Low-rank matrix factorization in multiple kernel learning

A dissertation presented

by

Martin Stražar

to

The Faculty of Computer and Information Science

in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in the subject of

Computer and Information Science

Ljubljana, 2018

APPROVAL

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

— Martin Stražar, September 2018

The submission has been approved by

dr. Tomaž Curk

Assistant Professor of Computer and Information Science

ADVISOR

University of Ljubljana, Faculty of Computer and Information Science

dr. Matej Kristan

Associate Professor of Computer and Information Science

EXAMINER

University of Ljubljana, Faculty of Computer and Information Science

dr. Bor Plestenjak

Professor of Mathematics

EXAMINER

University of Ljubljana, Faculty of Mathematics and Physics

dr. Dino Sejdinović

Associate Professor in Statistics

EXAMINER

University of Oxford, Department of Statistics

PREVIOUS PUBLICATION

I hereby declare that the research reported herein was previously published/submitted for publication in peer reviewed journals or publicly presented at the following occasions:

[1] M. Stražar, J. Ule, B. Zupan, M. Žitnik, and T. Curk. Orthogonal matrix factorization enables integrative analysis of multiple RNA binding proteins. In Jonathan Wren, editors, volume 31.10 of *Bioinformatics*, pages 1527–1935, Oxford, 2016. Oxford University Press.

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Ljubljana.

University *of Ljubljana*
Faculty *of Computer and Information Science*
Martin Stražar
Low-rank matrix factorization in multiple kernel learning

ABSTRACT

The increased rate of data collection, storage, and availability results in a corresponding interest for data analyses and predictive models based on simultaneous inclusion of multiple data sources. This tendency is ubiquitous in practical applications of machine learning, including recommender systems, social network analysis, finance and computational biology. The heterogeneity and size of the typical datasets calls for simultaneous dimensionality reduction and inference from multiple data sources in a single model. Matrix factorization and multiple kernel learning models are two general approaches that satisfy this goal. This work focuses on two specific goals, namely i) finding interpretable, non-overlapping (orthogonal) data representations through matrix factorization and ii) regression with multiple kernels through the low-rank approximation of the corresponding kernel matrices, providing non-linear outputs and interpretation of kernel selection.

The motivation for the models and algorithms designed in this work stems from RNA biology and the rich complexity of protein-RNA interactions. Although the regulation of RNA fate happens at many levels - bringing in various possible data views - we show how different questions can be answered directly through constraints in the model design. We have developed an integrative orthogonality nonnegative matrix factorization (iONMF) to integrate multiple data sources and discover non-overlapping, class-specific RNA binding patterns of varying strengths. We show that the integration of multiple data sources improves the predictive accuracy of retrieval of RNA binding sites and report on a number of inferred protein-specific patterns, consistent with experimentally determined properties.

A principled way to extend the linear models to non-linear settings are kernel methods. Multiple kernel learning enables modelling with different data views, but are limited by the $O(n^2)$ computation and storage complexity of the kernel matrix. Con-

siderable savings in time and memory can be expected if kernel approximation and multiple kernel learning are performed simultaneously. We present the Mklaren algorithm, which achieves this goal via Incomplete Cholesky Decomposition, where the selection of basis functions is based on Least-angle regression, resulting in linear complexity both in the number of data points and kernels. Considerable savings in approximation rank are observed when compared to general kernel matrix decompositions and comparable to methods specialized to particular kernel function families. The principal advantages of Mklaren are independence of kernel function form, robust inducing point selection and the ability to use different kernels in different regions of both continuous and discrete input spaces, such as numeric vector spaces, strings or trees, providing a platform for bioinformatics.

In summary, we design novel models and algorithms based on matrix factorization and kernel learning, combining regression, insights into the domain of interest by identifying relevant patterns, kernels and inducing points, while scaling to millions of data points and data views.

Key words: Machine learning, bioinformatics, matrix factorization, kernel methods, multiple kernel learning, linear regression, protein-RNA interactions.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*
Martin Stražar
Faktorizacija matrik nizkega ranga pri učenju z večjedrnimi metodami

POVZETEK

V času pospešenega zbiranja, organiziranja in dostopnosti podatkov se pojavlja potreba po razvoju napovednih modelov na osnovi hkratnega učenja iz več podatkovnih virov. Konkretni primeri uporabe obsegajo področja strojnega učenja, priporočilnih sistemov, socialnih omrežij, financ in računske biologije. Heterogenost in velikost tipičnih podatkovnih zbirk vodi razvoj postopkov za hkratno zmanjšanje velikosti (zgoščevanje) in sklepanje iz več virov podatkov v skupnem modelu. Matrična faktorizacija in jedrne metode (ang. *kernel methods*) sta dve splošni orodji, ki omogočata dosego navedenega cilja. Pričujoče delo se osredotoča na naslednja specifična cilja: i) iskanje interpretabilnih, neprekrivajočih predstavitev vzorcev v podatkih s pomočjo ortogonalne matrične faktorizacije in ii) nadzorovano hkratno faktorizacijo več jedrnih matrik, ki omogoča modeliranje nelinearnih odzivov in interpretacijo pomembnosti različnih podatkovnih virov.

Motivacija za razvoj modelov in algoritmov v pričujočem delu izhaja iz RNA biologije in bogate kompleksnosti interakcij med proteini in RNA molekulami v celici. Čeprav se regulacija RNA dogaja na več različnih nivojih — kar vodi v več podatkovnih virov/pogledov — lahko veliko lastnosti regulacije odkrijemo s pomočjo omejitev v fazi modeliranja. V delu predstavimo postopek hkratne matrične faktorizacije z omejitvijo, da se posamezni vzorci v podatkih ne prekrivajo med seboj — so neodvisni oz. ortogonalni. V praksi to pomeni, da lahko odkrijemo različne, neprekrivajoče načine regulacije RNA s strani različnih proteinov. Z vzključitvijo več podatkovnih virov izboljšamo napovedno točnost pri napovedovanju potencialnih vezavnih mest posameznega RNA-vezavnega proteina. Vzorci, odkriti iz podatkov so primerljivi z eksperimentalno določenimi lastnostmi proteinov in obsegajo kratka zaporedja nukleotidov na RNA, kooperativno vezavo z drugimi proteini, RNA strukturnimi lastnostmi ter funkcijsko anotacijo.

Klasične metode matrične faktorizacije tipično temeljijo na linearnih modelih podatkov. Jedrne metode so eden od načinov za razširitev modelov matrične faktorizacije za modeliranje nelinearnih odzivov. Učenje z več jedri (ang. *Multiple kernel learning*) omogoča učenje iz več podatkovnih virov, a je omejeno s kvadratno računsko zahtevnostjo v odvisnosti od števila primerov v podatkih. To omejitev odpravimo z ustreznimi približki pri izračunu jedrnih matrik (ang. *kernel matrix*). V ta namen izboljšamo obstoječe metode na način, da hkrati izračunamo aproksimacijo jedrnih matrik ter njihovo linearno kombinacijo, ki modelira podan tarčni odziv. To dosežemo z metodo Mklaren (ang. *Multiple kernel learning based on Least-angle regression*), ki je sestavljena iz Nepopolnega razcepa Choleskega in Regresije najmanjših kotov (ang. *Least-angle regression*). Načrt algoritma vodi v linearno časovno in prostorsko odvisnost tako glede na število primerov v podatkih kot tudi glede na število jedrnih funkcij. Osnovne prednosti postopka so poleg računske odvisnosti tudi splošnost oz. neodvisnost od uporabljenih jedrnih funkcij. Tako lahko uporabimo različne, splošne jedrne funkcije za modeliranje različnih delov prostora vhodnih podatkov, ki so lahko zvezni ali diskretni, npr. vektorski prostori, prostori nizov znakov in drugih podatkovnih struktur, kar je prikladno za uporabo v bioinformatiki.

V delu tako razvijemo algoritme na osnovi hkratne matrične faktorizacije in jedrnih metod, obravnavnamo modele linearne in nelinearne regresije ter interpretacije podatkovne domene - odkrijemo pomembna jedra in primere podatkov, pri čemer je metode mogoče poganjati na milijonih podatkovnih primerov in virov.

