m,n}\widetilde{K} + \mathbb{1}_n^{m,n}\widetilde{K}\mathbb{1}_n$$

$$K'_{mm} = K_{mm} - \mathbb{1}_n^{m,n}K_{nm} - K_{mn}\mathbb{1}_n^{n,m} + \mathbb{1}_n^{m,n}\widetilde{K}\mathbb{1}_n^{m,n}$$

with $\widetilde{K} = K_{nm}K_{mm}^{-1}K_{mn}$ *and where* $\mathbb{1}_n$, $\mathbb{1}_n^{n,m}$ *and* $\mathbb{1}_n^{m,n}$ *are* $n \times n$, $n \times m$ *and* $m \times n$ *matrices respectively with each element equal to* $\frac{1}{n}$.

The perpendicular intersecting lines $\phi_0 + \langle\widetilde{\phi}_j\rangle$, $j = 1, 2, ..., m$ *in* $\mathcal{H}_S$ *along which the variance of the data is successively maximized, where the orthonormal vectors* $\{\widetilde{\phi}_j\}_{j=1}^m$ *are termed the principal components, are given by*

$$\phi_0 = \frac{1}{n}K_{nm}K_{mm}^{-1}\kappa_m(x)$$

$$\widetilde{\phi}_j = \sum_{k=1}^{m} u_{j,k}\left(k(x_k, x) - \phi_0\right)$$

and the variances along these directions are $\{\widetilde{\lambda}_j\}_{j=1}^m$, *termed the principal values or explained variance, where* $\kappa_m(x) = (k(x_1, x), k(x_2, x), ..., k(x_m, x))^T$, $u_j = K_{mm}^{\prime\,-1/2}v_j$ *and* $U = K_{mm}^{\prime\,-1/2}V$.

The projection coefficients of the centred data points onto the principal components, termed the principal scores, are given by

$$W = K'_{nm}U = K'_{nm}K_{mm}^{\prime\,-1/2}V$$

where each row of W *contains the scores of one data point onto the principal components.*

The principal scores of a new data point x^* is given by

$$w^* = U^T(\kappa_m(x^*) - K_{mn}\mathbf{1}_n - \mathbb{1}_n^{m,n}K_{nm}K_{mm}^{-1}\kappa_m(x^*) + \mathbb{1}_n^{m,n}\widetilde{K}\mathbf{1}_n) = U^T\,\widetilde{\kappa}(x^*)$$

where $\mathbf{1}_n$ is a length-n column vector given by $\mathbf{1}_n = (\frac{1}{n}, \frac{1}{n},\ ...,\ \frac{1}{n})^T$.

The principal components can be seen as defining new variables through linear combinations of the existing variables that have successively maximized variance and that are uncorrelated. The values of these new variables are given by the principal scores, which represent the data in a new coordinate system defined by the principal components as a new basis for the space. As such, the principal scores can be used as a drop-in replacement for the original data in arbitrary supervised or unsupervised learning methods, including after removing the scores corresponding to principal components with smaller eigenvalues. Please see Section 6.7 for an example of this.

To see that these new variables are uncorrelated also with the Nyström method we note that

$$W^TW = V^TK_{mm}'^{-1/2}K'_{mn}K'_{nm}K_{mm}'^{-1/2}V = nV^TV\widetilde{\Lambda}V^TV = n\widetilde{\Lambda}$$

which is a diagonal matrix.

The computational complexity of the method is $\mathcal{O}(nm^2)$ in time, which is the same as the Nyström method applied to regression. Centring of the matrix K_{nm} can be accomplished in $\mathcal{O}(m^3 + nm)$ operations, and so the centring in the proposed method adds no additional time requirements to the dominant $\mathcal{O}(nm^2)$ factor. We refer to the software implementation for full details.

The Nyström method approximates the corresponding full method, so when $m = n$ we should recover standard kernel PCA. In this case $\widetilde{K} = KK^{-1}K = K$

and as expected $K'_{mm} = K'_{nm} = K'$ and

$$K'^{-1/2}_{mm} K'_{mn} K'_{nm} K'^{-1/2}_{mm} = K'$$

and the scores are equal to $W = K'^{1/2} V = \sqrt{n}\, V \widetilde{\Lambda}^{1/2} V^T V = \sqrt{n}\, V \widetilde{\Lambda}^{1/2} = Q \Lambda^{1/2}$, which we know to be the scores for standard kernel PCA.

The scores of new data points are important when measuring the accuracy of PCA with a test set of hold-out data points, for example using the reconstruction error (Section 6.6), or when applying PCA as a preprocessing step for supervised learning methods and one wishes to create predictions for new data points, such as in principal component regression (Section 6.7).

If the data points are assumed to have zero mean in feature space then the matrices K'_{nm} and K'_{mm} may be replaced by K_{nm} and K_{mm} and the vector $\widetilde{\kappa}(x)$ by $\kappa_m(x)$. The principal components are then given by $\widetilde{\phi}_j = \sum_{k=1}^m u_{j,k} k(x_k, x)$.

The smallest $m - d$ Nyström eigenvalues $\sum_{j=d+1}^m \widetilde{\lambda}_j$ measure the residual variance of the data points *within* $\mathcal{H}_S$ and correspond to the reconstruction error $\frac{1}{n} \sum_{i=1}^n \| P_{\widetilde{V}_d} z'_i - P_{\mathcal{H}_S} z'_i \|^2_{\mathcal{H}}$, where $\widetilde{V}_d = \mathrm{span}\{\{\widetilde{\phi}_k\}_{k=1}^d\}$. The full reconstruction error with respect to the top d Nyström principal components is given by

$$R_n(\widetilde{V}_d) = \frac{1}{n} \sum_{i=1}^n \| z'_i - P_{\widetilde{V}_d} z'_i \|^2_{\mathcal{H}} = \frac{1}{n} \mathrm{Tr}(K') - \sum_{j=1}^d \widetilde{\lambda}_j \tag{6.4}$$

where $\mathrm{Tr}(\cdot)$ is the trace and $\frac{1}{n}\mathrm{Tr}(K')$ is the variance of the full dataset in $\mathcal{H}$. From Theorem 2 we know that this is the smallest reconstruction error among all d-dimensional subspaces in $\mathcal{H}_S$.

Calculation of this quantity is $\mathcal{O}(n^2)$ due to the centring of K. However, it can be approximated for example by subtracting the mean of K_{nm} instead of the

mean of K, which becomes $\mathcal{O}(nm)$. This is included as an option in the software package accompanying the paper[3]. Please see Section 6.6 for further details.

Note that the reconstruction error above in Equation (6.4) is slightly different from the reconstruction error of the uncentred data points with respect to the affine subspace $\phi_0 + \widetilde{V}_d$, which becomes $\frac{1}{n}\sum_{i=1}^{n} \|(z_i - \phi_0) - P_{\widetilde{V}_d}(z_i - \phi_0)\|_{\mathcal{H}}^2 = \frac{1}{n}\sum_{i=1}^{n} \|(z_i - \phi_0) - P_{\widetilde{V}_d} z_i'\|_{\mathcal{H}}^2$. Both reconstruction errors are at a minimum for the proposed method.

Another quantity of interest is the reconstruction error of the full dataset on the eigenspace of the subset of m data points. Creating PCA from a random subset of m data points to describe the full dataset will be termed *Subset PCA*. We may use the same centring as for the Nyström method and maintain the $\mathcal{O}(m^3)$ time complexity – that is to say we use the mean of the n data points projected onto $\mathcal{H}_S$. This also ensures that the amount of variance captured is the same whether we project the centred data onto the principal components, or the uncentred data onto the lines translated from the origin. The principal components will then be given by, for $j = 1, 2, ..., m$

$$\hat{\phi}_j^{m,n} = \sum_{k=1}^{m} u_{j,k}^m (k(x_k, x) - \phi_0)$$

where u_j^m is the jth eigenvector of $\frac{1}{m}K'_{mm}$. The variance of the full data captured by these principal components and the associated reconstruction error are presented in the following theorem

[3] `https://github.com/fredhallgren/nystrompca`

Theorem 3 (Subset PCA)**.** *The variance of the dataset* $\{\phi(x_i)\}_{i=1}^n$ *along the* j*th principal component* $\hat{\phi}_j^{m,n}$ *is given by*

$$\hat{\lambda}_j^{m,n} = \frac{1}{n}\sum_{i=1}^{n} \|P_{\hat{\phi}_j^{m,n}} z_i'\|_{\mathcal{H}}^2 = \frac{1}{n \cdot m\hat{\lambda}_j^m} u_j^{m\,T} K'_{mn} K'_{nm} u_j^m$$

where $(\hat{\lambda}_j^m, u_j^m)$ *is the* j*th eigenpair of* $\frac{1}{m}K'_{mm}$*.*

The reconstruction error of the full dataset onto the corresponding d*-dimensional PCA subspace is*

$$R_n(\hat{V}_d^m) = \frac{1}{n}\sum_{i=1}^{n} \|z_i' - P_{\hat{V}_d^m} z_i'\|_{\mathcal{H}}^2 = \frac{1}{n}\mathrm{Tr}(K') - \frac{1}{n \cdot m}\mathrm{Tr}(K'_{nm} U_d^m \Lambda_d^{m\,-1} U_d^{m\,T} K'_{mn})$$

where $U_d^m \Lambda_d^m U_d^{m\,T}$ *is the truncated eigendecomposition of* $\frac{1}{m}K'_{mm}$*.*

As expected, if $n = m = d$ then the reconstruction error is zero.

The method proposed in this section for efficient kernel PCA can also be applied to improve the scalability of MDS when these two methods are equivalent, as outlined in Section 6.2.

6.4 PRELUDE: A SPECIAL CASE

Before studying the statistical accuracy of kernel PCA with Nyström method with a confidence bound we present a majorization relation between Nyström and Subset PCA and consider the special case when the number of principal components retained equals the number of subsampled data points, $d = m$. In this case the reconstruction error for the Nyström method is the same as subset PCA, both for the empirical and true reconstruction errors.

Proposition 3.1. *We have the following majorization relation for the empirical error*

$$(\widetilde{\lambda}_1, \widetilde{\lambda}_2, ..., \widetilde{\lambda}_m) \succ (\hat{\lambda}_1^{m,n}, \hat{\lambda}_2^{m,n}, ..., \hat{\lambda}_m^{m,n})$$

The majorization is strict in the sense that $\sum_{j=1}^{d} \widetilde{\lambda}_j > \sum_{j=1}^{d} \hat{\lambda}_j^{m,n}$ for $d < m$, by the assumption of a continuous data distribution. Otherwise it is strict if $\mathrm{span}\{\{z_i\}_{i=1}^n\} \not\subseteq \mathcal{H}_S$.

A direct consequence of the proposition is that

$$R_n(\widetilde{V}_m) = R_n(\hat{V}_m^m)$$

For the true reconstruction error we consider the case where the sampling of the Nyström subset occurs independently of the values of the data points

Proposition 3.2. *Let $d = m$ and let the Nyström subset be sampled according to $p(S \mid x_1, x_2, ..., x_n)$. Then if*

$$p(S \mid x_1, x_2, ..., x_n) = p(S)$$

i.e. the subsampling is independent of the data, we have

$$R(\widetilde{V}_m) = R(\hat{V}_m^m)$$

The above proposition includes the common case of uniform sampling for the Nyström subset. It holds whether the n data points are considered fixed or unobserved.

If retaining all the Nyström principal components then there is no gain in accuracy

compared to Subset PCA from the the perspective of the reconstruction error. Indeed, application of the Nyström method should be discouraged, since it can lead to numerical instabilities from the matrix inverse of K'_{mm}. However, for a smaller PCA dimension the Nyström method will perform strictly better than PCA directly on the subset. Furthermore, other strategies for sampling of the subset may lead to a higher accuracy for the Nyström method even when $d = m$.

6.5 STATISTICAL ACCURACY OF NYSTRÖM KERNEL PCA

In this section we provide a high probability confidence bound on the empirical reconstruction error of kernel PCA with the Nyström method versus the one for full kernel PCA. In line with essentially all results on the statistical accuracy on kernel PCA we assume data has zero mean in feature space. This also leads to a more concise proof and less unwieldy notation.

The actual difference between the reconstruction errors of the Nyström method and standard kernel PCA for a specific dataset is given by

$$R_n(\widetilde{V}_d) - R_n(\hat{V}_d) = \frac{1}{n}\mathrm{Tr}(K) - \sum_{j=1}^{d}\widetilde{\lambda}_j - \sum_{j=d+1}^{m}\hat{\lambda}_j^n \;=\; \hat{\lambda}^n_{<d} \;-\; \widetilde{\lambda}_{<d} \qquad (6.5)$$

However, the eigenvalues $\hat{\lambda}_j^n$ of $\frac{1}{n}K$ are not available – if they were there would be no need to apply the Nyström method. When the Nyström method is being considered for a problem then the size of the data n is very large and calculating the full kernel matrix K, let alone its eigendecomposition, is prohibitively expensive.

At a minimum, any measure of accuracy should not be more computationally

demanding than the method itself, which is $\mathcal{O}(nm^2)$. We present a bound that does not require that we have observed the entire dataset, only the subset $x_1, x_2, ..., x_m$. It takes $\mathcal{O}(m^3)$ time to calculate and is $\mathcal{O}(m^2)$ in memory. It holds for any subsampling distribution.

Theorem 4 (Confidence bound)**.** *With confidence* $1-2e^{-\delta}$ *for* $d = 1, 2, ..., m-1$ *and* $\{x_i\}_{i \notin S} \sim p_X(x)$*, where* $B := \sup_x k(x,x)$*,* $\Phi(\cdot)$ *is the standard normal cumulative distribution function,* $\{\hat{\lambda}_j^m\}_{j=1}^m$ *are the eigenvalues of the kernel matrix* $\frac{1}{m}K_{mm}$ *from the Nyström subset,* $\hat{\lambda}_0^m$ *is defined to be* ∞*, and*

$$D := \frac{n-m}{n}\left(\frac{B\sqrt{2\delta}}{\sqrt{n-m}} + \frac{B^2}{\sqrt{m}}\left(\sqrt{2\log 2} + 2\sqrt{2\pi}\,\Phi\left(-\sqrt{2\log 2}\right)\right)\right)$$

$$D_k := \frac{2D}{\min\left\{\hat{\lambda}_{k-1}^m - \hat{\lambda}_k^m, \hat{\lambda}_{k-1}^m - \hat{\lambda}_{k+1}^m\right\}} \wedge 1$$

we have

$$R_n(\widetilde{V}_d) - R_n(\hat{V}_d) \leq \sum_{j=1}^{d} \hat{\lambda}_j^m \cdot D_j^2 + D \cdot \max_{1 \leq i \leq d} D_i^2$$

The bound becomes infinite if $k(x,x)$ is not bounded for all x. One may create a bounded kernel from an unbounded one through the transformation

$$k'(x,y) := \frac{k(x,y)}{\sqrt{k(x,x)k(y,y)}} \tag{6.6}$$

which has $\sup_x k'(x,x) = 1$, although the transformed kernel will have different characteristics compared to the original one, and induces a different RKHS. It corresponds to scaling the feature vectors to have norm one before calculating the inner product.