Ključne besede: Strojno učenje, bioinformatika, matrična faktorizacije, jerdne metode, učenje z več jedrnimi funkcijami, linearna regresija, interakcije proteini-RNA.

ACKNOWLEDGEMENTS

This book is a small piece of puzzle in the vast scientific world. On the other hand, it constitutes a major part of what I have learned in the last couple of years, which has been possible with the help of numerous persons, to whom I express my sincere gratitude.

My thesis advisor dr. Tomaž Curk, University of Ljubljana, and dr. Jernej Ule, The Crick Institute, London, for introducing me to computational RNA biology, and providing critical feedback on a regular basis. To be part of a dynamic, international team and a fast-paced environment is the best a graduate student can wish for.

The head of Biolab at University of Ljubljana, dr. Blaž Zupan, for sharing important lessons in science and beyond, and being a great example of perseverance and resisting instances of higher power.

The coworkers at Biolab, fellow teachers and other staff members at the Faculty for a pleasant working and social environment.

To Tina and my family — Olga, Božo and Eva — for supporting me, which I often did not make any easier.

— Martin Stražar, Ljubljana, September 2018.

CONTENTS

I

Introduction

Data integration in machine learning refers to the exploitation of multiple data sources to improve model interpretability and performance. With the increased rate of data collection, storage, and availability, there exist a corresponding interest for data analyses and predictive models based on simultaneous inclusion of multiple data sources.

This tendency is ubiquitous in practical applications of machine learning; in recommender systems, side information collected on the customers along with their preferences and purchase history improves future recommendations [1]; similarly, social network analysis and prediction is improved by modelling explicit dependencies between users [2]; bioinformatics and precision medicine are often based on modelling a biological system behavior spanning multiple, inter-dependent regulatory levels, for example: gene expression, metabolic pathways, or macromolecule interactions [3, 4].

The heterogeneity and sheer size of the data sets in these domains must be addressed by tailored models and algorithms, providing scalable inference and interpretable decisions. The scalability problem can be solved with an appropriate dimensionality reduction scheme that results in model or data approximation. In a technical sense, the heterogeneity of data sets might present itself as features-of-features [5, 6], fixed relations between multiple object types [7], class labels [8], or multiple possible ways to measure object similarity, e.g. in kernel-based learning [9]. Hence, the two seemingly limiting properties within mentioned domains should be encoded within a single model.

Colloquially, the term *data integration* or *data fusion* has first arisen as the problem of combining different sensory information coming from one or more sensors [10, 11]. The methods forming this field can be broadly split into three major categories [12]; *early integration* comprises problems of merging the input data into a single data structure to be used with standard machine learning models. In *late integration*, an independent model is inferred for each data source, and the corresponding multiple predictions are combined at a later stage by a form of averaging. Both of the aforementioned strategies disregard the modular structure of the data. Finally, in *intermediate integration*, a single model is inferred by selecting information from multiple data sources, exploiting the information of the implicit structure assumed by multiple data sources.

This work focuses on two specific goals within intermediate data integration model, with the goal of finding:

1. non-overlapping data representations through the lens of different data sources

- *orthogonal matrix factorization*, and

2. complementary information within multiple similarity measures in order to model the target response - *multiple kernel regression*.

The proposed algorithms are rigorously evaluated using general benchmark, as well as domain specific data sets. Additionally, we provide extensive experiments on realistic data sets in the bioinformatics domain. The motivations, challenges and brief descriptions of the solutions are presented below.

1.1 Data integration by matrix factorization

A *data matrix* can be seen a relation between two types of objects encoded as corresponding rows and columns. Multiple data sources are thus represented as matrices relating different types of objects (entities). The principal assumption when matrix factorization is used in machine learning is that a data matrix is generated by a process depending on a small number of latent variables. Most commonly, this results in a large data matrix being approximated with two or more low-rank matrices which define the model [13]. Extensions to settings with multiple, dependent data matrices naturally follow [7, 14, 15].

The low-rank matrices are interpreted as projections to spaces of lower dimensionality, preserving the similarities between object in the original data space. Optimal projections are found by minimizing a divergence measure between the original data and its approximation [16–18]. Considering only the divergence criterion can lead to suboptimal results if the projections are later used for prediction [19]. Hence, improvements to the models focus on additional constraints such as sparseness, locality, margin and initialization methods [20–24].

Various improvements of the canonical NMF model have been suggested to obtain comprehensive models. Sparseness constraints can improve the interpretability and modularity of projections, and is achieved by including L_1 norm constraints on the model coefficients. Alternatively, the L_1/L_2 norm ratio of the resulting projection can be explicitly tuned [25]. Other methods constrain the basis vectors to convex sets [26, 27]. The mentioned methods, however, do not focus on modular decompositions where samples and features do not overlap within clusters. This is a substantial drawback when classes are discriminated by multiple patterns of varying strengths.

This phenomenon is common in the domain of protein-RNA interactions, as strong patterns common to many proteins may occlude weaker signals characteristic for specific proteins.

We design a novel matrix factorization model based on orthogonality in multiple data sources, to identify combinations of features that reflect cluster and class separation. For example, in modeling protein-RNA interactions, we are interested in discovering non-overlapping features in multiple data sources on sequence, function, conservation, structure and other genome annotation that describe the binding properties of a particular protein.

1.2 Low-rank approximation of multiple kernel matrices

The matrix factorization algorithms discussed so far are linear models for multi-variate, multi-output linear regression. Non-linear response can be modelled by transforming the input space which affects subsequent model inference. Such a transformation can be specified manually, e.g. by specifying higher order dependencies between the input features, or, more generally, with kernel functions [28]. Kernel methods define the covariance structure between the input data points, thus confining the space of allowed output functions (supervised learning) or probability densities (unsupervised learning) [29]. Kernel functions are inner products in Hilbert spaces, typically encoding higher-order feature expansions. Consequently, any algorithm that depends on the inner products between data points can be *kernelized* - replacing the inner products with the evaluation of the kernel.

The choice of a kernel function constrains the space of output functions in the sense of variance, smoothness, differentiability and more. If this choice is not known *a priori*, kernels can be combined due to the properties of sums of inner products. This is referred to as *multiple kernel learning (MKL)*; in the context of data integration, different data sources may be used to construct multiple kernel matrices encoding similarities between the same set of data points [9, 30]. The MKL algorithms are broadly categorized as fixed rules, risk minimization or optimization of a similarity measure with respect to the ideal kernel [31]. The solutions are often defined by a constrained optimization problem to learn a linear or convex combination of scalar kernel weights. This is solvable by off-the-shelf optimizers, which assume polynomial storage and time complexity in the number of data point or kernels.

A principal limitation of kernel methods is the computational complexity $O(n^2)$

associated with storage and evaluation of the kernel matrix — evaluation of the kernel between all pairs of n data points — and further increasing to $O(n^3)$ when solving the associated linear systems. Various kernel approximation schemes are thus designed to enable kernel learning with large data sets. The approximations are broadly divided in factorization of the kernel matrix — by using a small number of *inducing* inputs [32], or approximation of the kernel function — using a number of *basis functions* [33]. The approaches within the two paradigms are different in terms of computational complexity, and applicability to different types of both input spaces and kernels. Both sets of approaches avoid evaluating the full kernel matrix while simultaneously optimizing a downstream modeling task [34, 35].

In this work, we present simultaneous low-rank approximation of multiple kernel matrices. Contemporary matrix factorization methods do not consider non-overlapping low-dimensional projections of objects in context of side information (e.g., class labels or other circumstantial data sources). These prove particularly important when seeking efficient, low-dimensional projections for supervised learning. We present the Mklaren algorithm, based on supervised Incomplete Cholesky Decomposition to simultaneously learn multiple low-rank kernel approximations and a regression model. In regression, the inducing points define the *basis functions* spanning the space of possible output functions. Our basis function selection is based on a heuristic used in least-angle regression [36]. In comparison to existing methods, it has the following advantages.

- We show how sampling of basis functions from multiple kernels is not equivalent to approximation of an uniform kernel matrix sum. It is well-known that summing the kernel functions is equivalent to the concatenation of the respective implicit feature mappings in terms of solving for the optimal regression estimate [37]. However, approximating the uniform kernel matrix sum with inducing point-based methods causes the basis function to include all kernels in equal proportions. Thus, some included kernels may not be relevant to the targets, or even causing unwanted distortions in the output functions. This limitation motivates more careful basis function selection.

- Our selection criterion only considers the gain with respect to the current regression residual without considering approximation accuracy of the original kernel matrices. Increasing the accuracy of the kernel matrix approximation

causes the regression estimates to be increasingly more similar to the estimates obtained with the full kernel matrix [38]. However, the expected generalization error is largely affected by the alignment of the regression targets and the low-dimensional space spanned by the kernel matrix approximation [39]. Thus, constructing the kernel matrix approximation in a supervised manner promises a rapid drop in generalization error.

- The importance of a kernel is estimated at the time of its approximation, without assuming knowledge of the full kernel matrix. Further benefits are memory efficiency — irrelevant kernels are discarded early — and data interpretation, based on selected inducing points and kernels.

Further technical derivations are provided related to out-of-sample prediction, regularization and interpretability. In contrast to MKL algorithms based on convex optimization or sampling methods, our approach relies solely on geometrical principles, enabling efficient implementation with proven linear complexity in both the number of data points and kernels.

Even though the currently most efficient kernel approximations are based on optimization of the kernel function, our approach favours general applicability and is independent of a particular kernels and types of input spaces, which could also be arbitrarily combined. Nevertheless, the performance on benchmark data sets is statistically indistinguishable. As the majority of kernel matrix approximations assume a single kernel, we show how the notion of multiple kernels within low-rank approximation can lead to better compression and more flexible output function spaces. Namely, the covariance structure can vary between different regions of the input space. This capability proves beneficial for a wide range of realistic regression problems and provides insights into the domain of interest.

1.3 Summary of the scientific contributions

Finally, we summarize the scientific contributions proposed in this work, with references to the relevant sections of the thesis.

C1 Integrative orthogonal matrix factorization (iONMF, Chapters 3-4):

- formal definition of the orthogonal matrix factorization model on multiple data sources,

- derivation and mathematical analysis of the optimization algorithm,
- inference of latent factors for unseen instances given a subset of data sources (prediction function),
- analysis of a RNA-protein interaction data set with iONMF, discovering novel patterns characterizing RNA-RBP interactions.

C2 A general approach to approximate multiple kernel learning (Chapters 6-7):

- algorithm for supervised low-rank matrix approximations of multiple kernel matrices based on least-angle regression (Mklaren),
- functional analysis of multiple kernel learning with low-rank approximations,
- methods for interpretability of approximate multiple kernel regression model for vector and string input data,
- kernel approximation and multiple kernel learning library for Python programming language.

1.4 Availability

We provide the following open source software packages, providing the implementation of the proposed methods in the Python programming language, as well as scripts to reproduce the included experiments.

- Integrative orthogonal non-negative matrix factorization
  `https://github.com/mstrazar/iONMF`
- A Multiple kernel learning Python library
  `https://github.com/mstrazar/mklaren`

1.5 Overview of thesis structure

On a general level, the thesis is split in two major parts, describing different, but related views on learning with multiple data sources: the first part treats matrix factorization (Chapters 2-4; linear data models) and the second part extends ideas to kernel matrix factorization (Chapters 5-7; non-linear models). The general trajectory is the design of models and algorithms that perform dimensionality reduction in context of multiple data sources. Optionally, the reader can start with a short introduction to the biological domain, which provides context and motivation to many of the proposed methods (Appendix A).

The first part starts by introducing matrix factorization techniques, which mainly differ in constraints to the optimization problems. Here, we extend the state in the field by designing a model for treating multiple data sources, focusing on interpretability and discovery of multiple, non-overlapping patterns in the data. The experiments are based on modeling interactions between proteins and RNA (supervised machine learning problem), but nevertheless focusing on modeling and computational aspects of the work. This chapter can be read wearing a computational biologist hat, a machine learning hat, or both.

In the second part, the ideas of learning with multiple data sources are extended to non-linear output functions via kernel methods. After presenting the basic concepts, we present the related work in approximate kernel learning. We then present the main contribution of the work — the Mklaren algorithm — which selectively approximates multiple kernel matrices, that might present different data views. A thorough experimental evaluation compares our method to state-of-the-art kernel approximations and multiple kernel learning algorithms on different kinds of input spaces.

A graphical representation of the dependencies between the chapters is shown in Figure 1.1. The used mathematical notation is listed in Table B.1, p. 121.

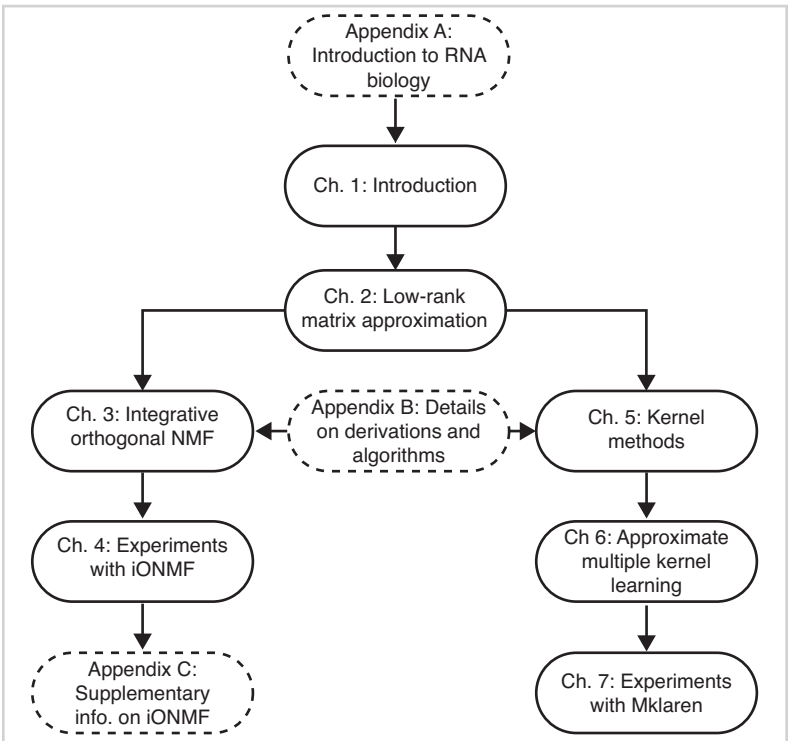


Figure 1.1

A roadmap of the thesis. Chapters are displayed as nodes and dependencies are denoted as arrows. The appendix chapters containing additional information and/or further technical details are marked with dashed borders.

2

Low-rank matrix approximation

Matrix approximation is a core task in numerical mathematics and linear algebra. In machine learning, it plays an essential role in enabling tractable computation for problems with large datasets. In this section, we will discuss various matrix approximation algorithms underpinning this work, with particular focus on the interpretation of solutions.

The general problem is stated as follows. Given a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, find the matrices $\mathbf{A} \in \mathbb{R}^{n \times r}$ and $\mathbf{B} \in \mathbb{R}^{d \times r}$, where $r \leq \min(n, d)$, such that:

$$\mathbf{X} \approx \mathbf{A}\mathbf{B}^T.$$

The notion of approximation as well as additional constraints on $\mathbf{A}$ and $\mathbf{B}$ depend on the task at hand. Often, $\mathbf{X}$ is a data matrix encoding a numerical relation between two sets of objects, represented as rows and columns. The rank r is a typically much smaller than n and d. Optimization constraints are posed on $\mathbf{A}$ and $\mathbf{B}$ to preserve certain properties of $\mathbf{X}$, enabling machine learning with substantial savings in computational resources.

2.1 Notions of error

It makes sense to first define what it means for $\mathbf{X}$ to be approximately equal to the product $\mathbf{A}\mathbf{B}^T$. The error (loss) functions are used as explicit optimization objectives and provide a blueprint on which the solving algorithms are based.