The bound does not require that we have observed the entire sample. For example,

if data is generated sequentially and iid from $p_X(x)$ then picking the first m points for the Nyström subset is equivalent to sampling all points and then selecting m points uniformly (in the sense that the data points in the subset have the same distribution in both instances).

If data is stored on disk, and reading from disk is expensive, then only m records need to be read in order to calculate the bound, assuming this can be done in such a way as to respect the sampling distribution of the subset of data points[4].

We may also note that if a bound versus the true unknown reconstruction error of kernel PCA is desired, the above bound can be combined with an existing finite-sample confidence bound of standard kernel PCA.

Proof outline

A proof outline is as follows. Please see the appendix for a full proof.

1. Rewrite the difference in reconstruction errors in terms of the eigenpairs of the empirical operators C_n and C_m, to obtain

$$R_n(\widetilde{V}_d) - R_n(\hat{V}_d) \leq \sum_{j=1}^{d} \hat{\lambda}_j^n \left(1 - \langle \hat{\phi}_j^n, \hat{\phi}_j^m \rangle_{\mathcal{H}}^2\right)$$

2. Apply the Davis-Kahan theorem to convert the angle between the eigenvectors into a difference between successive eigenvalues of C_m and the norm of the difference between the empirical operators $\|C_n - C_m\|_{\mathrm{HS}(\mathcal{H})}$

3. Convert the unknown eigenvalues $\hat{\lambda}_j^n$ into the ones based on the observed

[4] In many implementations of the SQL query language, including MySQL and PostgreSQL, this would correspond to appending `LIMIT(m)` to the end of the query, which interrupts it after finding the first m records [Beaulieu, 2020]

data $\hat{\lambda}_j^m$ plus the difference $\|C_n - C_m\|_{\mathrm{HS}(\mathcal{H})}$, using Lidskii's inequality

4. Now $\|C_n - C_m\|_{\mathrm{HS}(\mathcal{H})}$ is the only random and unknown quantity. Split it up into two independent terms through

$$\|C_n - C_m\|_{\mathrm{HS}(\mathcal{H})} \leq \frac{n-m}{n}(\|C_{n-m} - \mathbb{E}[C_{n-m}]\|_{\mathrm{HS}(\mathcal{H})} + \|C_m - \mathbb{E}[C_m]\|_{\mathrm{HS}(\mathcal{H})})$$

where $C_{n-m} = \frac{1}{n-m}\sum_{i=m+1}^{n} z_i \otimes z_i$

5. Apply Hoeffding's inequality in Banach spaces to the first term

6. Write the second term in terms of the evaluation operator, then apply Hoeffding's inequality to the random part, and then calculate its expectation based on the obtained distribution function

6.5.1 A corollary

From the proof of Theorem 4 one can deduce sharper versions of Theorem 7 and Propositions 10 and 11 from Rosasco et al. [2010], by a factor 2 or 4, although at the expense of slightly longer proofs. These follow from showing that $\mathbb{P}(\|C - C_n\|_{\mathrm{HS}(\mathcal{H})} \leq B) = 1$ since C and C_n are positive.

For Theorem 7, the sharper result states that with probability at least $1 - 2e^{-\delta}$ we have

$$\|C - C_n\|_{\mathrm{HS}(\mathcal{H})} \leq \frac{B\sqrt{2\delta}}{\sqrt{n}}$$

The sharper version of Proposition 10 states that with probability $1 - 2e^{-\delta}$

$$\sum_{j=1}^{\infty} \left(\lambda_j - \hat{\lambda}_j^n\right)^2 \leq \frac{2B^2\delta}{n}$$

$$\sup_j |\lambda_j - \hat{\lambda}_j^n| \leq \frac{B\sqrt{2\delta}}{\sqrt{n}}$$

And for Proposition 11 we obtain that also with probability $1 - 2e^{-\delta}$

$$\left|\sum_{j=1}^{\infty} \lambda_j - \sum_{j=1}^{n} \hat{\lambda}_j^n\right| = |\mathrm{Tr}(C) - \mathrm{Tr}(C_n)| \leq \frac{B\sqrt{2\delta}}{\sqrt{n}}$$

6.6 NUMERICAL EXPERIMENTS

In this section we illustrate the method and bound through experiments on real-world datasets with different kernel functions. We first compare the proposed method to a number of other unsupervised learning methods by measuring the reconstruction error on hold-out data sets. We then evaluate the bound and compare it to the actual errors and the errors for PCA using the subset of data points.

The methods and experiments are implemented in the Python programming language and the source code is available at `https://github.com/fredhallgren/nystrompca`. The package can be installed with one simple command using the Python package manager. It includes a command-line tool to run the different experiments with different parameter values and kernel functions.

The principal components are unique only up to a sign, so in the package we switch the sign of the scores and components such that the range of values in each dimension of the scores is mostly positive. This will ensure that we will get exactly the same values for the scores and components every time we run the

algorithm.

For these experiments we convert categorical variables to discrete ones through one-hot encoding. We remove any date or time variables. For comparability we cut each dataset to 1000 data points.

We normalize the input data to have mean zero and variance one. Note that this does not mean that data has zero mean in the feature space. As previously mentioned, normalizing the input data makes the analysis independent of the units used to measure the variables and unaffected by the scale of the variables, which may otherwise dominate the PCA results. Furthermore, it makes it easier to compare results across different data sets and kernel functions and makes the same kernel parameters appropriate for all data sets.

We keep discrete numerical variables in the data, treating them as continuous for the purposes of PCA. We also remove variables that are constant. These will differ depending on how many data points we include in the total dataset when we run the experiments. We cut eigenvalues that are smaller than 10^{-12} when performing matrix inversions to improve the condition number of the matrix. We also remove any negative eigenvalues – in theory all kernel matrices will be positive definitive, however numerical inaccuracies may occasionally lead to small negative eigenvalues in practice.

We use three different kernel functions, the radial basis functions (RBF), polynomial and Cauchy kernels, described in Section 1.4.2. For the RBF and Cauchy kernels we set the bandwidth to $\sigma = 1$ and for the polynomial kernel we use $R = 1$ and $d = 2$. The RBF and Cauchy kernels are bounded by $\sup_x k(x, x) = 1$ and we normalize the polynomial kernel according to Equation (6.6) before applying it in the experiments. All these kernels are positive definite – in practice occasionally non-positive definite kernels are used [Schölkopf, 2001], but then

the results in this chapter no longer hold. The software package includes a number of additional kernel functions that can be used when running either of the experiments.

6.6.1 Methods comparison

We compare the proposed methods to other unsupervised learning techniques to evaluate its behaviour. We compare with linear PCA, full kernel PCA, sparse PCA [Wang et al., 2016], locally linear embeddings (LLE), a manifold method [Roweis and Saul, 2000] and independent component analysis (ICA) [Hyvärinen and Oja, 2000]. We run the methods for all the datasets in Table 1.1 above. We split each dataset randomly in half, fitting the methods on one half and then evaluating them on the other half. We compare the fraction of variances captured for the different methods for different dimensions. For kernel PCA and Nyström kernel PCA we measure the variances captured in the RKHS and not in the input space. For this experiment we only display the results for the RBF kernel. Please see Table 6.1 for the results.

Sparse PCA is computationally demanding for very high-dimensional data so we don't run it for all the datasets.

Note that the purpose of each of these methods is not necessarily to capture as much variance as possible, however it can still be enlightening to contrast this quantity between different methods. Furthermore, since linear PCA acts in the input space and kernel PCA and its derivations act in the feature space, comparison of the amount of variance captured are not necessarily clear-cut.

Dataset	*d*	*Subset PCA*	*Nyström*	*Kernel PCA*	*Linear PCA*	*Sparse PCA*	*LLE*	*ICA*
magic								
	1	0.2116	0.2268	0.2274	0.5126	0.5056	0.1050	0.0937
	2	0.3459	0.3593	0.3610	0.6257	0.6090	0.2223	0.1798
	3	0.4089	0.4246	0.4269	0.7155	0.7080	0.3035	0.2630
	4	0.4752	0.4912	0.4923	0.7929	0.7342	0.3805	0.3488
	5	0.5292	0.5506	0.5539	0.8635	0.7841	0.5017	0.3488
	6	0.5584	0.5875	0.5933	0.9250	0.8407	0.5966	0.5535
	7	0.5897	0.6230	0.6291	0.9633	0.8647	0.7055	0.6962
	8	0.6126	0.6467	0.6540	0.9816	0.9008	0.7884	0.6962
	9	0.6300	0.6699	0.6781	0.9978	0.9603	0.9048	0.6962
	10	0.6459	0.6891	0.6982	1.0000	0.9803	1.0000	1.0000
yeast								
	1	0.1264	0.1387	0.1396	0.1214	0.1181	0.0339	0.0431
	2	0.2336	0.2600	0.2614	0.2177	0.2080	0.0784	0.0911
	3	0.3197	0.3756	0.3777	0.2773	0.2648	0.1323	0.1223
	4	0.4391	0.4521	0.4550	0.4057	0.4051	0.1848	0.1977
	5	0.4804	0.4998	0.5037	0.5052	0.4958	0.2299	0.2598
	6	0.5238	0.5435	0.5474	0.5869	0.5972	0.3271	0.3149
	7	0.5559	0.5771	0.5839	0.6484	0.6546	0.3867	0.3655
	8	0.5718	0.6102	0.6169	0.7057	0.7098	0.4344	0.4145
	9	0.6022	0.6439	0.6509	0.7520	0.7505	0.4720	0.4569
	10	0.6346	0.6736	0.6828	0.7988	0.7972	0.5252	0.5056
cardiotocography								
	1	0.1329	0.1351	0.1357	0.2223	-	0.0509	0.0260
	2	0.2183	0.2284	0.2306	0.3564	-	0.0795	0.0521
	3	0.2833	0.3012	0.3043	0.4577	-	0.0928	0.0765
	4	0.3374	0.3556	0.3594	0.5258	-	0.1241	0.1019
	5	0.3766	0.3983	0.4029	0.5782	-	0.1449	0.1264
	6	0.4043	0.4346	0.4399	0.6259	-	0.1763	0.1539
	7	0.4342	0.4672	0.4738	0.6636	-	0.2149	0.1790
	8	0.4594	0.5001	0.5061	0.7013	-	0.2468	0.2051
	9	0.4791	0.5217	0.5312	0.7342	-	0.2682	0.2296
	10	0.5056	0.5467	0.5571	0.7687	-	0.3122	0.2576
segmentation								
	1	0.2563	0.2620	0.2621	0.3107	0.3044	0.0222	0.0387
	2	0.3871	0.3952	0.3955	0.5541	0.5468	0.0580	0.1223
	3	0.4988	0.5040	0.5044	0.6369	0.6165	0.1555	0.1573
	4	0.5494	0.5556	0.5565	0.6787	0.6539	0.1885	0.1934
	5	0.6017	0.6039	0.6048	0.7274	0.6971	0.2690	0.2429
	6	0.6434	0.6535	0.6543	0.7930	0.7649	0.3116	0.3109
	7	0.6785	0.6886	0.6921	0.8427	0.7969	0.3733	0.3676
	8	0.6967	0.7102	0.7139	0.8816	0.8190	0.4278	0.4284
	9	0.7147	0.7295	0.7332	0.9087	0.8558	0.4412	0.4739
	10	0.7314	0.7469	0.7513	0.9665	0.9127	0.5149	0.6188

Table continues on the next page

Dataset	d	*Subset PCA*	*Nyström*	*Kernel PCA*	*Linear PCA*	*Sparse PCA*	*LLE*	*ICA*
drug								
	1	0.1342	0.1398	0.1425	0.2316	-	0.0376	0.0278
	2	0.1688	0.1791	0.1837	0.3031	-	0.0602	0.0573
	3	0.2010	0.2219	0.2284	0.3594	-	0.1104	0.0874
	4	0.2261	0.2464	0.2538	0.4059	-	0.1226	0.1149
	5	0.2446	0.2734	0.2827	0.4462	-	0.1559	0.1418
	6	0.2773	0.3006	0.3105	0.4846	-	0.1944	0.1699
	7	0.2970	0.3217	0.3338	0.5094	-	0.2362	0.1905
	8	0.3162	0.3487	0.3631	0.5515	-	0.3044	0.2276
	9	0.3330	0.3648	0.3805	0.5791	-	0.3405	0.2530
	10	0.3483	0.3807	0.3977	0.6045	-	0.3771	0.2766
digits								
	1	0.0715	0.0734	0.0749	0.1261	-	0.0190	0.0148
	2	0.1459	0.1542	0.1572	0.2261	-	0.0310	0.0289
	3	0.2072	0.2163	0.2210	0.3156	-	0.0516	0.0442
	4	0.2530	0.2684	0.2754	0.3914	-	0.0730	0.0595
	5	0.2960	0.3093	0.3183	0.4526	-	0.0876	0.0772
	6	0.3220	0.3403	0.3509	0.4946	-	0.1208	0.0914
	7	0.3502	0.3718	0.3847	0.5353	-	0.1479	0.1067
	8	0.3730	0.3976	0.4122	0.5693	-	0.1590	0.1221
	9	0.3984	0.4203	0.4371	0.6018	-	0.2006	0.1373
	10	0.4113	0.4425	0.4624	0.6321	-	0.2258	0.1534
dailykos								
	1	0.0879	0.0856	0.0843	0.0079	-	0.0086	0.0033
	2	0.0917	0.0915	0.0914	0.0096	-	0.0094	0.0040
	3	0.0918	0.0915	0.0926	0.0109	-	0.0147	0.0048
	4	0.0918	0.0915	0.0932	0.0119	-	0.0155	0.0054
	5	0.0918	0.0915	0.0935	0.0126	-	0.0174	0.0058
	6	0.0918	0.0915	0.0939	0.0131	-	0.0201	0.0062
	7	0.0918	0.0915	0.0939	0.0135	-	0.0218	0.0065
	8	0.0918	0.0915	0.0940	0.0140	-	0.0262	0.0069
	9	0.0919	0.0916	0.0940	0.0143	-	0.0335	0.0071
	10	0.0921	0.0916	0.0940	0.0147	-	0.0368	0.0074
neurips								
	1	0.1035	0.0479	0.0435	0.0011	-	0.0039	0.0046
	2	0.1036	0.0480	0.0439	0.0024	-	0.0084	0.0114
	3	0.1037	0.0482	0.0443	0.0034	-	0.0088	0.0164
	4	0.1037	0.0482	0.0445	0.0037	-	0.0088	0.0187
	5	0.1038	0.0482	0.0446	0.0039	-	0.0089	0.0195
	6	0.1040	0.0483	0.0447	0.0042	-	0.0133	0.0216
	7	0.1040	0.0483	0.0448	0.0045	-	0.0161	0.0242
	8	0.1040	0.0483	0.0449	0.0049	-	0.0189	0.0267
	9	0.1041	0.0484	0.0451	0.0051	-	0.0217	0.0281
	10	0.1041	0.0484	0.0451	0.0054	-	0.0236	0.0305

Table 6.1: Comparison of the variance captured by different dimensionality reduction methods across the maximum dimension d

Nyström kernel PCA generally captures more variance than Subset PCA, apart from the two bag-of-words datasets (`dailykos` and `neurips`). Since we are calculating the reconstruction error on a hold-out dataset it's possible that Subset PCA achieves better performance – we know this to be impossible for the training dataset by Proposition 3.1. For datasets with a small number of dimensions standard linear PCA captures the most amount of variance whilst being simpler and more computationally efficient. For all datasets the performance of Nyström kernel PCA is very close to the method it is attempting to approximate, despite being many times more efficient.