Sum of squared errors / explained variance. The sum of squared errors is the variance of errors in corresponding elements of $\mathbf{X}$ and $\mathbf{A}\mathbf{B}^T$, also known as matrix Frobenius norm. The underlying assumption is that each value in $\mathbf{X}$ should be approximated as accurately as possible. This error function is most often used due to its favourable optimization properties (to be discussed further):

$$\begin{aligned} \text{var}(\mathbf{X} - \mathbf{A}\mathbf{B}^T) &= \|\mathbf{X} - \mathbf{A}\mathbf{B}^T\|_F^2 \\ &= \sum_{i=1}^{n} \sum_{j=1}^{d} (x_{ij} - \langle \mathbf{a}_i, \mathbf{b}_j \rangle)^2 \\ &= \sum_{i=1}^{n} \sum_{j=1}^{d} (x_{ij} - \mathbf{a}_i^T \mathbf{b}_j)^2. \end{aligned} \tag{2.1}$$

The value of the error can be made interpretable if written as a fraction of explained variance, a simple ratio of remaining and initial variances,

$$\text{explained variance} = \frac{\text{var}(\mathbf{X}) - \text{var}(\mathbf{X} - \mathbf{A}\mathbf{B}^T)}{\text{var}(\mathbf{X})},$$

with values in $(-\infty, 1)$ where the value 0 is natural threshold above which $\mathbf{A}$ and $\mathbf{B}$ contain meaningful information on $\mathbf{X}$. A traditional way to solve the unconstrained, explained variance optimization for low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ is the Principal component analysis (PCA, Appendix B.1.2). Alternative optimization objectives and the corresponding algorithms are reviewed in the Appendix B.1.

2.2 *Non-negative matrix factorization*

Non-negative matrix factorization (NMF) is a special case where $\mathbf{X}$, $\mathbf{A}$ and $\mathbf{B}$ are constrained to be non-negative element-wise, i.e. $x_{ij} \geq 0$ for all values in $\mathbf{X}$ and similarly for $\mathbf{A}$, $\mathbf{B}$. The modelling assumption encoded by the constraint is that data is expressed as a *sum of parts*, as values in $\mathbf{X}$ are approximated only with addition [13]. The underlying motivation stems from practical applications; if r is a predefined rank of the solution, then $\mathbf{A}$ and $\mathbf{B}$ contain r prototype rows and columns, respectively. The solution thus encodes commonly occurring patterns present in $\mathbf{X}$. The non-negativity constraint appears to provide interpretable solutions, as the summing the parts suit human interpretation. This is of particular interest in signal processing, image decomposition and other pattern recognition tasks.

Optimizing explained variance. The solution optimized with respect to the explained variance can be obtained via gradient descent. In order to do so, partial derivatives of all the parameters in $\mathbf{A}, \mathbf{B}$ with respect to the cost function are required:

$$\begin{aligned} J &= ||\mathbf{X} - \mathbf{A}\mathbf{B}^T||_F^2 \\ \frac{\delta J}{\delta a_{it}} &= 2\sum_{j=1}^{d}(x_{ij} - \mathbf{a}_i^T\mathbf{b}_j)(-b_{jt}), \\ \frac{\delta J}{\delta b_{jt}} &= 2\sum_{i=1}^{n}(x_{ij} - \mathbf{a}_i^T\mathbf{b}_j)(-a_{it}), \end{aligned}$$

or in matrix form:

$$\frac{\delta J}{\delta \mathbf{A}} = -2(\mathbf{X} - \mathbf{A}\mathbf{B}^T)\mathbf{B},$$
$$\frac{\delta J}{\delta \mathbf{B}} = -2(\mathbf{X} - \mathbf{A}\mathbf{B}^T)^T\mathbf{A}.$$

A gradient descent-type algorithm would update all of the parameters multiple times by moving in the direction of derivatives using a fixed learning rate η, e.g.:

$$\mathbf{A}^{\text{new}} = \mathbf{A} - \eta\frac{\delta J}{\delta \mathbf{A}},$$
$$\mathbf{B}^{\text{new}} = \mathbf{B} - \eta\frac{\delta J}{\delta \mathbf{B}}.$$

The element-wise non-negativity constraint must be guaranteed manually in this case. A suitable option is the projected gradient descent, where parameters in $\mathbf{A}$ or $\mathbf{B}$ less than zero are set to zero in each iteration.

Another option are the multiplicative update rules, obtained by allowing learning rates $\boldsymbol{\eta}_A \in \mathbb{R}^{n\times r}$ and $\boldsymbol{\eta}_B \in \mathbb{R}^{d\times r}$, which are specific to each element in $\mathbf{A}$, $\mathbf{B}$ and are variable between iterations. The update rules are derived from the gradient descent updates:

$$\mathbf{A} = \mathbf{A} - \boldsymbol{\eta}_A(\mathbf{X}\mathbf{B} - \mathbf{A}\mathbf{B}^T\mathbf{B}),$$

$$\mathbf{B} = \mathbf{B} - \boldsymbol{\eta}_B(\mathbf{X}^T\mathbf{A} - \mathbf{B}\mathbf{A}^T\mathbf{A}),$$

if the learning rates $\boldsymbol{\eta}_A$ and $\boldsymbol{\eta}_B$ are set

$$\boldsymbol{\eta}_A = \frac{\mathbf{A}}{\mathbf{A}\mathbf{B}^T\mathbf{B}},$$

$$\boldsymbol{\eta}_B = \frac{\mathbf{B}}{\mathbf{B}\mathbf{A}^T\mathbf{A}},$$

we obtain the update rules

$$\mathbf{A}^{\text{new}} = \mathbf{A} \circ \frac{\mathbf{XB}}{\mathbf{AB}^T\mathbf{B}}, \qquad \mathbf{B}^{\text{new}} = \mathbf{B} \circ \frac{\mathbf{X}^T\mathbf{A}}{\mathbf{BA}^T\mathbf{A}}, \tag{2.2}$$

where $\circ$ and $\frac{\cdot}{\cdot}$ denote the element-wise (Hadamard) product and element-wise division, respectively. As the multiplicative updates effectively perform gradient descent on each element individually (although with different learning rates), the same convergence properties of gradient descent apply.

Note that if $\mathbf{A}$ and $\mathbf{B}$ are initialized to have all strictly positive values, division by zero is avoided by definition. In practice, the limited machine precision can cause numerical instabilities in evaluating Eq. 2.2. This can happen if the denominators are either too small, or too large. The recommended solutions are to add a small amount of noise ϵ to the parameters, $\mathbf{A} + \epsilon$, $\mathbf{B} + \epsilon$ [40]. Alternatively, one can (Hadamard) multiply the parameters with a binary *mask* matrices $\mathbf{M}_A$, $\mathbf{M}_B$ indicating the valid elements in $\mathbf{A}$ and $\mathbf{B}$, respectively. In the latter solution, whenever a parameter reaches an invalid value (i.e. a *NaN*), it is set to zero for the remaining iterations. This strategy is somewhat similar to NMF with missing values [41].

2.3 Constrained matrix factorization

The NMF models presented above typically approximate the data up to error, which typically decreases with increasing rank r. However, there exist a substantial possibility that the found patterns are due to chance and give deceivingly low approximation errors due to high model capacity. This phenomenon is colloquially known as *overfitting* and arises in many other contexts as a consequence of the known bias-variance tradeoff [42]. A typical antidote is to include different constraints in the optimization of model parameters (regularization, preconditioning) or include explicit prior assumptions about the model parameters (prior distributions). Below, we present some common types of constraints that are added to error functions presented in Section 2.1.

Matrix norm-based constraints. An increase in the bias of model approximation is achieved by including L_1 or L_2 norm constrains on the vectors of model parameters. The vector L_2 norm corresponding to matrices is the Frobenius norm (Eq. 2.1).

Sparseness. A measure of sparseness can also be constrained in attempt to find interpretable solutions. Hoyer [25] defines sparseness as a function of the ratio between the L_1 (sum of the absolute values) and the L_2 norm (the Euclidean norm). Thus, for a vector $\mathbf{x}$, the sparseness value is defined as

$$\text{sparseness}(\mathbf{x}) = \frac{\sqrt{n} - \frac{||\mathbf{x}||_1}{||\mathbf{x}||_2}}{\sqrt{n} - 1}. \tag{2.3}$$

Sparseness is equal to 0 if all elements are equal up to signs and increases towards 1 as increasingly small subsets of values take up significantly high values compared to the remaining elements. Note that sparseness can be applied to a matrix if the latter is treated as a vector - the L_2 vector norm is replaced by the matrix Frobenius norm. The measure of sparseness is used in the Sparse NMF (SNMF) model and algorithms.

Orthogonality. In certain cases, presented in later chapters, the vectors of $\mathbf{A}$ that make up the model can be constrained to be orthogonal (non-overlapping). This is useful in practice if non-overlapping patterns in the data are expected. By definition, the matrix $\mathbf{A}$ is orthonormal if $\mathbf{A}^T\mathbf{A} = \mathbf{I}$. With the algorithms to compute the matrix QR decomposition or the SVD, this hard constraint is satisfied by iterative construction of the subspace [43]. These methods however are not constrained to non-negative solutions. A matrix decomposition-based approach of non-negative PCA can be employed to solve to orthogonal NMF problem for one matrix [44].

Alternatively, a measure of how close a matrix $\mathbf{A}$ is to an orthogonal matrix can also be defined (up to magnitude) as the distance to an identity matrix:

$$\text{orthogonality}(\mathbf{A}) = ||\mathbf{A}^T\mathbf{A} - \mathbf{I}||_F. \tag{2.4}$$

It is important to note that this definition is clearly affected by the scale of $\mathbf{A}$. However, in our application, we find this definition appropriate since it has favourable optimization properties for gradient-based optimization (smoothness, differentiability) and the magnitude of $\mathbf{A}$ can be compensated for by other matrices in the model.

In the subsequent chapter, we show that sparseness can be achieved with a combination of orthogonality and non-negativity constraints. This stems from the fact that if the relevant vectors are non-nonnegative, orthogonality implies that at least $n/2$ values must equal zero for two vectors of size n to be orthogonal, with the number decreasing accordingly with the increasing number of columns of $\mathbf{A}$.

2.4 Simultaneous matrix factorization

Factorization of a single matrix can be extended to multiple matrices. Multiple matrices may describe a set of objects with different data views, or data sources. Objects might be represented as rows of p matrices $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_p$ of respective sizes $n \times d_1$, $n \times d_2, ...n \times d_p$. The goal is then to find matrices $\mathbf{A}$ and $\mathbf{B}_1, \mathbf{B}_2, ..., \mathbf{B}_p$, each with r columns, such that:

$$\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_p \approx \mathbf{A}\mathbf{B}_1^T, \mathbf{A}\mathbf{B}_2^T, ..., \mathbf{A}\mathbf{B}_p^T.$$

Here, $\mathbf{A}$ represents a set of parameters that are shared between data views and matrices $\mathbf{B}_I$ are data-type specific loadings. In practice, $\mathbf{A}$ can be interpreted as a (soft) clustering of objects, while the $\mathbf{B}_I$ can be seen as typical patterns (profiles) of a cluster in each of the data views. The non-negativity constraint is again favoured to interpret the data as a sum of non-negative parts.

With no additional constraints, this problem is equivalent to the (non-negative) matrix factorization problems stated above if $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_p$ are concatenated into a single matrix. Constraints that make this setting different to single matrix factorization are discussed further in the forthcoming sections and present the basis of the proposed integrative, orthogonal NMF.

3

Integrative orthogonal nonnegative matrix factorization

In this part, we present an algorithm based on orthogonal decomposition of multiple data sources/matrices; our model is referred to as Integrative orthogonal nonnegative matrix factorization (iONMF) . This is achieved with gradient-based optimization by a joint minimization of the distance of a) data and its approximations, and b) the data-source specific parameter matrices to an orthogonal matrix. A principal contribution of this part of the work is the exploration of how the design of the low-rank matrix approximation algorithm influences the performance and interpretability of the resulting models.

3.1 The iONMF model and algorithms

Data sources are represented as matrices $\mathbf{X}_q$, $q = 1, ..., p$, with n rows representing data points and d_q the dimensionality of each data source. The d_q dimensions in each data source are hereby referred to as *features*. Non-negative matrix factorization (NMF) approximates each $\mathbf{X}_q \in \mathbb{R}^{n \times d_q}$ with the following parameters, making up a factor model: a product of a common coefficient matrix $\mathbf{W} \in \mathbb{R}^{n \times r}$ and data source-specific basis matrices $\mathbf{H}_q \in \mathbb{R}^{d_q \times r}$, where the rank $r < \min(n, \sum_q d_q)$. This scenario is depicted on Fig. 3.1a-b.

The model parameters are interpreted as follows. Each sample is projected to r latent factors, which is reflected by the coefficients $\mathbf{W}$. The features of each data source $\mathbf{H}_q$ are projected to the same r latent factors. The projection to the r latent factors can be interpreted as soft-clustering, resembling the well-known k-means clustering algorithm [45], with the crucial difference of a sample being assigned to multiple clusters. For brevity, we refer to the r latent factors and the corresponding assigned samples and features as *modules*. This model assumes that highly correlated samples and features are assigned to common modules, depending on their similarity within all data sources $\mathbf{X}_q$. Hence, a set of highly correlated features will emerge as *patterns* in the corresponding vectors in $\mathbf{H}_q$. One can identify both the correlated features within a single data source, as well as correlated patterns between different data sources.

The patterns can be of varying magnitudes (in the sense of the vector norm). If the optimization is based on the error-norm, as with the explained variance / Frobenius norm (Chapter 2, Eq. 2.1) the patterns of larger magnitudes will be preferentially identified, occluding the remaining patterns of smaller magnitudes. One goal of a pattern discovery model is to identify a number of patterns that both maximize the explained variance as well as having little or no overlap among themselves. The phenomenon