Calculation of Nyström kernel PCA takes on average 0.988 seconds across the eight datasets on an AWS EC2 m5.large instance with an Intel Xeon® Platinum 8175M CPU[5] running Ubuntu Server 20.04, versus 2.753 seconds for full kernel PCA ($n = 500$, $m = 100$). In both instances the kernel matrices are created in Python whilst the eigendecomposition uses built-in LAPACK routines written in Fortran[6]. For these values of n and m the cubic time complexity is not attained and the constant, linear and quadratic factors are still important.

6.6.2 Bound evaluation

To evaluate the Nyström kernel PCA algorithm and the bound as applied to data we compare them to the actual difference between the Nyström reconstruction error and the standard one, as well to the difference between the standard reconstruction error and the reconstruction error for PCA directly on the subset. These quantities are generally not available when applying the Nyström method since they depend on the eigenvalues of the full kernel matrix, but we calculate them here for purposes of illustration.

[5] `https://aws.amazon.com/ec2/instance-types/`
[6] `https://numpy.org/devdocs/reference/generated/numpy.linalg.eigh.html`

Note that the confidence bound does not quite measure the difference between the Nyström reconstruction error and the actual full reconstruction error as calculated in these experiments, but hold for arbitrary values for the data points $x_{m+1}, x_{m+2}, ..., x_n$ with a certain probability, not just the actual ones observed in the considered datasets. In other words these data points are not the ground truth.

We use a Nyström subset of size $m = 100$ which we sample uniformly without replacement. We calculate the bound for PCA dimensions 1 through 10 and use a confidence level of 0.9 when calculating the bound. We run the experiments for multiple samples of the Nyström subset and plot the averages for the relevant quantities using 100 samples. The individual runs for different samples are run in parallel to leverage multi-core CPUs.

It is straightforward to re-run the experiments with different values for these parameters, and for different kernel functions, using the supplied command-line tool. If the experiments are run for unbounded kernel functions $k(x, x)$ then the bound will be infinite.

We plot the results of the experiments for the first four datasets in Table 1.1 and the kernels in Table 1.2, for different PCA dimensions below in Figures 6.1, 6.2 and 6.3. Each plot contains

1. The values of the confidence bound ("Conf. bound")
2. The actual difference between the Nyström and standard reconstruction errors, $R_n(\widetilde{V}_d) - R_n(\hat{V}_d^n)$ ("Nyström diff.")
3. The difference between the reconstruction errors of the full dataset onto the subset PCA subspace and the standard PCA subspace, $R_n(\hat{V}_d^m) - R_n(\hat{V}_d^n)$ ("Subset diff.")

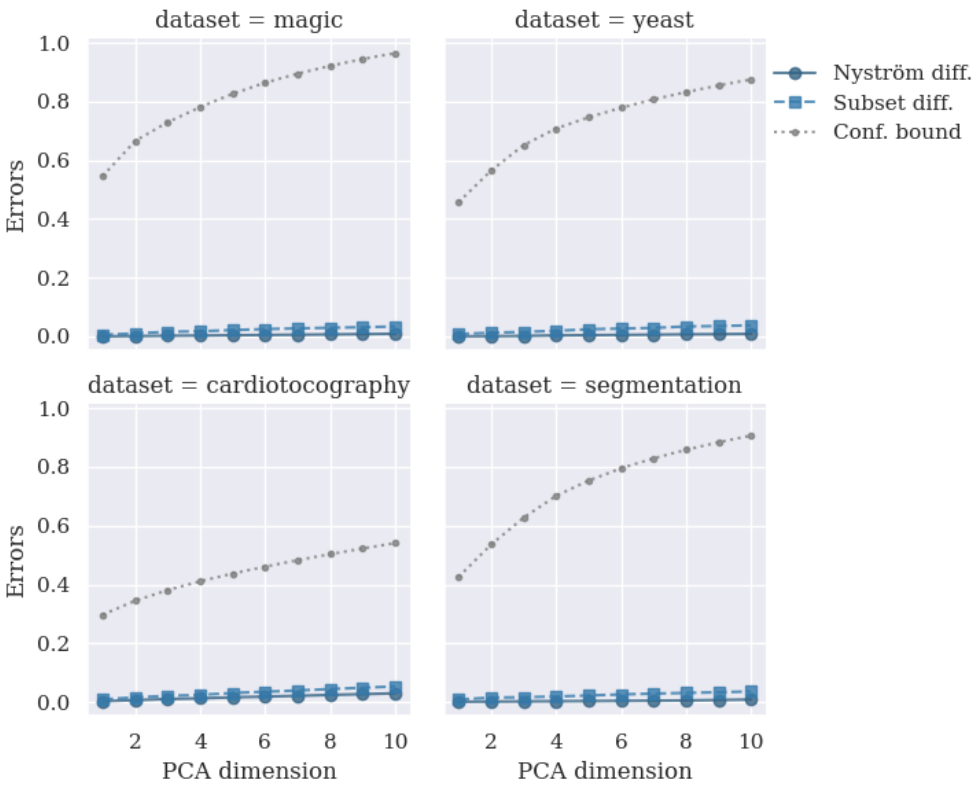

Figure 6.1: Error comparison with the RBF kernel

Both the Nyström difference, the subset difference and the bound increase as the PCA dimension increases. The bound increases more rapidly as the PCA dimension increases from low values, but levels out for larger values as the tail eigenvalues decrease.

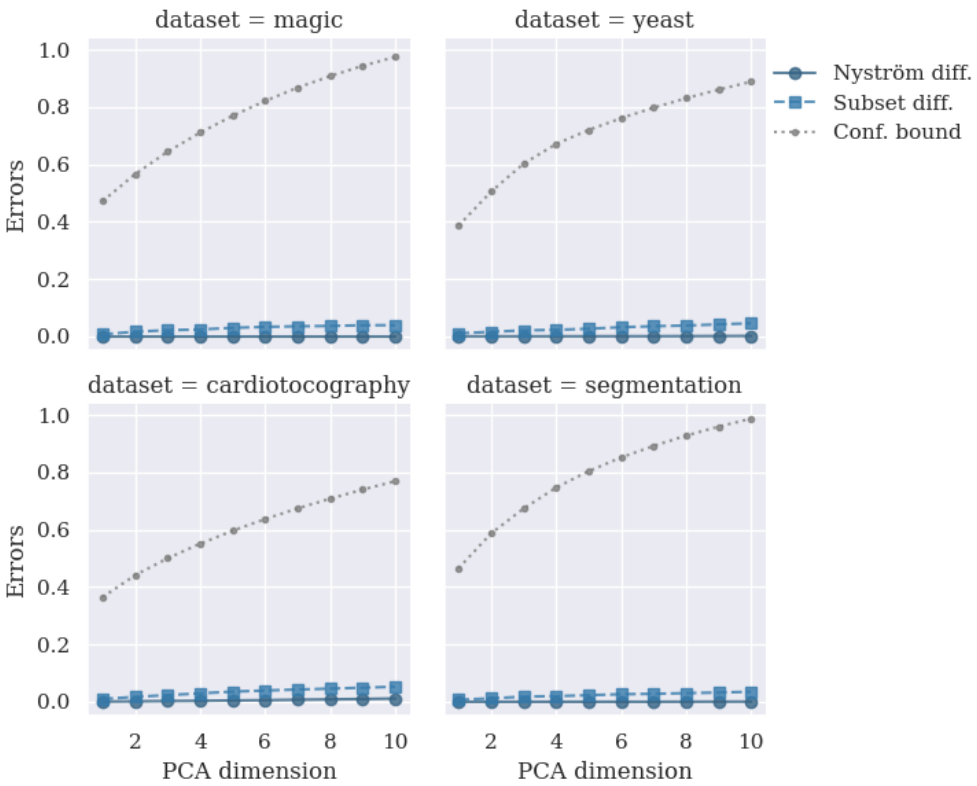

Figure 6.2: Error comparison with the polynomial kernel

The bound seems fairly conservative for these datasets and these choices of hyperparameters. In real-life applications of the Nyström method the datasets are usually much larger, with the number of data points sometimes in the millions, and with much larger n and m the bound will be significantly smaller. The main purpose of the current experiments is rather to investigate differences between datasets and kernel functions and across PCA dimensions.

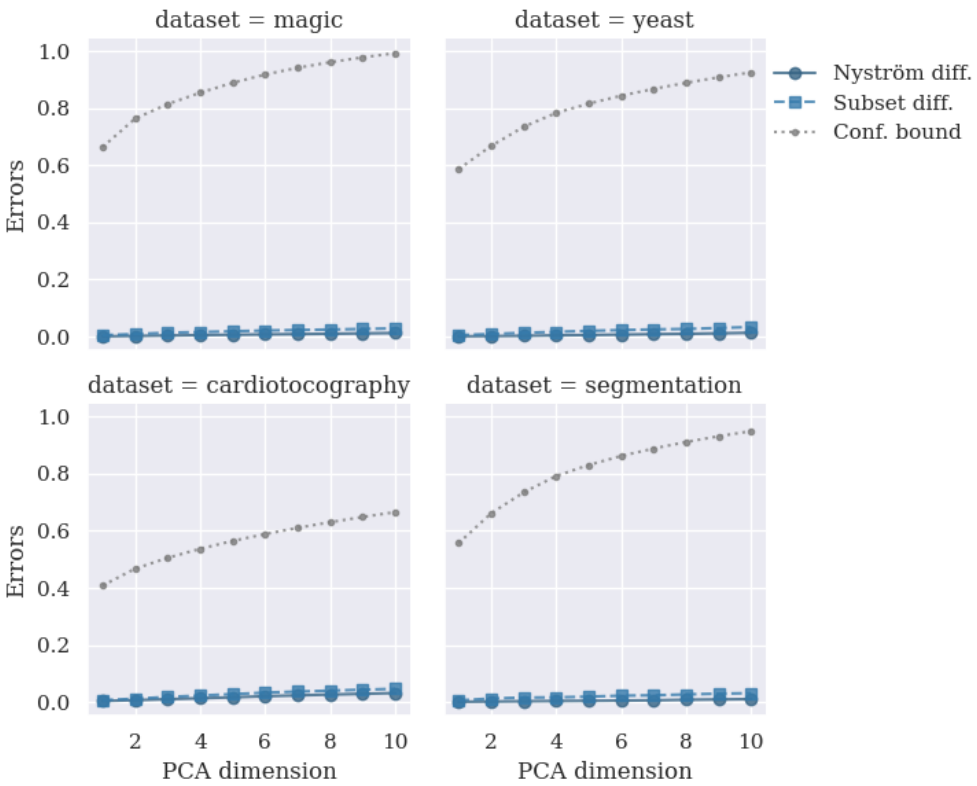

Figure 6.3: Error comparison with the Cauchy kernel

6.7 APPLICATION: NYSTRÖM PRINCIPAL COMPONENT REGRESSION

As an application of Nyström kernel PCA we present kernel principal component regression with the Nyström method, or Nyström kernel PCR. The proposed method may be used for regularized kernel regression, for example as an alternative to kernel ridge regression with the Nyström method. Its derivation demonstrates how the principal scores from Nyström kernel PCA may be used as new data points for supervised learning methods.

We first derive standard kernel PCR without the Nyström method. This derivation

appears to be novel, as previous presentations of kernel principal component regression assumed data to have zero mean in feature space [Rosipal et al., 2000, 2001].

Suppose thus that each data point x_i is paired with an observation of a target variable y_i in $\mathbb{R}$ which we wish to predict using a new observation x^* of the independent variable. The regression model is

$$y = \alpha + S_d\beta + \varepsilon$$

with parameters α and $\beta = (\beta_1, \beta_2, ..., \beta_d)^T$, where $y = (y_1, y_2, ..., y_n)^T$, S_d are the principal scores from kernel PCA with respect to the top d principal components, and ε is a noise vector $\varepsilon = (\varepsilon_1, \varepsilon_2, ..., \varepsilon_n)^T$, whose components we assume are generated from a zero-mean distribution with finite variance $\mathrm{Var}(\varepsilon_i)$. From Section 6.2 the principal scores are given by $S_d = Q_d\Lambda_d^{1/2}$, where $Q_d\Lambda_d Q_d^T$ is the truncated eigendeomposition of K'. Since we assumed Z to be square-integrable we may apply least squares estimation to obtain that [Sen et al., 2010]

$$\hat{\beta} = (S_d^T S_d)^{-1} S_d^T y' = \Lambda_d^{-1/2} Q_d^T y' = S_d^{-1} y'$$

where $y' = (y_1 - \bar{y},\, y_2 - \bar{y},\, ...,\, y_n - \bar{y})^T$. We recall that the principal scores of a new data point x^*, which we centre since we estimated the regression for zero-mean data points, are given by, with respect to the top d principal components

$$w_d^* = \Lambda_d^{-1/2} Q_d^T \kappa'(x^*) = \Lambda_d^{-1/2} Q_d^T \left(\kappa(x^*) - \mathbb{1}_n \kappa(x^*) - K\mathbf{1}_n + \mathbb{1}_n K \mathbf{1}_n\right)$$

and so the prediction for a new data point becomes

$$\hat{y} = \bar{y} + \beta^T w_d^{*T} = \bar{y} + {y'}^T Q_d \Lambda_d^{-1} Q_d^T \kappa'(x^*)$$

For the Nyström method, the principal scores are given by $W = K'_{nm} K'^{-1/2}_{mm} V =$

$K'_{nm}U$, and so the principal scores with respect to the top d principal components are given by $W_d = K'_{nm}K'^{-1/2}_{mm}V_d = K'_{nm}U_d$ where $V_d\widetilde{\Lambda}_d V_d^T$ is the truncated eigendecomposition of $\frac{1}{n}K'^{-1/2}_{mm}K'_{mn}K'_{nm}K'^{-1/2}_{mm}$ and $U_d = K'^{-1/2}_{mm}V_d$. The regression model then becomes

$$y = \alpha + W_d\beta + \varepsilon = \alpha + K'_{nm}U_d\beta + \varepsilon = \alpha + K'_{nm}K'^{-1/2}_{mm}V_d\beta + \varepsilon$$

The least squares parameter estimates are $\hat{\alpha} = \bar{y}$ and

$$\hat{\beta} = (W_d^T W_d)^{-1}W_d^T y' = \left(V_d^T K'^{-1/2}_{mm}K'_{mn}K'_{nm}K'^{-1/2}_{mm}V_d\right)^{-1} V_d^T K'^{-1/2}_{mm}K'_{mn}y'$$

$$= \left((V_d^T V\widetilde{\Lambda}V^T V_d\right)^{-1} V_d^T K'^{-1/2}_{mm}K'_{mn}y' = \widetilde{\Lambda}_d^{-1}V_d^T K'^{-1/2}_{mm}K'_{mn}y' = \widetilde{\Lambda}_d^{-1}U_d^T K'_{mn}y'$$

And so the prediction becomes

$$\hat{y} = \bar{y} + y'^T K'_{nm}U_d\widetilde{\Lambda}_d^{-1}U_d^T\,\widetilde{\kappa}(x^*)$$

We implement kernel principal component regression with the Nyström method (Nyström KPCR) in computer experiments and compare it with Nyström kernel ridge regression (Nyström KRR) [Rudi et al., 2015], which is given by[7]

$$\hat{y} = \bar{y} + \beta^T\kappa(x^*)$$

$$\hat{\beta} = (K_{mn}K_{nm} + \gamma K_{mm})^{-1}K_{mn}y'$$

where $\gamma \geq 0$ is a regularization parameter.