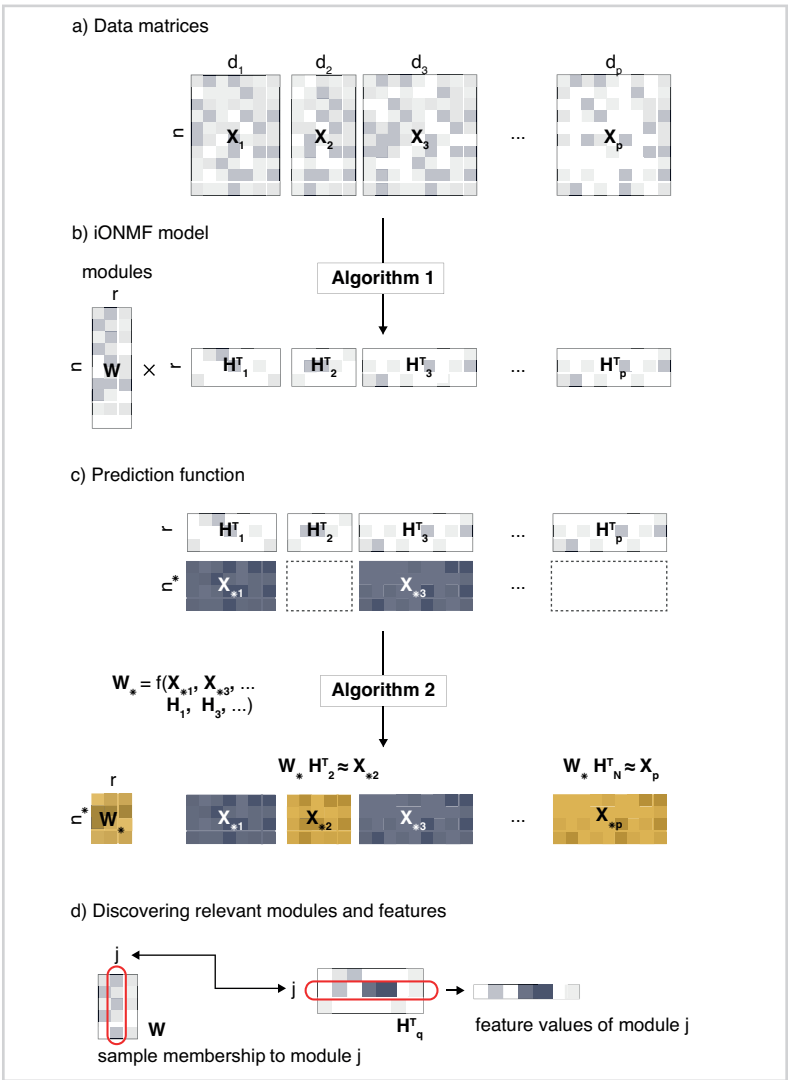


Figure 3.1

Graphical representation of the iONMF model. a) The data matrices $\mathbf{X}_1, \mathbf{X}_2, ...\mathbf{X}_p$ are decomposed by orthogonal, non-negative matrix factorization (Algorithm 1). b) The iONMF model is composed of the p approximately orthogonal basis matrices $\mathbf{H}_1, \mathbf{H}_2, ...\mathbf{H}_p$ and a common coefficient matrix $\mathbf{W}$. c) The prediction function (Algorithm 2) is used to estimate the coefficient matrix $\mathbf{W}_*$ for an arbitrary number of test samples, given a subset of the data sources. A potentially unknown data matrix $\mathbf{X}_q$ for test samples can then be predicted by $\mathbf{X}_{*q} = \mathbf{W}_* \mathbf{H}_q^T$. Given test data is shown in blue and the predicted variables are shown in orange. d) Discovering relevant features for different modules in the data. Samples are assigned to modules based on rows in $\mathbf{W}$. Row j in $\mathbf{H}_q^T$ describes the common patterns of each module (j).

where a number of patterns of varying magnitudes appear in the data in a practical domain is discussed in the Chapter 4, Section 4.8.

Non-overlapping features relevant to each module are obtained by imposing orthogonality on the vectors in $\mathbf{H}$ (Eq. 2.4). The iONMF algorithm implements orthogonality regularization in the following cost function, given the data matrices $\mathbf{X}_q$:

$$J(\mathbf{W},\mathbf{H}_q) = \sum_{q=1}^{p}(||\mathbf{X}_q - \mathbf{W}\mathbf{H}_q^T||_F^2 + \alpha||\mathbf{H}_q^T\mathbf{H}_q - \mathbf{I}||_F^2), \tag{3.1}$$

subject to $\mathbf{W},\mathbf{H}_q \geq \mathbf{0}$, with $\geq$ referring to element-wise inequality and $\mathbf{I}$ the identity matrix. The first term represents the approximation error and second term the orthogonality regularizer of column vectors in $\mathbf{H}_q$, where the trade-off is controlled by the hyperparameter α. Note that the optimization problem would be equivalent if the matrices $\mathbf{X}_q$ are concatenated to a single matrix had the orthogonality constraints not been included. The orthogonality between the vectors is thus limited to a single data source, so the parameters of the model are dependent on the assignment of features to data sources. The orthogonality constraint introduces an additional property related to limiting the model capacity; namely, as the vectors in $\mathbf{H}$ are strictly non-negative, the orthogonality constraints will force a number of terms to zero. This could be explained intuitively as two non-zero, non-negative vectors are orthogonal if and only if a non-zero value in one vector implies a corresponding value in the second vector to be zero. Additionally, assuming the same parameter α for all data sources, lengths of vectors in $\mathbf{H}_q$ will tend to one regardless of size.

The optimization problem is non-convex and can be solved by projected gradient descent, alternating non-negative least squares [46], multiplicative update rules [47] or second order gradient methods [48]. We propose a multiplicative update-based algorithm (Algorithm 1), which is an instance of gradient descent with variable learning rate and implicitly constraints the parameters to non-negative values. The optimization algorithm samples the initial values of $\mathbf{W}$ and $\mathbf{H}_q$ uniformly from $(0,1)$, and updates them with the following rules until convergence:

$$\mathbf{W} = \mathbf{W} \circ \sqrt{\frac{\sum_q \mathbf{X}_q\mathbf{H}_q}{\sum_q \mathbf{W}\mathbf{H}_q^T\mathbf{H}_q}}, \tag{3.2}$$

$$\mathbf{H}_q = \mathbf{H}_q \circ \sqrt{\frac{\mathbf{X}_q^T\mathbf{W} + \alpha\mathbf{H}_q}{\mathbf{H}_q\mathbf{W}^T\mathbf{W} + 2\alpha\mathbf{H}_q\mathbf{H}_q^T\mathbf{H}_q}}, \tag{3.3}$$

where $\circ$ represents the element-wise (Hadamard) product. The derivation of the update rules is presented in Section 3.2, below.

A special case arises when one or more data matrices consist of a single column — a common example where the data samples are related to regression targets, referred to as $\mathbf{Y}$. In this case, the corresponding orthogonality constraints are omitted since $\mathbf{H}_Y$ consists only of a single column. The stopping criterion can be set by e.g. thresholding the fraction of change in the cost function or explained variance in subsequent iterations. Further discussion on the choice of algorithm, derivation of update rules, relation to gradient descent are shown in Section 3.2, below.

In practice, due to non-convexity, the algorithm is run for multiple random initializations and the model with the lowest approximation error is selected. Alternatively, one could use an independent validation set for model selection. The numerically unstable evaluations of the denominators in Eqs. 3.2-3.3 are alleviated by masking out the invalid values, as outlined in Section 2.2.

3.2 *Derivation of the iONMF optimization algorithm*

In this section, we present derivations that are part of the algorithm to find the parameters of the iONMF model (Algorithm 1-2).

To learn the parameters of the iONMF model, we solve the following constrained minimization problem with respect to $\mathbf{W}$ and $\mathbf{H}_q$ for $q = 1, ..., p$:

$$J = \sum_{q=1}^{p} (||\mathbf{X}_q - \mathbf{W}\mathbf{H}_q^T||_F^2 + \alpha||\mathbf{H}_q^T\mathbf{H}_q - \mathbf{I}||_F^2).$$

The parameter α determines the trade-off between explained variance (the data fit term) and orthogonality of vectors in $\mathbf{H}_q$ (model capacity term). Since the optimization problem is non-convex in all $\mathbf{W}$, $\mathbf{H}_q$, a local minimum can be found by fixing all but one matrix and applying multiplicative update rules. The cost function can be rewritten as

Algorithm 1: The iONMF model inference algorithm pseudocode.

Input:

$\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_p$ set of $\mathbb{R}^{n \times d_q}$ matrices,
$\mathbf{Y} \in \mathbb{R}^{n \times 1}$ target matrix,
r factorization rank
α orthogonality regularization parameter.

Result:

$\mathbf{W} \in \mathbb{R}^{n \times r}$ coefficient matrix,
$\mathbf{H}_1, \mathbf{H}_2, ..., \mathbf{H}_p$ set of $\mathbb{R}^{d_q \times r}$ basis matrices,
$\mathbf{H}_Y \in \mathbb{R}^{n \times 1}$ target basis matrix.

Initialize:
  $\mathbf{W} \sim \mathscr{U}(0,1)^{n \times r}$
  $\mathbf{H}_q \sim \mathscr{U}(0,1)^{d_q \times r}$ (for each q)
  $\mathbf{H}_Y \sim \mathscr{U}(0,1)^{r \times 1}$

while not converged do
  $\mathbf{W} = \mathbf{W} \circ \sqrt{\frac{\sum_q \mathbf{X}_q \mathbf{H}_q + \mathbf{Y}\mathbf{H}_Y}{\sum_q \mathbf{W}\mathbf{H}_q^T \mathbf{H}_q + \mathbf{W}\mathbf{H}_Y^T \mathbf{H}_Y}}$
  $\mathbf{H}_q = \mathbf{H}_q \circ \sqrt{\frac{\mathbf{X}_q^T \mathbf{W} + \alpha \mathbf{H}_q}{\mathbf{H}_q \mathbf{W}^T \mathbf{W} + 2\alpha \mathbf{H}_q \mathbf{H}_q^T \mathbf{H}_q}}$ (for each q)
  $\mathbf{H}_Y = \mathbf{H}_Y \circ \sqrt{\frac{\mathbf{Y}^T \mathbf{W}}{\mathbf{H}_Y \mathbf{W}^T \mathbf{W}}}$

$$J = \sum_{q=1}^{p} \text{tr}(\mathbf{X}_q^T\mathbf{X}_q - 2\mathbf{X}_q^T\mathbf{W}\mathbf{H}_q + \mathbf{H}_q\mathbf{W}^T\mathbf{W}\mathbf{H}_q^T)$$
$$+ \alpha\ \text{tr}(\mathbf{H}_q^T\mathbf{H}_q\mathbf{H}_q^T\mathbf{H}_q - 2\mathbf{H}_q^T\mathbf{H}_q + \mathbf{I}^T\mathbf{I}).$$