Here we use the `airfoil` dataset. Again we normalize the attributes to have mean 0 and variance 1. Note that we must not normalize the entire dataset at once so as to not introduce look-ahead bias in the regression – when creating a

[7]This is a slightly different specification than in Rudi et al. [2015], where we have demeaned the target variable and subsumed a factor n into the ridge parameter

prediction for a new data point we need to normalize using the mean and variance from the training set.

We use the radial basis functions kernel with parameter $\sigma = 1$. The source code for these experiments is available in the same package at `https://github.com/fredhallgren/nystrompca`. We estimate the regression on a training dataset with a random sample of 75 % of all data points, and evaluate the method on a test set with the remaining data points.

We plot the R^2 for the regression on the test set for different subset sizes m and PCA dimensions d below in Figure 6.4.

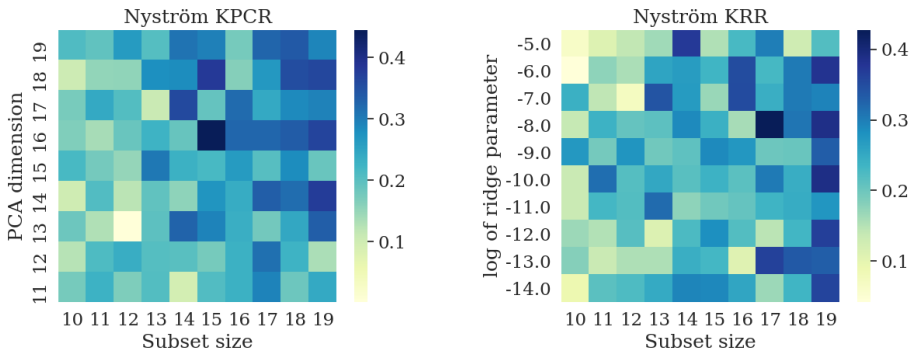

Figure 6.4: Heat maps with regression R^2

For Nyström kernel PCR the regression accuracy improves as we increase the number of principal components used in the regression and as the size of the subset increases. For Nyström KRR the accuracy also improves with a larger subset, but the pattern is less clear as we change the regularization parameter.

To further elucidate the behaviour of the methods we also plot the actual target values versus the predicted ones on the test set for one instance of the parameters. Please see below Figure 6.5. We now use $m = 100$, $d = 90$ and $\gamma = 10^{-11}$. The parameters d and γ were manually tuned to give good results. In this particular example Nyström KPCR obtained an R^2 of 0.73 and Nyström KRR 0.70.

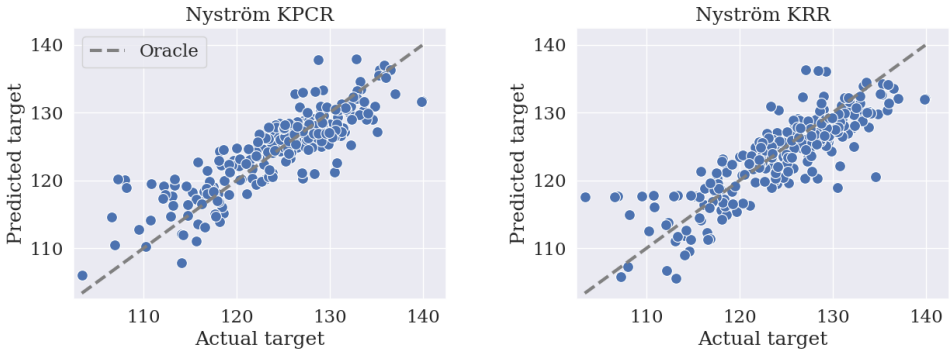

Figure 6.5: Scatter plot with regression predictions

The scatter plots of the predictions versus the actual targets look as expected for an R^2 of around 0.7. The predictions for the two methods look quite similar, but slightly different characteristics are exhibited by the plots due to the different regularization methodologies.

6.8 SUMMARY

In this chapter we have presented an efficient implementation of non-linear PCA by combining kernel PCA with the Nyström method, providing the principal components, explained variance, the principal scores and the reconstruction error. The algorithm centres the data according to the standard definition of PCA.

We further showed that there is little use in applying the Nyström method from the perspective of the reconstruction error when the number of subsampled data points is equal to the PCA dimension. In this case it is preferable to create the principal components directly from only the subset of data points.

We also provided a finite-sample confidence bound on the empirical reconstruction error of the method, which allows us to measure its statistical accuracy before

the entire dataset has been observed. The bound assumes data has zero mean in feature space. It could be adapted to account for centring of data points, although the analysis would become more involved and the notation more unwieldy.

The principal scores from the method may be used instead of the original data matrix in any supervised learning method, in order for example to achieve regularization and denoising. We demonstrated this for linear regression by presenting Nyström kernel principal component regression.

6.9 APPENDIX: PROOFS

In this section we present the proofs of Propositions 3.1 and 3.2, and Theorems 2, 3 and 4.

Proof of Theorem 2.

Standard principal component analysis finds the perpendicular intersecting lines in $\mathbb{R}^d$ along which the variance of the data is successively maximized. These lines are affine subspaces of $\mathbb{R}^d$ which are orthogonal with respect to the associated vector space. To derive kernel PCA with the Nyström method we apply PCA in the span of the subset of data points $\mathcal{H}_S$, i.e. finding the orthogonal one-dimensional affine subspaces of $\mathcal{H}_S$ where the projected data has maximum variance. These are on the form

$$\phi_0 + \langle f_j \rangle = \phi_0 + \{\, a f_j \,|\, a \in \mathbb{R} \,\}$$

where $\phi_0 \in \mathcal{H}_S$ is the translation of the vector space $\langle f_j \rangle$, and the $f_j \in \mathcal{H}_S$, taken to have norm one, are the principal components. It is known from standard PCA that the translation vector is given by the mean of the data points, which in our case is the mean of the data points projected onto $\mathcal{H}_S$. Using $P_{\mathcal{H}_S} = G_m^*(G_m G_m^*)^{-1} G_m = m \cdot G_m^* K_{mm}^{-1} G_m$, where G_m is the sampling operator [Rudi et al., 2015], we obtain

$$\phi_0 = \frac{1}{n}\sum_{r=1}^{n} P_{\mathcal{H}_S}\phi(x_r) = \frac{1}{n}\sum_{r=1}^{n} m \cdot G_m^* K_{mm}^{-1} G_m \phi(x_r)$$

$$= \frac{1}{n}\sum_{r=1}^{n} \sqrt{m} \cdot G_m^* K_{mm}^{-1} \kappa_m(x_r) = \frac{1}{n} K_{nm} K_{mm}^{-1} \kappa_m(x)$$

Any element $\phi \in \mathcal{H}_S$ can be written as $\phi = \phi_0 + \sum_{k=1}^{m} a_k \cdot (\phi(x_k) - \phi_0)$

for some coefficients $a_1, a_2, ..., a_m$ and so the principal components are on the form $f_j = \sum_{k=1}^{m} u_{j,k}(\phi(x_k) - \phi_0)$ with coefficients $u_{j,1}, u_{j,2}, ..., u_{j,m}$. The affine projection of a data point $\phi(x)$ onto $\phi_0 + \langle f_j \rangle$ is then

$$P_{\phi_0+\langle f_j\rangle}\phi(x) = \phi_0 + \langle \phi(x) - \phi_0, f_j\rangle_{\mathcal{H}} f_j$$

The variance of the full dataset along $\phi_0 + \langle f_j \rangle$ then becomes

$$\mathrm{Var}_{f_j}\left(\{\phi(x_i)\}_{i=1}^n\right) =$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\phi_0 + \langle \phi(x_i) - \phi_0, f_j\rangle_{\mathcal{H}} - \frac{1}{n}\sum_{\ell=1}^{n}\left(\phi_0 + \langle \phi(x_\ell) - \phi_0, f_j\rangle_{\mathcal{H}}\right)\right)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left\langle \phi(x_i) - \frac{1}{n}\sum_{\ell=1}^{n}\phi(x_\ell), \sum_{k=1}^{m} u_{j,k}\left(\phi(x_k) - \phi_0\right)\right\rangle_{\mathcal{H}}^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{k=1}^{m} u_{j,k}\left(k_{k,i} - \frac{1}{n}\sum_{\ell=1}^{n} k_{k,\ell} - \langle \phi(x_i), \phi_0\rangle_{\mathcal{H}} + \frac{1}{n}\sum_{\ell=1}^{n}\langle \phi(x_\ell), \phi_0\rangle_{\mathcal{H}}\right)\right)^2$$

Using

$$\langle \phi(x_i), P_{\mathcal{H}_S}\phi(x_r)\rangle_{\mathcal{H}} = \langle \phi(x_i), m \cdot G_m^* K_{mm}^{-1} G_m \phi(x_r)\rangle_{\mathcal{H}}$$

$$= \sqrt{m}\langle \phi(x_i), G_m^* K_{mm}^{-1}\kappa_m(x_r)\rangle_{\mathcal{H}} = \kappa_m(x_i)^T K_{mm}^{-1}\kappa_m(x_r)$$

where $\kappa_m(x) = (k(x_1, x),\ k(x_2, x),\ ...,\ k(x_m, x))^T$, and setting $\kappa_m(x_a) = \kappa_{m,a}$

and $\kappa(a,b) = \kappa_{m,a}^T K_{mm}^{-1} \kappa_{m,b}$, we obtain

$$\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{k=1}^{m} u_{j,k}\left(k_{k,i} - \frac{1}{n}\sum_{\ell=1}^{n} k_{k,\ell} - \langle\phi(x_i),\phi_0\rangle_{\mathcal{H}} + \frac{1}{n}\sum_{\ell=1}^{n}\langle\phi(x_\ell),\phi_0\rangle_{\mathcal{H}}\right)\right)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{k=1}^{m} u_{j,k}\left(k_{k,i} - \frac{1}{n}\sum_{\ell=1}^{n} k_{k,\ell} - \frac{1}{n}\sum_{r=1}^{n}\kappa(i,r) + \frac{1}{n^2}\sum_{\substack{\ell=1\\r=1}}^{n}\kappa(\ell,r)\right)\right)^2$$

$$= \frac{1}{n} u_j^T K'_{mn} K'_{nm} u_j$$

where

$$K'_{mn} = K_{mn} - K_{mn}\mathbb{1}_n - \mathbb{1}_n^{m,n} K_{nm} K_{mm}^{-1} K_{mn} + \mathbb{1}_n^{m,n} K_{nm} K_{m,m}^{-1} K_{mn}\mathbb{1}_n$$

with $\mathbb{1}_n^{m,n}$ an $m \times n$ matrix with each element equal to $\frac{1}{n}$, and $K'_{nm} = K'^T_{mn}$.

The principal components are then given by the orthonormal vectors $f_j = \sum_{k=1}^{m} u_{j,k}(\phi(x_k) - \phi_0)$, $j = 1, 2, ..., m$ that successively maximize the variance. The inner product between two principal components is

$$\langle f_j, f_p\rangle_{\mathcal{H}} = \left\langle \sum_{k=1}^{m} u_{j,k}\left(\phi(x_k) - \phi_0\right), \sum_{q=1}^{m} u_{p,q}\left(\phi(x_q) - \phi_0\right)\right\rangle_{\mathcal{H}}$$

$$= \sum_{\substack{k=1\\q=1}}^{m} u_{j,k} u_{p,q}\left(k_{k,q} - \frac{1}{n}\sum_{r=1}^{n}\kappa(r,k) - \frac{1}{n}\sum_{\ell=1}^{n}\kappa(\ell,q) + \frac{1}{n^2}\sum_{\substack{r=1\\\ell=1}}^{n}\kappa(r,\ell)\right)$$

$$= u_j^T K'_{mm} u_p$$

where $K'_{mm} = K_{mm} - \mathbb{1}_n^{m,n} K_{nm} - K_{mn}\mathbb{1}_n^{n,m} + \mathbb{1}_n^{m,n} K_{nm} K_{mm}^{-1} K_{mn}\mathbb{1}_n^{m,n}$. Maximizing the variance therefore becomes a generalized eigenvalue problem. We

have

$$\langle f_j, f_p \rangle_{\mathcal{H}} = u_j^T K'_{mm} u_p = \left(K'^{1/2}_{mm} u_j\right)^T \left(K'^{1/2}_{mm} u_p\right) := v_j^T v_p$$

where $K'^{1/2}_{mm}$ is the unique positive semi-definite square root of K'_{mm} given by $m \cdot U^m \Lambda^{m\,1/2} U^{m\,T}$, where $U^m \Lambda^m U^{m\,T}$ is the eigendecomposition of $\frac{1}{m} K'_{mm}$. Therefore the variance can be written

$$\frac{1}{n} v_j^T K'^{-1/2}_{mm} K'_{mn} K'_{nm} K'^{-1/2}_{mm} v_j = \left\langle v_j,\ \frac{1}{n} K'^{-1/2}_{mm} K'_{mn} K'_{nm} K'^{-1/2}_{mm} v_j \right\rangle_{\mathbb{R}^m}$$

Then by the Courant-Fischer-Weyl theorem [Bhatia, 1997, Corollary III.1.2] the maximum values over successively orthonormal vectors v_j are given by the eigenvalues of $\frac{1}{n} K'^{-1/2}_{mm} K'_{mn} K'_{nm} K'^{-1/2}_{mm}$, and they occur at its eigenvectors. These eigenvectors will be unique (up to a sign), since all data points are different by assumption.

The principal components are then given by

$$\widetilde{\phi}_j = \sum_{k=1}^{m} u_{j,k} \left(\phi(x_k) - \phi_0\right) \qquad j = 1, 2, ..., m$$

where $u_j = K'^{-1/2}_{mm} v_j$, and the affine subspaces with maximum variances are $\{\, \phi_0 + t\widetilde{\phi}_j \mid t \in \mathbb{R} \,\}$, $j = 1, 2, ..., m$.

The principal score of a centred data point i with respect to the principal component j is given by

$$\begin{aligned} w_{j,i} &= \left\langle \phi(x_i) - \frac{1}{n} \sum_{\ell=1}^{n} \phi(x_\ell),\ \sum_{k=1}^{m} u_{j,k} (\phi(x_k) - \phi_0) \right\rangle_{\mathcal{H}} \\ &= \sum_{k=1}^{m} u_{j,k} \left(k_{k,i} - \frac{1}{n} \sum_{\ell=1}^{n} k_{k,\ell} - \frac{1}{n} \sum_{r=1}^{n} \kappa(i, r) + \frac{1}{n^2} \sum_{\substack{\ell=1 \\ r=1}}^{n} \kappa(\ell, r) \right) \end{aligned}$$

for $j = 1, 2, ..., n$. Or in matrix format

$$(w_{i,j}) = W = K'_{nm}U$$

where $U = K'^{-1/2}_{mm}V$ and $\frac{1}{n}K'^{-1/2}_{mm}K'_{mn}K'_{nm}K'^{-1/2}_{mm} = V\widetilde{\Lambda}V^T$, and so $W = K'_{nm}K'^{-1/2}_{mm}V$.

The scores of a *new* data point x^* which is centred in feature space, i.e. the coordinates of $\phi(x^*) - \frac{1}{n}\sum_{\ell=1}^{n}\phi(x_\ell)$ in terms of the principal components, are given by

$$w_j^* = \left\langle \phi(x^*) - \frac{1}{n}\sum_{\ell=1}^{n}\phi(x_\ell),\ \sum_{k=1}^{m} u_{j,k}\left(\phi(x_k) - \phi_0\right)\right\rangle_{\mathcal{H}}$$

$$= \sum_{k=1}^{m} u_{j,k}\left(k(x_k, x^*) - \frac{1}{n}\sum_{\ell=1}^{n}k_{k,\ell} - \frac{1}{n}\sum_{r=1}^{n}\kappa_{m,r}^T K_{mm}^{-1}\kappa_m(x^*) + \frac{1}{n^2}\sum_{\substack{r=1\\ \ell=1}}^{n}\kappa(r,\ell)\right)$$

or in matrix format

$$w^* = U^T\left(\kappa_m(x^*) - K_{mn}\mathbf{1}_n - \mathbb{1}_n^{m,n}K_{nm}K_{mm}^{-1}\kappa_m(x^*) + \mathbb{1}_n^{m,n}K_{nm}K_{mm}^{-1}K_{mn}\mathbf{1}_n\right)$$

$$:= U^T\,\widetilde{\kappa}(x^*)$$

where $\mathbf{1}_n$ is a length-n column vector given by $\mathbf{1}_n = (\frac{1}{n}, \frac{1}{n}, \ ..., \ \frac{1}{n})^T$.

□

Proof of Theorem 3.

The projection of a data point $\phi(x_i)$ onto a principal component is given by

$$P_{\hat{\phi}_j^{m,n}}\phi(x_i) = \frac{1}{\sqrt{m\hat{\lambda}_j^m}} \sum_{k=1}^{m} u_{j,k}^m \langle \phi(x_i),\, \phi(x_k) - \phi_0 \rangle_{\mathcal{H}}\, \hat{\phi}_j^{m,n}$$

$$= \frac{1}{\sqrt{m\hat{\lambda}_j^m}} \sum_{k=1}^{m} u_{j,k}^m \left(k(x_k, x_i) - \frac{1}{n} K_{nm} K_{mm}^{-1} \kappa_m(x_i) \right) \hat{\phi}_j^{m,n}$$

where $(\hat{\lambda}_j^m, u_j^m)$ is the jth eigenpair of $\frac{1}{m}K'_{mm}$ and $u_{j,k}^m$ is the kth element of u_j^m [Shawe-Taylor et al., 2005].

The projection of a centred data point $\phi'(x_i)$ is then, similarly to Theorem 2, with $k_{a,b} := k(x_a, x_b)$ and $\kappa_m(x_a) = \kappa_{m,a}$

$$P_{\hat{\phi}_j^{m,n}}\phi'(x_i) = \frac{1}{\sqrt{m\hat{\lambda}_j^m}} \sum_{k=1}^{m} u_{j,k}^m \left\langle \phi(x_i) - \frac{1}{n}\sum_{\ell=1}^{n} \phi(x_\ell),\, \phi(x_k) - \phi_0 \right\rangle_{\mathcal{H}} \hat{\phi}_j^{m,n}$$

$$= \frac{1}{\sqrt{m\hat{\lambda}_j^m}} \sum_{k=1}^{m} u_{j,k}^m \left(k_{k,i} - \frac{1}{n}\sum_{\ell=1}^{n} k_{k,\ell} - \frac{1}{n}\sum_{r=1}^{n} \kappa(i,r) + \frac{1}{n^2}\sum_{\substack{\ell=1 \\ r=1}}^{n} \kappa(\ell, r) \right) \hat{\phi}_j^{m,n}$$

Taking the norm and summing over $\phi(x_1), \phi(x_2), ..., \phi(x_n)$ we obtain

$$\frac{1}{n}\sum_{i=1}^{n} \|P_{\hat{\phi}_j^{m,n}}\phi'(x_i)\|_{\mathcal{H}}^2 =$$

$$\frac{1}{n \cdot m\hat{\lambda}_j^m} \sum_{i=1}^{n} \left(\sum_{k=1}^{m} u_{j,k}^m \left(k_{k,i} - \frac{1}{n}\sum_{\ell=1}^{n} k_{k,\ell} - \frac{1}{n}\sum_{r=1}^{n} \kappa(i,r) + \frac{1}{n^2}\sum_{\substack{\ell=1 \\ r=1}}^{n} \kappa(\ell, r) \right) \right)^2$$

$$= \frac{1}{n \cdot m\hat{\lambda}_j^m} u_j^{m\,T} K'_{mn} K'_{nm} u_j^m =: \hat{\lambda}_j^{m,n}$$

For the reconstruction error we have

$$\begin{aligned} R_n(\hat{V}_d^m) &= \frac{1}{n}\sum_{i=1}^{n} \|\phi'(x_i) - P_{\hat{V}_d^m}\phi'(x_i)\|_{\mathcal{H}}^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} \|\phi'(x_i)\|_{\mathcal{H}} - \frac{1}{n}\sum_{i=1}^{n} \left\| P_{\hat{V}_d^m}\phi'(x_i) \right\|_{\mathcal{H}} \\ &= \frac{1}{n}\mathrm{Tr}(K') - \frac{1}{n}\sum_{i=1}^{n} \left\| P_{\hat{V}_d^m}\phi'(x_i) \right\|_{\mathcal{H}} \end{aligned}$$

And so similarly to above, the second term becomes

$$\frac{1}{n}\sum_{i=1}^{n} \left\| P_{\hat{V}_d^m}\phi'(x_i) \right\|_{\mathcal{H}}^2 =$$

$$\frac{1}{n}\sum_{i=1}^{n} \left(\sum_{j=1}^{d} \frac{1}{\sqrt{m\hat{\lambda}_j^m}} \sum_{k=1}^{m} u_{j,k}^m \left(k_{k,i} - \frac{1}{n}\sum_{\ell=1}^{n} k_{k,\ell} - \frac{1}{n}\sum_{r=1}^{n} \kappa(i,r) + \frac{1}{n^2}\sum_{\substack{\ell=1 \\ r=1}}^{n} \kappa(\ell,r) \right) \right)^2$$

$$= \frac{1}{n \cdot m} \mathrm{Tr}(K'_{nm} U_d^m \Lambda_d^{m-1} U_d^{mT} K'_{mn})$$

with $U_d^m \Lambda_d^m U_d^{mT}$ the truncated eigendecomposition of $\frac{1}{m}K'_{mm}$.

□

Proof of Proposition 3.1.

Since $\hat{V}_d^m \subset \mathcal{H}_S$ for any d and by Theorem 2

$$\widetilde{\lambda}_{<d} = \max_{\substack{\dim(V)=d \\ a+V \subset \mathcal{H}_S}} \frac{1}{n} \sum_{i=1}^{n} \|P_{a+V} z_i\|_{\mathcal{H}}^2 = \max_{\substack{\dim(V)=d \\ V \subset \mathcal{H}_S \\ a \in \mathcal{H}_S}} \frac{1}{n} \sum_{i=1}^{n} \|P_V (z_i - a)\|_{\mathcal{H}}^2$$

$$\geq \frac{1}{n} \sum_{i=1}^{n} \|P_{\hat{V}_d^m}(z_i - \phi_0)\|_{\mathcal{H}}^2 = \hat{\lambda}_{<d}^{m,n}$$

The case $d = m$ follows since both $\langle \{\hat{\phi}_j^{m,n}\}_{j=1}^m \rangle$ and $\langle \{\widetilde{\phi}_j\}_{j=1}^m \rangle$ capture the full variance of the data in $\mathcal{H}_S$.

□

Proof of Proposition 3.2.

By the previous proposition we have $\widetilde{V}_m = \hat{V}_m^m$ for a fixed ω and so we will have $\widetilde{V}_m \stackrel{d}{=} \hat{V}_m^m$ if $\{X_{i_1}, X_{i_2}, ..., X_{i_m}\} \stackrel{d}{=} \{X_1, X_2, ..., X_m\}$, where $S = \{i_1, i_2, ..., i_m\}$ are the indices for the subsampled data points. By the law of total probability

$$\mathbb{P}(\{X_{i_1} \leq a_1, X_{i_2} \leq a_2, \ ..., \ X_{i_m} \leq a_m\})$$

$$= \sum_S \mathbb{P}(\{X_{i_1} \leq a_1, X_{i_2} \leq a_2, \ ..., \ X_{i_m} \leq a_m\}|S)\mathbb{P}(S)$$

$$= \sum_S \mathbb{P}(\{X_1 \leq a_1, X_2 \leq a_2, \ ..., \ X_m \leq a_m\}|S)\mathbb{P}(S)$$

since conditional on the sample S, we have m random variables generated according to $\mathbb{P}_X$, which we can take to be $X_1, X_2, ..., X_m$. If the subsampling is

independent of the data then

$$\sum_S \mathbb{P}(\{X_1 \leq a_1, X_2 \leq a_2,\ ...,\ X_m \leq a_m\}|S)\mathbb{P}(S)$$

$$= \mathbb{P}(\{X_1 \leq a_1, X_2 \leq a_2,\ ...,\ X_m \leq a_m\}) \sum_S \mathbb{P}(S) = \prod_{k=1}^{m} \mathbb{P}(\{X_k \leq a_k\})$$

so the subsampled data points are generated i.i.d. from $\mathbb{P}_X$. We can therefore conclude that $\widetilde{V}_m \overset{d}{=} \hat{V}_m^m$. Since Z has the same distribution $\mathbb{P}_Z$ regardless of the subspace and since $\widetilde{V}_m \overset{d}{=} \hat{V}_m^m$ we have $P_{\widetilde{V}_m} Z' \overset{d}{=} P_{\hat{V}_m^m} Z'$ and can conclude that, since Z is square-integrable

$$\mathbb{E}[\|P_{\widetilde{V}_m} Z' - Z'\|_{\mathcal{H}}^2] = \mathbb{E}[\|P_{\hat{V}_m^m} Z' - Z'\|_{\mathcal{H}}^2]$$

and so $R(\widetilde{V}_m) = R(\hat{V}_m^m)$ when $p(S \mid x_1, x_2, ..., x_n) = p(S)$. □

Proof of Theorem 4.

The difference in errors can be rewritten through

$$
\begin{aligned}
& R_n(\widetilde{V}_d) - R_n(\hat{V}_d) = \\
& = \min_{\substack{\dim(V)=d \\ V \subset \mathcal{H}_S}} \frac{1}{n} \sum_{i=1}^{n} \|P_V z_i - z_i\|_{\mathcal{H}}^2 - \min_{\dim(V)=d} \frac{1}{n} \sum_{i=1}^{n} \|P_V z_i - z_i\|_{\mathcal{H}}^2 \\
& = \max_{\dim(V)=d} \frac{1}{n} \sum_{i=1}^{n} \|P_V z_i\|_{\mathcal{H}}^2 - \max_{\substack{\dim(V)=d \\ V \subset \mathcal{H}_S}} \frac{1}{n} \sum_{i=1}^{n} \|P_V z_i\|_{\mathcal{H}}^2 \\
& = \frac{1}{n} \sum_{i=1}^{n} \|(P_{\mathcal{H}_S} + P_{\mathcal{H}_S^\perp}) P_{\hat{V}_d} z_i\|_{\mathcal{H}}^2 - \max_{\substack{\dim(V)=d \\ V \subset \mathcal{H}_S}} \frac{1}{n} \sum_{i=1}^{n} \|P_V z_i\|_{\mathcal{H}}^2 \\
& \leq \frac{1}{n} \sum_{i=1}^{n} \|P_{\mathcal{H}_S} P_{\hat{V}_d} z_i\|_{\mathcal{H}}^2 + \frac{1}{n} \sum_{i=1}^{n} \|P_{\mathcal{H}_S^\perp} P_{\hat{V}_d} z_i\|_{\mathcal{H}}^2 - \max_{\substack{\dim(V)=d \\ V \subset \mathcal{H}_S}} \frac{1}{n} \sum_{i=1}^{n} \|P_V z_i\|_{\mathcal{H}}^2 \\
& \leq \frac{1}{n} \sum_{i=1}^{n} \|P_{\mathcal{H}_S^\perp} P_{\hat{V}_d} z_i\|_{\mathcal{H}}^2
\end{aligned}
$$

Expanding the projection operator $P_{\hat{V}_d}$ we obtain

$$
\begin{aligned}
& \frac{1}{n} \sum_{i=1}^{n} \|P_{\mathcal{H}_S^\perp} P_{\hat{V}_d} z_i\|_{\mathcal{H}_S}^2 = \frac{1}{n} \sum_{i=1}^{n} \left\| P_{\mathcal{H}_S^\perp} \sum_{j=1}^{d} \langle z_i, \hat{\phi}_j^n \rangle_{\mathcal{H}} \hat{\phi}_j^n \right\|_{\mathcal{H}}^2 \\
& = \frac{1}{n} \sum_{i=1}^{n} \left\| \sum_{j=1}^{d} \langle z_i, \hat{\phi}_j^n \rangle_{\mathcal{H}} P_{\mathcal{H}_S^\perp} \hat{\phi}_j^n \right\|_{\mathcal{H}}^2 \leq \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{d} \left\| \langle z_i, \hat{\phi}_j^n \rangle_{\mathcal{H}} P_{\mathcal{H}_S^\perp} \hat{\phi}_j^n \right\|_{\mathcal{H}}^2
\end{aligned}
$$

The last inequality is fairly tight. It becomes an equality without the projection $P_{\mathcal{H}_S^\perp}$, and the further the projection is from the identity, the smaller the norm of the resulting vector.