Following standard theory of constrained multivariate optimization [49], the Lagrangian equals

$$L(\mathbf{W}, \mathbf{H}_1, ..., \mathbf{H}_p, \lambda_0, \lambda_1, ..., \lambda_p) = J - \text{tr}(\lambda_0\mathbf{W}) - \sum_{q=1}^{p} tr(\lambda_q\mathbf{H}_q),$$

where $\lambda_0, \lambda_1, ..., \lambda_p$ denote the slack variables. By fixing all the $\mathbf{H}_q$, the derivative of the Lagrangian with respect to $\mathbf{W}$ is

$$\frac{\delta L}{\delta \mathbf{W}} = \sum_{q=1}^{p} -2\mathbf{X}_q\mathbf{H}_q^T + 2\mathbf{W}\mathbf{H}_q^T\mathbf{H}_q - \lambda_0.$$

To satisfy the Karush-Kuhn-Tucker optimality conditions at a stationary point, it must be that case that

$$\mathbf{W} \circ \lambda_0 = \mathbf{0}.$$

Writing $\lambda_0 = (\lambda_0^+ - \lambda_0^-)$ we have:

$$\mathbf{W}^2 \circ (\lambda_0^+ - \lambda_0^-) = \mathbf{0}. \tag{3.4}$$

The exponent in $\mathbf{W}^2$ is assumed to act element-wise and is used to specify the relation between parameters $\mathbf{W}$ in two subsequent iterations. The operator $\mathbf{A}^+$ on matrix (or scalar) $\mathbf{A}$ retains only positive elements of $\mathbf{A}$ and replaces negative elements with zeros. The operator $\mathbf{A}^-$ retains the absolute values of the negative elements in $\mathbf{A}$ and places zeros everywhere else. The Eq. 3.4 is a fixed point equation, which can be solved by iteratively applying the update rule

$$\mathbf{W} = \mathbf{W} \circ \sqrt{\frac{\lambda_0^-}{\lambda_0^+}} = \mathbf{W} \circ \sqrt{\frac{\sum_{q=1}^{p}(\mathbf{X}_q\mathbf{H}_q^T)^+ + (\mathbf{W}\mathbf{H}_q^T\mathbf{H}_q)^-}{\sum_{q=1}^{p}(\mathbf{X}_q\mathbf{H}_q^T)^- + (\mathbf{W}\mathbf{H}_q^T\mathbf{H}_q)^+}}.$$

Since $\mathbf{X}_q, \mathbf{H}_q, \mathbf{W}$ are non-negative for all $q = 1, ..., p$, the update rule equals:

$$\mathbf{W} = \mathbf{W} \circ \sqrt{\frac{\sum_{q=1}^{p}(\mathbf{X}_q\mathbf{H}_q^T)^+}{\sum_{q=1}^{p}(\mathbf{W}\mathbf{H}_q^T\mathbf{H}_q)^+}},$$

which is the update rule given in Eq. 3.2. All matrices vanish to $\mathbf{0}$ after applying the operator $^-$ following the strictly positive initialization defined in Algorithm 1. Following a similar argument, the update rules for coefficient matrices $\mathbf{H}_q$ can be derived. Fixing $\mathbf{W}$ and all $\mathbf{H}_j$, $j \neq q$, the derivative of the Lagrangian with respect to $\mathbf{H}_q$ is equal to:

$$\frac{\delta L}{\delta \mathbf{H}_q} = -2\mathbf{X}_q^T\mathbf{W} + 2\mathbf{H}_q\mathbf{W}^T\mathbf{W} + \alpha(4\mathbf{H}_q\mathbf{H}_q^T\mathbf{H}_q - 2\mathbf{H}_q) - \lambda_q = 0,$$

so that

$$\lambda_q = -\mathbf{X}_q^T\mathbf{W} + \mathbf{H}_q\mathbf{W}^T\mathbf{W} + \alpha(2\mathbf{H}_q\mathbf{H}_q^T\mathbf{H}_q - \mathbf{H}_q).$$

To satisfy the Karush-Kuhn-Tucker optimality conditions at a stationary point we must have:

$$\mathbf{H}_q \circ \lambda_q = \mathbf{0},$$

$$\mathbf{H}_q^2 \circ (\lambda_q^+ - \lambda_q^-) = \mathbf{0},$$

which leads to the following update rules:

$$\mathbf{H}_q = \mathbf{H}_q \circ \sqrt{\frac{\lambda_0^-}{\lambda_0^+}} =$$

$$= \mathbf{H}_q \circ \sqrt{\frac{(\mathbf{H}_q\mathbf{W}^T\mathbf{W})^- + 2\alpha(\mathbf{H}_q\mathbf{H}_q^T\mathbf{H}_q)^- + (\mathbf{X}_q^T\mathbf{W})^+ + \alpha(\mathbf{H}_q)^+}{(\mathbf{H}_q\mathbf{W}^T\mathbf{W})^+ + 2\alpha(\mathbf{H}_q\mathbf{H}_q^T\mathbf{H}_q)^+ + (\mathbf{X}_q^T\mathbf{W})^- + \alpha(\mathbf{H}_q)^-}},$$

$$\mathbf{H}_q = \mathbf{H}_q \circ \sqrt{\frac{(\mathbf{X}_q^T\mathbf{W})^+ + \alpha(\mathbf{H}_q)^+}{(\mathbf{H}_q\mathbf{W}^T\mathbf{W})^+ + 2\alpha(\mathbf{H}_q\mathbf{H}_q^T\mathbf{H}_q)^+}}.$$

Again, this is exactly the update rule in Eq. 3.3 (Section 3.1).

■

3.3 The prediction function

A common assumption when applying NMF for prediction is that all objects in the domain, including the test samples, are available in the learning phase [50]. Cold-start approaches [51] or regression on the obtained factors [52] can be used to predict values for test samples. Alternatively, non-negative least-squares optimization is used to approximate the coefficient matrix values from available matrices describing new samples [7].

We reuse the inferred low-rank matrices to predict the values for any number of test samples, given the values of at least one data source. This is achieved by projecting new data to existing latent factors, using the available data sources. Algorithm 2 is a special case of Algorithm 1; given fixed basis matrices $\mathbf{H}_q$, and n_* test data points with a subset of known $\mathbf{X}_{*q}$, we use the update rule 3.2 to first solve for $\mathbf{W}_*$ and then predict using

$$\mathbf{X}_{*q} = \mathbf{W}_* \mathbf{H}_q^T.$$

Note that this approach preserves non-negativity of $\mathbf{W}_*$ and can be used even if only a subset of data sources is available for test data. The scenario is presented in Fig. 3.1c.

3.4 Discovering relevant modules and features

The obtained coefficient matrix $\mathbf{W}$ is used to assign data samples (in rows) to specific modules (in columns). The values of $\mathbf{W}$ are determined based on all $\mathbf{X}_q$ and define the modules, while individual $\mathbf{H}_q$ are determined based only on the corresponding data sources $\mathbf{X}_q$.

Proposed methods include assigning the sample to the module with maximum row value or restricting the assignment to only one module [53]. Alternatively, the ability to assign samples to multiple modules may be desired. One such approach, developed by Zhang et al. [14], converts each entry in the coefficient matrix to the corresponding column-wise z-score. Samples are assigned to modules where the corresponding z-score exceeds a predefined threshold. For each of the $j = 1...r$ modules, we obtain a count C_j of how many positive samples (e.g. according to $\mathbf{Y}$) are related to the module j.

Our approach is depicted on Fig. 3.1d. The modules are sorted on descending value of C_j and the corresponding (column) vectors of matrices $\mathbf{H}_q$ were then examined to discover the relevant features of each data source.

Algorithm 2: The iONMF prediction algorithm pseudocode.