Now we have

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}\left\|\langle z_i,\hat{\phi}_j^n\rangle_{\mathcal{H}}P_{\mathcal{H}_S^\perp}\hat{\phi}_j^n\right\|_{\mathcal{H}}^2=\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}|\langle z_i,\hat{\phi}_j^n\rangle_{\mathcal{H}}|^2\left\|P_{\mathcal{H}_S^\perp}\hat{\phi}_j^n\right\|_{\mathcal{H}}^2$$

$$=\sum_{j=1}^{d}\left(\frac{1}{n}\sum_{i=1}^{n}|\langle z_i,\hat{\phi}_j^n\rangle_{\mathcal{H}}|^2\right)\left\|P_{\mathcal{H}_S^\perp}\hat{\phi}_j^n\right\|_{\mathcal{H}}^2=\sum_{j=1}^{d}\hat{\lambda}_j^n\left\|P_{\mathcal{H}_S^\perp}\hat{\phi}_j^n\right\|_{\mathcal{H}}^2$$

Expanding the other projection operator we get

$$\sum_{j=1}^{d}\hat{\lambda}_j^n\left\|P_{\mathcal{H}_S^\perp}\hat{\phi}_j^n\right\|_{\mathcal{H}}^2=\sum_{j=1}^{d}\hat{\lambda}_j^n\left\|\hat{\phi}_j^n-P_{\mathcal{H}_S}\hat{\phi}_j^n\right\|_{\mathcal{H}}^2=\sum_{j=1}^{d}\hat{\lambda}_j^n\left\|\hat{\phi}_j^n-\sum_{k=1}^{m}\langle\hat{\phi}_j^n,\hat{\phi}_k^m\rangle_{\mathcal{H}}\hat{\phi}_k^m\right\|_{\mathcal{H}}^2$$

We may only keep the jth index in the sum over the m data points

$$\sum_{j=1}^{d}\hat{\lambda}_j^n\left\|\hat{\phi}_j^n-\sum_{k=1}^{m}\langle\hat{\phi}_j^n,\hat{\phi}_k^m\rangle_{\mathcal{H}}\hat{\phi}_k^m\right\|_{\mathcal{H}}^2\leq\sum_{j=1}^{d}\hat{\lambda}_j^n\left\|\hat{\phi}_j^n-\langle\hat{\phi}_j^n,\hat{\phi}_j^m\rangle_{\mathcal{H}}\hat{\phi}_j^m\right\|_{\mathcal{H}}^2$$

$$=\sum_{j=1}^{d}\hat{\lambda}_j^n\left(1-\langle\hat{\phi}_j^n,\hat{\phi}_j^m\rangle_{\mathcal{H}}^2\right)=\sum_{j=1}^{d}\hat{\lambda}_j^n\cdot\sin^2\theta_j$$

where θ_j is the angle between $\hat{\phi}_j^m$ and $\hat{\phi}_j^n$.

By the Davis-Kahan theorem [Davis and Kahan, 1970, Yu et al., 2015], defining $\hat{\lambda}_0^m$ to be ∞

$$\sin\theta_j\leq\frac{2\|C_n-C_m\|_{\mathrm{HS}(\mathcal{H})}}{\min\left\{\hat{\lambda}_{j-1}^m-\hat{\lambda}_j^m,\hat{\lambda}_j^m-\hat{\lambda}_{j+1}^m\right\}}\wedge 1=:D_j$$

Next we have, by Lidskii's inequality [Kato, 2013, Chapter 3, Theorem 6.11]

$$\sum_{j=1}^{d} \hat{\lambda}_j^n \cdot D_j^2 = \sum_{j=1}^{d} \left(\hat{\lambda}_j^m + \hat{\lambda}_j^n - \hat{\lambda}_j^m \right) \cdot D_j^2$$

$$\leq \sum_{j=1}^{d} \hat{\lambda}_j^m \cdot D_j^2 + \sum_{j=1}^{d} \left| \hat{\lambda}_j^n - \hat{\lambda}_j^m \right| \cdot D_j^2$$

$$\leq \sum_{j=1}^{d} \hat{\lambda}_j^m \cdot D_j^2 + \|C_n - C_m\|_{\mathrm{HS}(\mathcal{H})} \max_{1 \leq k \leq d} D_k^2$$

Now the only unknown and random quantity is $\|C_n - C_m\|_{\mathrm{HS}(\mathcal{H})}$. It depends both on the unobserved data points $z_{m+1}, z_{m+2}, ..., z_n$ and the observed ones $z_1, z_2, ..., z_m$. First we rewrite it as one term that only contains observed data points and another that only contains unobserved ones

$$\begin{aligned}
\|C_n - C_m\|_{\mathrm{HS}(\mathcal{H})} &= \left\| \frac{1}{n} \sum_{j=1}^{n} \otimes^2 z_j - \frac{1}{m} \sum_{k=1}^{m} \otimes^2 z_k \right\|_{\mathrm{HS}(\mathcal{H})} \\
&= \left\| \frac{1}{n} \sum_{j=m+1}^{n} \otimes^2 z_j - \frac{n-m}{nm} \sum_{k=1}^{m} \otimes^2 z_k \right\|_{\mathrm{HS}(\mathcal{H})} \\
&= \frac{n-m}{n} \left\| \frac{1}{n-m} \sum_{j=m+1}^{n} \otimes^2 z_j - \frac{1}{m} \sum_{k=1}^{m} \otimes^2 z_k \right\|_{\mathrm{HS}(\mathcal{H})} \\
&= \frac{n-m}{n} \|C_{n-m} - C_m\|_{\mathrm{HS}(\mathcal{H})}
\end{aligned}$$

Noting that

$$\mathbb{E}[C_m] = \mathbb{E}\left[\frac{1}{m}\sum_{k=1}^{m} \otimes^2 z_k\right] = \frac{1}{m}\sum_{k=1}^{m} \mathbb{E}\left[\otimes^2 z_k\right] = \mathbb{E}\left[\otimes^2 Z\right]$$

$$= \frac{1}{n-m}\sum_{j=m+1}^{n} \mathbb{E}\left[\otimes^2 z_j\right] = \mathbb{E}[C_{n-m}]$$

we may split the norm up into two separate independent norms

$$\begin{aligned} &\frac{n-m}{n}\left\|C_{n-m} - C_m\right\|_{\mathrm{HS}(\mathcal{H})} \\ &\qquad \leq \frac{n-m}{n}\left(\left\|C_{n-m} - \mathbb{E}[C_{n-m}]\right\|_{\mathrm{HS}(\mathcal{H})} + \left\|C_m - \mathbb{E}[C_m]\right\|_{\mathrm{HS}(\mathcal{H})}\right) \end{aligned} \tag{6.7}$$

If we let $Y_i = \otimes^2 z_j - \mathbb{E}[\otimes^2 Z]$, then the random variables Y_i have zero mean and are bounded by $B := \sup_x k(x,x)$, since both $\otimes^2 z_j$ and $\mathbb{E}[\otimes^2 Z]$ are positive. This can be seen for example as follows.

Consider the Hilbert subspace of $\mathcal{H} \otimes \mathcal{H}$ of positive operators, which is closed and so indeed a Hilbert space. We recall that by the Riesz representation theorem every Hilbert space can be identified with its dual through the isometry $C \mapsto \langle \cdot, C\rangle$. And so $\|C - \mathbb{E}[C]\|_{\mathrm{HS}(\mathcal{H})} = \|f - g\|$ for some bounded linear functionals f, g in $(\mathcal{H} \otimes \mathcal{H})^*$.

To see that each f, g will be positive, we first note that

$$f(T) = \langle T, C\rangle_{\mathrm{HS}(\mathcal{H})} = \sum_{i=1}^{\infty} \langle Te_i, Ce_i\rangle_{\mathcal{H}}$$

for any basis $\{e_i\}$ in $\mathcal{H}$. If we take $\{e_i\}$ to be the eigenvectors of C, arbitrarily extended to a basis for the entire space if $\mathrm{Ker(C)} \neq \{0\}$, we obtain, for each i, that $\langle Te_i, Ce_i\rangle_{\mathcal{H}} = \langle Te_i, \lambda_i e_i\rangle_{\mathcal{H}} = \lambda_i \langle Te_i, e_i\rangle_{\mathcal{H}} \geq 0$ since T is positive and $\lambda_i \geq 0$. And so $f(T) \geq 0$ for each T.

Since f, g are positive everywhere we have $\|f - g\| \leq \max\{\|f\|, \|g\|\} \leq B$.

Then by Hoeffding's inequality in Banach spaces [Pinelis, 1994, Theorem 3.5], we have that with confidence $1 - 2e^{-\delta}$

$$\frac{n-m}{n}\left\|\frac{1}{n-m}\sum_{j=m+1}^{n}\otimes^2 z_j - \mathbb{E}[\otimes^2 Z]\right\|_{\mathcal{H}\otimes\mathcal{H}} \leq \frac{n-m}{n}\frac{\sqrt{2\delta}B}{\sqrt{n-m}}$$

$$= \sqrt{2\delta}B\frac{\sqrt{n-m}}{n}$$

In the second term above in Equation (6.7) the data points are observed but the expectation is unknown. Through an application of the evaluation operator we can still use Hoeffding's inequality to devise a bound as follows. We have, since $E_\omega(a) = a$ if a is constant

$$\left\|\frac{1}{m}\sum_{k=1}^{m}\otimes^2 z_k - \mathbb{E}[\otimes^2 Z]\right\|_{\mathcal{H}\otimes\mathcal{H}} = \left\|\frac{1}{m}\sum_{k=1}^{m}E_{\omega_k}(\otimes^2 Z_k) - \mathbb{E}[\otimes^2 Z]\right\|_{\mathcal{H}\otimes\mathcal{H}}$$

$$= \left\|\frac{1}{m}\sum_{k=1}^{m}E_{\omega_k}\left(\otimes^2 Z_k - \mathbb{E}[\otimes^2 Z]\right)\right\|_{\mathcal{H}\otimes\mathcal{H}}$$

Now since $E_\omega(Z_i) = E_\omega(Z_j)$ even if $i \neq j$

$$\left\|\frac{1}{m}\sum_{k=1}^{m}E_{\omega_k}\left(\otimes^2 Z_k - \mathbb{E}[\otimes^2 Z]\right)\right\|_{\mathcal{H}\otimes\mathcal{H}}$$

$$= \left\|\frac{1}{m^2}\sum_{k=1}^{m}\sum_{j=1}^{m}E_{\omega_k}\left(\otimes^2 Z_j - \mathbb{E}[\otimes^2 Z]\right)\right\|_{\mathcal{H}\otimes\mathcal{H}}$$

$$= \left\|\frac{1}{m}\left(\sum_{k=1}^{m}E_{\omega_k}\right)\left(\frac{1}{m}\sum_{j=1}^{m}\otimes^2 Z_j - \mathbb{E}[\otimes^2 Z]\right)\right\|_{\mathcal{H}\otimes\mathcal{H}}$$

$$\leq \frac{1}{m}\left\|\sum_{k=1}^{m} E_{\omega_k}\right\| \left\|\frac{1}{m}\sum_{k=1}^{m} \otimes^2 Z_k - \mathbb{E}[\otimes^2 Z]\right\|_1$$

$$\leq \frac{1}{m}\sum_{k=1}^{m} \|E_{\omega_k}\| \left\|\frac{1}{m}\sum_{k=1}^{m} \otimes^2 Z_k - \mathbb{E}[\otimes^2 Z]\right\|_1 = B \cdot \left\|\frac{1}{m}\sum_{k=1}^{m} \otimes^2 Z_k - \mathbb{E}[\otimes^2 Z]\right\|_1$$

$$= B \cdot \mathbb{E}\left[\left\|\frac{1}{m}\sum_{k=1}^{m} \otimes^2 Z_k - \mathbb{E}[\otimes^2 Z]\right\|_{\mathcal{H}\otimes\mathcal{H}}\right]$$

Through another application of Hoeffding's inequality we obtain that

$$\mathbb{P}\left(\left\|\frac{1}{m}\sum_{k=1}^{m} \otimes^2 Z_k - \mathbb{E}[\otimes^2 Z]\right\|_{\mathcal{H}\otimes\mathcal{H}} \leq \frac{\sqrt{2\delta}B}{\sqrt{m}}\right) \geq 1 - 2e^{-\delta}$$

We can then bound the distribution function of $\left\|\frac{1}{m}\sum_{k=1}^{m} \otimes^2 Z_k - \mathbb{E}[\otimes^2 Z]\right\|_{\mathcal{H}\otimes\mathcal{H}}$ through

$$\mathbb{P}\left(\left\|\frac{1}{m}\sum_{k=1}^{m} \otimes^2 Z_k - \mathbb{E}[\otimes^2 Z]\right\|_{\mathcal{H}\otimes\mathcal{H}} \leq y\right) \geq \left(1 - 2\exp\left\{-\frac{y^2 m}{2B^2}\right\}\right) \vee 0$$

$$=: F(y)$$

Letting $\sigma^2 := B^2/m$ we obtain

$$\mathbb{E}\left[\left\|\frac{1}{m}\sum_{k=1}^{m} \otimes^2 Z_k - \mathbb{E}[\otimes^2 Z]\right\|_{\mathcal{H}\otimes\mathcal{H}}\right] \leq \int_{\mathbb{R}_+} (1 - F(y))dy$$

$$= \sigma\sqrt{2\log 2} + \int_{\sigma\sqrt{2\log 2}}^{+\infty} 2e^{-y^2/2\sigma^2} dy$$

$$= \sigma\sqrt{2\log 2} + 2\sqrt{2\pi}\sigma \frac{1}{\sqrt{2\pi}\sigma} \int_{\sigma\sqrt{2\log 2}}^{+\infty} e^{-y^2/2\sigma^2} dy$$

$$= \sigma\sqrt{2\log 2} + 2\sqrt{2\pi}\sigma\Phi\left(-\sqrt{2\log 2}\right)$$

Adding together the two terms gives that with confidence $1 - 2e^{-\delta}$

$$\|C_n - C_m\|_{\mathrm{HS}(\mathcal{H})} =$$

$$\leq \frac{n-m}{n}\left(B\frac{\sqrt{2\delta}}{\sqrt{n-m}} + \frac{B^2}{\sqrt{m}}\left(\sqrt{2\log 2} + 2\sqrt{2\pi}\,\Phi\left(-\sqrt{2\log 2}\right)\right)\right)$$

$$=: D$$

and so also with confidence $1 - 2e^{-\delta}$ that

$$D_j \leq \frac{2D}{\min\left\{\hat{\lambda}^m_{j-1} - \hat{\lambda}^m_j, \hat{\lambda}^m_{j-1} - \hat{\lambda}^m_{j+1}\right\}}$$

We recall that the eigenvalues of the empirical covariance operator equal the eigenvalues of the kernel matrix $\frac{1}{m}K_{mm}$, which completes the proof.

□

Chapter 7

Conclusion

Thank you for your interest in this work. We hope that the research presented here will be of interest to people in the statistics and machine learning communities and contribute to the further understanding of both kernel PCA and the Nyström method, and that it may give ideas about future work in related areas.

In this thesis we have attempted to contribute to the theoretical, practical and algorthmic understanding of kernel PCA, including when used with the Nyström method. Kernel PCA has been less widely studied than some other kernel methods, notably regression and the support vector machine, which appear to be well understood from many perspectives. This is not necessarily due to the fact that these two methods may be of more interest in some settings, but because of the particular characteristics of PCA that makes analysis more involved. For example, kernel PCA estimates an entire new subspace and the variances of the data in all dimensions of this subspace, rather than just a vector of regression coefficients and a target value, or hyperplanes in the case of the support vector machine. This also leads to increased computational requirements for kernel PCA compared to regression or other methods, which need to be specifically

handled.

For this or for other reasons many methods and analyses that have been carried out for other methods do not have corresponding analyses for PCA, a gap that is narrowed by our contributions.

Statistical methods carried out on data points in a Hilbert space, such as kernel methods, generalize classical statistical methods where data lives in Euclidean space, and contain these methods as special cases. Much of the analysis carried out in this thesis is therefore also applicable to the classical linear case.

7.1 INCREMENTAL KERNEL PCA

We first presented an incremental algorithm for kernel PCA, which we applied to incremental calculation of the Nyström method. It is one of the few available incremental algorithms for kernel PCA, in particular when the kernel matrix is centred. In this setting only one other incremental algorithm had previously been presented, to the best of our knowledge, and our algorithm is more efficient.

Rank one update algorithms for the eigendecomposition other than the one chosen in this thesis could also be applied to the kernel PCA problem, for potentially improved accuracy and efficiency, including algorithms potentially not yet conceived. Methods from randomized linear algebra could also be applied, such as in the matrix multiplication to update the eigenvectors. Furthermore, it could be straightforward to adapt the proposed algorithm for incremental kernel PCA to only maintain a subset of the eigenvectors and eigenvalues. This would likely improve computational efficiency since it would obviate the $\mathcal{O}(n^3)$ matrix multiplication to update all the eigenvectors. When only the explained variance is of interest, more efficient incremental algorithms could potentially also be developed that only update the eigenvalues and not the full eigendecomposition.

Due to the increasing number of settings where data arrives in a streaming fashion, as evidenced by the growing popularity of software data streaming platforms[1], and due to the recent focus of model drift or parameter drift in applications[2], more efficient incremental versions of batch algorithms will likely become more and more important. The eigendecomposition is a fundamental computational method with expansive use cases across scientific fields, and more research to attempt to find more efficient algorithms would be highly useful.

7.2 STATISTICAL TESTING OF THE NYSTRÖM METHOD

We developed an F-test to select the number of data points to include in the randomly sampled subset of the Nyström method. This contribution appears to be one of the few formal methods presented to select the size of the Nyström subset, and the first time the selection of the Nyström subset is based on a motivation of the method as the true model for the underlying unknown reality.

Other techniques for formal decision-making could potentially also be created to the same end, with different assumptions or other advantages over ours.

Statistical testing using RKHS techniques has a long history and continues to attract considerable research interest. There are still many areas left where such techniques could be profitably applied, both to generalize and improve on equivalent classical tests and in research areas specific to kernel methods.

[1] `https://kafka.apache.org/`
[2] `https://www.ibm.com/uk-en/cloud/watson-studio/drift`

7.3 CROSS-VALIDATION FOR KERNEL PCA

We next adapted a cross-validation procedure for PCA to be used for kernel PCA. To our knowledge this is the first method for cross-validation specifically tailored to kernel PCA with a view to select the number of principal components to retain in a reduced representation for data. Many other ways could potentially be conceived for carrying out cross-validation of kernel PCA or for selecting the number of principal components to retain. Other resampling methods, such as the bootstrap, could also be studied further both for kernel PCA and other kernel methods.

7.4 KERNEL PCA WITH THE NYSTRÖM METHOD

We further presented a method for kernel PCA with the Nyström method for efficient non-linear PCA, including a study of its statistical accuracy. To demonstrate the use of the method we applied it to regression to create kernel principal component regression with the Nyström method.

The Nyström method has been proved in both theory and practice to be an effective method to improve the scalability of kernel methods, but it had not been derived for kernel PCA in line with linear PCA until the work carried out in this thesis. The method provides one of the few available methods to make kernel PCA scalable for large datasets, in particular when not assuming data to have zero mean in feature space. It provides an efficient extension of linear PCA for when non-linear patterns in data are to be analyzed.

The confidence bound we presented makes it possible to measure the statistical accuracy of the method to provide further guidance on when the method should be applied for a specific dataset or give insights into how closely the method approximates the original kernel PCA solution. It bounds the empirical

reconstruction error as obtained from data, and is stated only in terms of known quantities based on the data or the chosen kernel function. It is efficient to calculate and does not require that the full dataset has been observed, only the data points in the Nyström subset.

The approximate Nyström kernel matrix $\widetilde{K} = K_{nm}K_{mm}^{-1}K_{mn}$ may often be used as a drop-in replacement for the original kernel matrix to speed up kernel machines. However, for many methods, like kernel PCA, more work is needed for a complete treatment. There are still many kernel methods where application of the Nyström method is not necessarily trivial and has not been fully derived, including potentially kernel FDA and kernel PLS [Mika et al., 1999, Rosipal and Trejo, 2001].

Kernel PCA is closely related to functional PCA. Functional PCA may also suffer from scalability issues if the individual functions are sampled at a large number of points. It's possible that there are settings where the Nyström method could be successfully applied to functional data analysis for improved computational efficiency.

The proposed method for efficient kernel PCA is also applicable to MDS. Further work is required to determine precisely how the method can be applied to other areas that are related to kernel methods, including metric learning and Gaussian processes.

In addition to linear regression, there are many other methods based on PCA where kernel PCA with the Nyström method could be analyzed and explored, such as when PCA is applied in discriminant analysis or outlier detection.

7.5 SELECTION OF THE NYSTRÖM SUBSET

Throughout this thesis the set of data points in the Nyström subset are selected randomly from the full dataset to provide a subspace in which solutions to statistical problems are found, but these subspaces are not optimized in any way to best suit the problem at hand. In the Gaussian process literature, the so-called *inducing points*, which are analogous to the Nyström subset, are often optimized or selected to be optimal in some sense [Wild et al., 2021]. Many of the contributions in this thesis could benefit from equivalent analyses when the Nyström subspace is optimized instead of selected randomly. In particular, kernel PCA with the Nyström method could inspire a similarly derived method with potentially additional benefits, where the variance is maximized subject only to that the principal components lie in a subspace of dimension m, rather than the m-dimensional subspace created from the randomly subsampled data points.

7.6 A GAP IN THE UNDERSTANDING OF KERNEL PCA

The statistical accuracy of kernel PCA has been widely studied under the assumption of zero-mean data in feature space. To the best of our knowledge, for centred kernel PCA only one bound has been presented in previous literature [Blanchard et al., 2007], but this bound is more conservative than the available bounds for uncentred kernel PCA. This is because the bound is based on a bound for uncentred kernel PCA, with an additional factor to account for the error introduced by mean-adjustment. However, uncentred kernel PCA minimizes the reconstruction error under the constraint of the principal components going through the origin, whereas centred kernel PCA finds the optimal principal components without this restriction, and so the reconstruction error will be much smaller. In our practical experience this can even be by more than a factor 10.

The conclusion becomes that there is no statistical analysis of centred kernel PCA available that is as sharp as those with an assumption of zero-mean data, with some margin. Closing this gap would be a major achievement and in our view one of the most significant potential research contributions for kernel methods in the near term.

Bibliography

S. Aaronson. P = NP? In *Open problems in mathematics*, pages 1–122. Springer, 2016.

D. Achlioptas and F. McSherry. Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM)*, 54(2):9, 2007.

A. Alam and K. Fukumizu. Hyperparameter selection in kernel principal component analysis. 2014.

A. E. Alaoui and M. W. Mahoney. Fast randomized kernel methods with statistical guarantees. *arXiv preprint arXiv:1411.0306*, 2014.

S. Arlot, A. Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2010.

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

F. Bach. On the equivalence between kernel quadrature rules and random feature expansions. *Journal of Machine Learning Research*, 18(1):714–751, 2017.

F. R. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 105–112, 2009.

F. R. Bach. Sharp analysis of low-rank kernel matrix approximations. In *COLT*, volume 30, pages 185–209, 2013.

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

S. Banach. Théorie des opérations linéaires. *Monografie Matematyczne*, 1932.

L. Barreira and C. Valls. *Dynamical systems: an introduction*. Springer Science & Business Media, 2012.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov): 463–482, 2002.

A. Beaulieu. *Learning SQL: Generate, manipulate, and retrieve data*. O'Reilly Media, 3rd edition, 2020.

Y. Bengio, O. Delalleau, N. L. Roux, J.-F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, 16(10):2197–2219, 2004.

C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic analysis on semigroups*. Springer Science & Business Media, 1984.

J. O. Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

P. Besse. Approximation spline de l'analyse en composantes principales d'une variable aléatoire hilbertienne. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 12, pages 329–349. Université Paul Sabatier, 1991.

P. Besse and J. O. Ramsay. Principal components analysis of sampled functions. *Psychometrika*, 51(2):285–311, 1986.

R. Bhatia. *Matrix analysis*, volume 169 of *Graduate texts in mathematics*. Springer Science & Business Media, 1st edition, 1997.

C. M. Bishop. *Pattern recognition and machine learning*. Springer Science & Business Media, 2006.

G. Blanchard, O. Bousquet, and L. Zwald. Statistical properties of kernel principal component analysis. *Machine Learning*, 66(2-3):259–294, 2007.

G. Blom, J. Enger, G. Englund, J. Grandell, and L. Holst. *Sannolikhetsteori och statistikteori med tillämpningar*. Studentlitteratur, 2005.

V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.

B. Bollobás. *Linear analysis*. Cambridge mathematical textbooks. Cambridge University Press, 2nd edition, 1999.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

M. Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20–30, 2006.

R. Bro, K. Kjeldahl, A. Smilde, and H. Kiers. Cross-validation of component models: a critical look at current methods. *Analytical and Bioanalytical Chemistry*, 390(5):1241–1251, 2008.

J. R. Bunch, C. P. Nielsen, and D. C. Sorensen. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31(1):31–48, 1978.

C. Campbell. Kernel methods: a survey of current techniques. *Neurocomputing*, 48(1-4):63–84, 2002.

E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, pages 409–415, 2001.

S. L. Chau, J. Gonzalez, and D. Sejdinovic. RKHS-SHAP: Shapley values for kernel methods. *arXiv preprint arXiv:2110.09167*, 2021.

T.-J. Chin and D. Suter. Incremental kernel principal component analysis. *IEEE Transactions on Image Processing*, 16(6):1662–1674, 2007.

T.-J. Chin, K. Schindler, and D. Suter. Incremental kernel SVD for face recognition with image sets. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 461–466. IEEE, 2006.

I. Chiswell and W. Hodges. *Mathematical logic*, volume 3. Oxford University Press, 2007.

M. B. Cohen, Y. T. Lee, C. Musco, C. Musco, R. Peng, and A. Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190, 2015.

D. L. Cohn. *Measure theory*, volume 165 of *Birkhäuser Advanced Texts Basler Lehrbucher*. Springer Science & Business Media, 2nd edition, 1980.

D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 1–6. ACM, 1987.

T. H. Cormen. *Introduction to algorithms*. MIT press, 2009.

C. Cortes, M. Mohri, and A. Talwalkar. On the impact of kernel approximation on learning accuracy. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 113–120. JMLR Workshop and Conference Proceedings, 2010.

N. Cressie. The origins of kriging. *Mathematical geology*, 22(3):239–252, 1990.

B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. F. Balcan, and L. Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 3041–3049, 2014.

J. Dauxois, A. Pousse, and Y. Romain. Asymptotic theory for the principal component analysis of a vector random function: some applications to statistical inference. *Journal of Multivariate Analysis*, 12(1):136–154, 1982.

E. B. Davies. *Linear operators and their spectra*, volume 106 of *Cambridge studies in advanced mathematics*. Cambridge University Press, 1st edition, 2007.

C. Davis and W. M. Kahan. The rotation of eigenvectors by a perturbation. III. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970.

A. C. Davison and D. V. Hinkley. *Bootstrap methods and their application*. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1st edition, 1997.

F. De La Torre and M. J. Black. A framework for robust subspace learning. *International Journal of Computer Vision*, 54(1-3):117–142, 2003.

T. Dettmers. 8-bit approximations for parallelism in deep learning. *arXiv preprint arXiv:1511.04561*, 2015.

M. Dong, Y. Wang, X. Yang, and J.-H. Xue. Learning local metrics and influential regions for classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(6):1522–1529, 2019.

J. J. Dongarra and D. C. Sorensen. A fully parallel algorithm for the symmetric eigenvalue problem. *SIAM Journal on Scientific and Statistical Computing*, 8 (2):s139–s154, 1987.

P. Drineas and M. W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(Dec):2153–2175, 2005.

P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13(Dec):3475–3506, 2012.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

A. El Alaoui and M. W. Mahoney. Fast randomized kernel methods with statistical guarantees. *stat*, 1050:2, 2014.

T. Fernández, A. Gretton, D. Rindt, and D. Sejdinovic. A kernel log-rank test of independence for right-censored data. *Journal of the American Statistical Association*, pages 1–12, 2021.

M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern recognition*, 41(1):176–190, 2008.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec):243–264, 2001.

D. J. Foster, A. Rakhlin, and K. Sridharan. Online learning: sufficient statistics and the Burkholder method. In S. Bubeck, V. Perchet, and P. Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 3028–3064. PMLR, 06–09 Jul 2018.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Springer Science & Business Media, 2001.

K. Fukumizu, F. R. Bach, and A. Gretton. Statistical consistency of kernel canonical correlation analysis. *Journal of Machine Learning Research*, 8(Feb): 361–383, 2007.

W. A. Fuller. Properties of some estimators for the errors-in-variables model. *The Annals of Statistics*, pages 407–422, 1980.

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

A. Gittens and M. W. Mahoney. Revisiting the Nyström method for improved large-scale machine learning. *Journal of Machine Learning Research*, 17(1): 3977–4041, 2016.

D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys (CSUR)*, 23(1):5–48, 1991.

G. H. Golub. Some modified matrix eigenvalue problems. *Siam Review*, 15(2): 318–334, 1973.

G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 4th edition, 2013.

G. H. Golub, M. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

M. Gönen and E. Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12(Jul):2211–2268, 2011.

C. Graham and D. Talay. *Simulation stochastique et méthodes de Monte-Carlo*. École Polytechnique, Département de Mathématiques Appliquées, 2011.

S. Grégoire. *Processus et estimation*. École Polytechnique, Département de Mathématiques Appliquées, 2010.

M. Gu and S. C. Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 15(4):1266–1276, 1994.

A. Gut. *Probability: a graduate course*, volume 75. Springer Science & Business Media, 2013.

A. Gyori, L. Franklin, D. Dig, and J. Lahoda. Crossing the gap from imperative to functional programming through refactoring. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 543–553. ACM, 2013.

B. Haasdonk and H. Burkhardt. Invariant kernel functions for pattern analysis and machine learning. *Machine Learning*, 68(1):35–61, 2007.

M. Haddouche, B. Guedj, O. Rivasplata, and J. Shawe-Taylor. Upper and lower bounds on the performance of kernel PCA. *arXiv preprint arXiv:2012.10369*, 2020.

J. J. Hall, C. Robbiano, and M. R. Azimi-Sadjadi. Adaptive classification using incremental linearized kernel embedding. *IEEE Transactions on Signal Processing*, 2022.

P. Hall and M. Hosseini-Nasab. On properties of functional principal components analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, 68(1):109–126, 2006.

P. Hall, H.-G. Müller, and J.-L. Wang. Properties of principal component methods for functional and longitudinal data analysis. *The Annals of Statistics*, pages 1493–1517, 2006.

F. Hallgren. Kernel PCA with the Nyström method. *arXiv preprint arXiv:2109.05578*, 2021.

F. Hallgren and P. Northrop. Incremental kernel PCA and the Nyström method. *arXiv preprint arXiv:1802.00043*, 2018.

D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: an overview with application to learning methods. *Neural Computation*, 16 (12):2639–2664, 2004.

S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller. Kernel-based nonlinear blind source separation. *Neural Computation*, 15(5):1089–1124, 2003.

M. T. Heath. *Scientific computing: an introductory survey*, volume 2. McGraw-Hill New York, 2002.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

L. Hoegaerts, L. De Lathauwer, I. Goethals, J. A. Suykens, J. Vandewalle, and B. De Moor. Efficiently updating and tracking the dominant kernel principal components. *Neural Networks*, 20(2):220–229, 2007.

H. Hoffmann. Kernel PCA for novelty detection. *Pattern Recognition*, 40(3): 863–874, 2007.

T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, pages 1171–1220, 2008.

M. C. Hout, M. H. Papesh, and S. D. Goldinger. Multidimensional scaling. *Wiley Interdisciplinary Reviews: Cognitive Science*, 4(1):93–103, 2013.

P. J. Huber. *Robust statistical procedures*. SIAM, 1996.

P. J. Huber. *Robust statistics*. Springer Science & Business Media, 2011.

T. Hungerford. *Algebra*. Springer Science & Business Media, 2003.

Z. Hussain, J. Shawe-Taylor, D. R. Hardoon, and C. Dhanjal. Design and generalization analysis of orthogonal matching pursuit algorithms. *IEEE Transactions on Information Theory*, 57(8):5326–5341, 2011.

A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer Science & Business Media, 2013.

R. Jin, T. Yang, M. Mahdavi, Y.-F. Li, and Z.-H. Zhou. Improved bounds for the Nyström method with application to kernel classification. *IEEE Transactions on Information Theory*, 59(10):6939–6949, 2013.

I. T. Jolliffe. A note on the use of principal components in regression. *Applied Statistics*, pages 300–303, 1982.

I. T. Jolliffe. *Principal component analysis*. Springer Science & Business Media, 2nd edition, 2002.

I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

T. Kato. *Perturbation theory for linear operators*. Springer Science & Business Media, 2013.

K. I. Kim, M. O. Franz, and B. Schölkopf. Iterative kernel principal component analysis for image modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1351–1366, 2005.

D. Knuth. *All Questions Answered*. Seminar at KTH, Stockholm, on 16 January, 2018.

V. Koltchinskii and E. Giné. Random matrix approximation of spectra of integral operators. *Bernoulli*, 6(1):113–167, 2000.

E. Kreyszig. *Introductory functional analysis with applications*. Wiley, 1st edition, 1989.

S. Kumar, M. Mohri, and A. Talwalkar. Sampling techniques for the Nyström method. In *Conference on Artificial Intelligence and Statistics*, pages 304–311, 2009.

S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the Nyström method. *Journal of Machine Learning Research*, 13(1):981–1006, 2012.

T. L. Lai. Sequential analysis: some classical problems and new challenges. *Statistica Sinica*, pages 303–351, 2001.

M. Ledoux and M. Talagrand. *Probability in Banach spaces: isoperimetry and processes*. Springer Science & Business Media, 2013.

Y. Li. On incremental and robust subspace learning. *Pattern Recognition*, 37(7): 1509–1518, 2004.

E. Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 581–588. ACM, 2013.

J. Lim, D. A. Ross, R.-S. Lin, and M.-H. Yang. Incremental learning for visual tracking. In *Advances in Neural Information Processing Systems*, pages 793–800, 2004.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2 (Feb):419–444, 2002.

M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

M. D. McCool, A. D. Robison, and J. Reinders. *Structured parallel programming: patterns for efficient computation*. Elsevier, 2012.

C. McDiarmid. Centering sequences with bounded differences. *Combinatorics, Probability and Computing*, 6(1):79–86, 1997.

B. Mertens, T. Fearn, and M. Thompson. The efficient cross-validation of principal components applied to principal component regression. *Statistics and Computing*, 5(3):227–235, 1995.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In *Neural networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (cat. no. 98th8468)*, pages 41–48. IEEE, 1999.

T. K. Moon. The expectation-maximization algorithm. *IEEE Signal processing Magazine*, 13(6):47–60, 1996.

C. Moore. Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20):2354, 1990.

R. B. Myerson. *Game theory*. Harvard University Press, 2013.

E. J. Nyström. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54(1):185–204, 1930.

E. Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

M. Opper. An approximate inference approach for the pca reconstruction error. *Advances in Neural Information Processing Systems*, 18:1035, 2006.

V. I. Paulsen and M. Raghupathi. *An introduction to the theory of reproducing kernel Hilbert spaces*, volume 152 of *Cambridge studies in advanced mathematics*. Cambridge University Press, 2016.

K. Pearson. Principal components analysis. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 6(2):559, 1901.

I. Pinelis. Optimum bounds for the distributions of martingales in Banach spaces. *The Annals of Probability*, pages 1679–1706, 1994.

I. F. Pinelis and A. I. Sakhanenko. Remarks on inequalities for large deviation probabilities. *Theory of Probability & Its Applications*, 30(1):143–148, 1986.

J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6 (Dec):1939–1959, 2005.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2007.

K. M. Ramachandran and C. P. Tsokos. *Mathematical statistics with applications in R*. Academic Press, 2020.

J. O. Ramsay and B. W. Silverman. *Functional data analysis*. Springer Science & Business Media, 2nd edition, 2005.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

D. Rindt, D. Sejdinovic, and D. Steinsaltz. Consistency of permutation tests of independence using distance covariance, HSIC and dHSIC. *Stat*, 10(1):e364, 2021.

L. Rosasco, M. Belkin, and E. D. Vito. On learning with integral operators. *Journal of Machine Learning Research*, 11(Feb):905–934, 2010.

R. Rosipal and L. J. Trejo. Kernel partial least squares regression in reproducing kernel Hilbert space. *Journal of Machine Learning Research*, 2(Dec):97–123, 2001.

R. Rosipal, L. J. Trejo, and A. Cichocki. *Kernel principal component regression with EM approach to nonlinear principal components extraction*. University of Paisley, 2000.

R. Rosipal, M. Girolami, L. J. Trejo, and A. Cichocki. Kernel PCA for feature extraction and de-noising in nonlinear regression. *Neural Computing & Applications*, 10(3):231–243, 2001.

J. B. Rosser. Highlights of the history of the lambda-calculus. *Annals of the History of Computing*, 6(4):337–349, 1984.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

A. Rudi, R. Camoriano, and L. Rosasco. Less is more: Nyström computational regularization. In *Advances in Neural Information Processing Systems*, pages 1657–1665, 2015.

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. 1998.

B. Schölkopf. The kernel trick for distances. In *Advances in Neural Information Processing Systems*, pages 301–307, 2001.

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

B. Schölkopf, R. Herbrich, and A. Smola. A generalized representer theorem. In *Computational Learning Theory*, pages 416–426. Springer, 2001.

P. K. Sen, J. M. Singer, and A. C. P. De Lima. *From finite sample to asymptotic methods in statistics*, volume 28 of *Cambridge series in statistical and probabilistic mathematics*. Cambridge University Press, 2010.

J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 1st edition, 2004.

J. Shawe-Taylor and C. K. Williams. The stability of kernel principal components analysis and its relation to the process eigenspectrum. *Advances in neural information processing systems*, pages 383–390, 2003.

J. Shawe-Taylor, C. Williams, N. Cristianini, and J. Kandola. On the eigenspectrum of the Gram matrix and its relationship to the operator eigenspectrum. In *International Conference on Algorithmic Learning Theory*, pages 23–40. Springer, 2002.

J. Shawe-Taylor, C. K. Williams, N. Cristianini, and J. Kandola. On the eigenspectrum of the Gram matrix and the generalization error of kernel PCA. *IEEE Transactions on Information Theory*, 51(7):2510–2522, 2005.

F. Sheikholeslami, D. Berberidis, and G. B. Giannakis. Kernel-based low-rank feature extraction on a budget for big data streams. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 928–932. IEEE, 2015.

Y. Shiokawa, Y. Date, and J. Kikuchi. Application of kernel principal component analysis and computational machine learning to exploration of metabolites strongly associated with diet. *Scientific reports*, 8(1):1–8, 2018.

V. D. Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*, pages 705–712, 2002.

R. Singh, M. Sahani, and A. Gretton. Kernel instrumental variable regression. In *Advances in Neural Information Processing Systems*, pages 4593–4605, 2019.

M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition, 2013.

A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

E. Snelson and Z. Ghahramani. Local and global sparse Gaussian process approximations. In *International Conference on Artificial Intelligence and Statistics*, volume 11, pages 524–531. PMLR, 2007.

D. C. Sorensen and P. T. P. Tang. On the orthogonality of eigenvectors computed by divide-and-conquer techniques. *SIAM Journal on Numerical Analysis*, 28 (6):1752–1775, 1991.

N. Sterge and B. Sriperumbudur. Statistical optimality and computational efficiency of Nyström kernel PCA. *arXiv preprint arXiv:2105.08875*, 2021.

N. Sterge, B. Sriperumbudur, L. Rosasco, and A. Rudi. Gain with no pain: efficiency of kernel-PCA by Nyström sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 3642–3652. PMLR, 2020.

M. Stone. An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 44–47, 1977.

V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13 (4):354–356, 1969.

R. Sundberg. *Kompendium i tillämpad matematisk statistik*. KTH, 1981.

Y. Takeuchi, S. Ozawa, and S. Abe. An efficient incremental kernel principal component analysis for online feature selection. In *2007 International Joint Conference on Neural Networks*, pages 2346–2351. IEEE, 2007.

P. Tankov and N. Touzi. *Calcul stochastique en finance*. École Polytechnique, Département de Mathématiques Appliquées, 2010.

J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, 61(3): 611–622, 1999.

T. Tokumoto and S. Ozawa. A fast incremental kernel principal component analysis for learning stream of data chunks. In *International Joint Conference on Neural Networks (IJCNN)*, pages 2881–2888. IEEE, 2011.

L. N. Trefethen and D. Bau. *Numerical linear algebra*. SIAM, 1997.

J. W. Tukey. *Exploratory data analysis*, volume 2. Reading, Mass., 1977.

V. Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

H. X. Vo and L. J. Durlofsky. Regularized kernel PCA for the efficient parameterization of complex geological models. *Journal of Computational Physics*, 322:859–881, 2016.

Q. Wang. Kernel principal component analysis and its applications in face recognition and active shape models. *arXiv preprint arXiv:1207.3538*, 2012.

S. Wang and Z. Zhang. Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling. *Journal of Machine Learning Research*, 14(1):2729–2769, 2013.

S. Wang and Z. Zhang. Efficient algorithms and error analysis for the modified Nyström method. In *Artificial Intelligence and Statistics*, pages 996–1004. PMLR, 2014.

T. Wang, Q. Berthet, and R. J. Samworth. Statistical and computational trade-offs in estimation of sparse principal components. *The Annals of Statistics*, 44(5): 1896–1930, 2016.

A. Wibowo and Y. Yamamoto. A note on kernel principal component regression. *Computational Mathematics and Modeling*, 23(3):350–367, 2012.

D. Widjaja, C. Varon, A. Dorado, J. A. Suykens, and S. Van Huffel. Application of kernel principal component analysis for single-lead-ECG-derived respiration. *IEEE Transactions on Biomedical Engineering*, 59(4):1169–1176, 2012.

V. Wild, M. Kanagawa, and D. Sejdinovic. Connections and equivalences between the Nyström method and sparse variational gaussian processes. *arXiv preprint arXiv:2106.01121*, 2021.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688, 2001.

C. K. Williams. On a connection between kernel PCA and metric multidimensional scaling. *Machine Learning*, 46(1):11–19, 2002.

T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou. Nyström method vs random Fourier features: A theoretical and empirical comparison. *Advances in Neural Information Processing Systems*, 25:476–484, 2012.

Y. Yu, T. Wang, and R. J. Samworth. A useful variant of the Davis–Kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2015.

Q. Zhang, S. Filippi, A. Gretton, and D. Sejdinovic. Large-scale kernel methods for independence testing. *Statistics and Computing*, 28(1):113–130, 2018.

Y. Zhang, J. C. Duchi, and M. J. Wainwright. Divide and conquer kernel ridge regression. In *COLT*, pages 592–617, 2013.