Input:

$\mathbf{X}_q \subset \{\mathbf{X}_{*1}, \mathbf{X}_{*2}, ..., \mathbf{X}_{*p}\}$ subset of known $\mathbb{R}^{n_* \times d_q}$ test data matrices,
$\mathbf{H}_1, \mathbf{H}_2, ..., \mathbf{H}_p$ set of $\mathbb{R}^{d_q \times r}$ basis matrices.

Result:

$\mathbf{W}_* \in \mathbb{R}^{n_* \times r}$ coefficient matrix for test data,
$\mathbf{X}_{*t} \in \mathbb{R}^{n_* \times d_t}$ predicted values for unknown test data matrices ($t \neq q$).

Initialize:

$\mathbf{W}_* \sim \mathscr{U}(0,1)^{n_* \times r}$

while not converged do

$$\mathbf{W}_* = \mathbf{W}_* \circ \sqrt{\frac{\sum_q \mathbf{X}_{*q}\mathbf{H}_q}{\sum_q \mathbf{W}_*\mathbf{H}_q^T\mathbf{H}_q}}$$

$\mathbf{X}_{*t} = \mathbf{W}_*\mathbf{H}_t^T$

4

Experiments with iONMF

In this chapter, we describe extensive experiments with iONMF on a problem in predicting RNA-protein interactions using experimental data on 31 RNA-binding proteins. An introduction to protein-RNA interactions, the underlying modeling motivations are described in Appendix A.2. Regardless of the very domain-specific nature of this chapter, we demonstrate some general properties of the iONMF algorithm in relation to other matrix factorization models, such as predictive performance, overlap in the discovered patterns and sparseness. Computationally-oriented reader will nevertheless recognize the basic elements of a supervised machine learning task.

4.1 Sampling of genomic positions

In this chapter, we refer to a *sample* as a genomic location with a length of 100 nucleotides. A *CLIP experiment* refers to one measurement of an RNA-binding protein (RBP) interactions across the whole genome of an organism[1]. The interaction affinity is measured by *cDNA counts* - a continuous value quantifying affinity of an RBP to interact with the RNA of interest. A position where an RBP is found to interact with the RNA is termed a *crosslink*. The selection of positive samples (harbouring crosslinks) and negative samples (with no proteins interacting) is further explained below.

We inferred a model for each of the available 31 CLIP experiments (Suppl. Table C.1). In each CLIP experiment, we first identified up to 100,000 positions with the highest cDNA count. These were used as a pool of positive examples of protein-RNA interacting nucleotides. Among positions which were less than 15 nucleotides apart, we devised a simple peak calling strategy by considering only the positions with the highest cDNA count and ignored all others within a 15 nucleotide distance, as suggested in the original iCLIP publication [54]. With this step we prevented the duplication of practically consecutive genomic positions, which are very similar in composition.

To reduce processing time we sampled up to 10,000 positions per CLIP experiment. For proteins with less than 20,000 identified crosslinking sites, we randomly split the sites into training and test sets. Including more than 10,000 positive examples did not significantly improve the predictive performance of our models (Fig. 4.2, see below). Negative examples of protein-RNA interaction sites were sites within genes that were not detected as interacting in any experiment. Among them we sampled at least 40,000

[1] We use the term CLIP to refer to multiple crosslinking and immunoprecipitation based protocols: CLIPSeq, iCLIP, PARCLIP, HITSCLIP.

positions and used them as negative examples of crosslinking nucleotides. In total, the training set included 50,000 positions (Fig. 4.1a,b). The test set (Fig. 4.1c) was constructed similarly. To ensure a clear separation between the two sets, positions for the test set were sampled only from the genes not used for training. The total number of detected crosslink sites per CLIP experiment are listed in Suppl. Table C.1.

4.2 Data matrices

Each training data matrix included up to 50,000 rows (genomic positions). For experiments performed on a smaller number of positions, the number is explicitly stated. Each row represents a nucleotide position described with the following data sources, providing a number of features (columns):

Y: *selected RBP experiment CLIP cDNA count*, $50{,}000 \times 1$ binary vector. Protein-RNA cDNA counts are reported for a selected RBP experiment for the selected crosslink, resulting in 1 column. This column was used as a target and is the basis for predictive performance evaluation.

$\mathbf{X}_{CLIP}$: *other proteins CLIP cDNA counts*, $50{,}000 \times 3{,}030$ binary matrix. For each of the remaining (up to 30) RBP experiments that were not from the same group as the selected RBP experiment, the cDNA counts at positions $[-50..50]$ relative to the crosslink were reported as 1 for nonzero cDNA counts or 0 otherwise, resulting in up to $30 \times 101 = 3{,}030$ columns. By explicitly ignoring experiments within the same biological group (shown in Suppl. Table C.1), we assured that replicate information was not used in evaluation.

$\mathbf{X}_{RG}$: *Region type*, $50{,}000 \times 505$ binary matrix. Each position $[-50..50]$ relative to the crosslink was assigned to five types of gene regions, as determined by the Ensembl annotation for human genome assembly hg19 [55]: exon, intron, 5'UTR, 3'UTR, CDS, resulting in $5 \times 101 = 505$ columns. Precise boundaries of regions near crosslink sites could thus be captured.

$\mathbf{X}_{RNA}$: *RNA secondary structure*, $50{,}000 \times 101$ real valued matrix. Sequences at positions $[-50..50]$ relative to the crosslink were processed with RNAfold software [56], resulting in probabilities of double-stranded RNA secondary structure at each of 101 relative positions.

$\mathbf{X}_{KMER}$: *RNA k-mers*, 50,000 × 25,856 binary matrix. Positions [−50..50] relative to the crosslink were scanned for the presence of RNA k-mers, with $k = 4$ in all experiments. The presence of a k-mer at a relative position was indicated with a binary value.

$\mathbf{X}_{GO}$: *Gene annotation*, 50,000 × 39,560 binary matrix. Genomic positions within known genes were annotated with Gene Ontology [57] terms for *goa_human*, 39,560 terms (revision 5758736).

Test data matrices ($\mathbf{Y}_*$, $\mathbf{X}_{*\text{CLIP}}$, $\mathbf{X}_{*\text{RG}}$, $\mathbf{X}_{*\text{RNA}}$, $\mathbf{X}_{*\text{KMER}}$, $\mathbf{X}_{*\text{GO}}$) have the same structure, but they described a different subset of positions not included in the training set.

4.3 Analysis overview

A factor model of the training set was inferred with iONMF (Fig. 4.1a). The resulting coefficient matrix $\mathbf{W}$ determined the grouping of samples into r modules, based on similarity across all data sources. A *module* reveals characteristic features in each data source and is represented as a column vector in matrices $\mathbf{H}_q$, corresponding to: co-binding to the same targets as other RBPs ($\mathbf{H}_{\text{CLIP}}$), RNA k-mers ($\mathbf{H}_{\text{KMER}}$), surrounding region types ($\mathbf{H}_{\text{RG}}$), RNA secondary structure ($\mathbf{X}_{\text{RNA}}$) and Gene Ontology terms ($\mathbf{X}_{\text{GO}}$) (Fig. 4.1b).

Having learned the coefficient and basis matrices with iONMF, we estimated the crosslinking affinity of the samples in the test set for all RBP experiments (columns) in the target $\mathbf{Y}$ column (Fig. 4.1c). The test samples were projected into the inferred low dimensional space spanned by $\mathbf{W}$, using all additional data sources ($\mathbf{Y}_*$, $\mathbf{X}_{*\text{CLIP}}$, $\mathbf{X}_{*\text{RG}}$, $\mathbf{X}_{*\text{RNA}}$, $\mathbf{X}_{*\text{KMER}}$, $\mathbf{X}_{*\text{GO}}$) that describe the test set. Each step is described in detail in the following.

4.4 Predictive performance

We compared iONMF against various matrix factorization models: NMF with multiplicative updates [14]; Sparse NMF (SNMF, Eq. 2.3) using alternating non-negative least-squares with L_1 regularization [58]; NMF-QNO using quasi-newton optimization (Section 2.2).

For each RBP experiment, the methods were run on the training set for three different random initializations. The model with the lowest value of the corresponding cost function was used for prediction of the test set with Algorithm 2 (adapted for NMF,

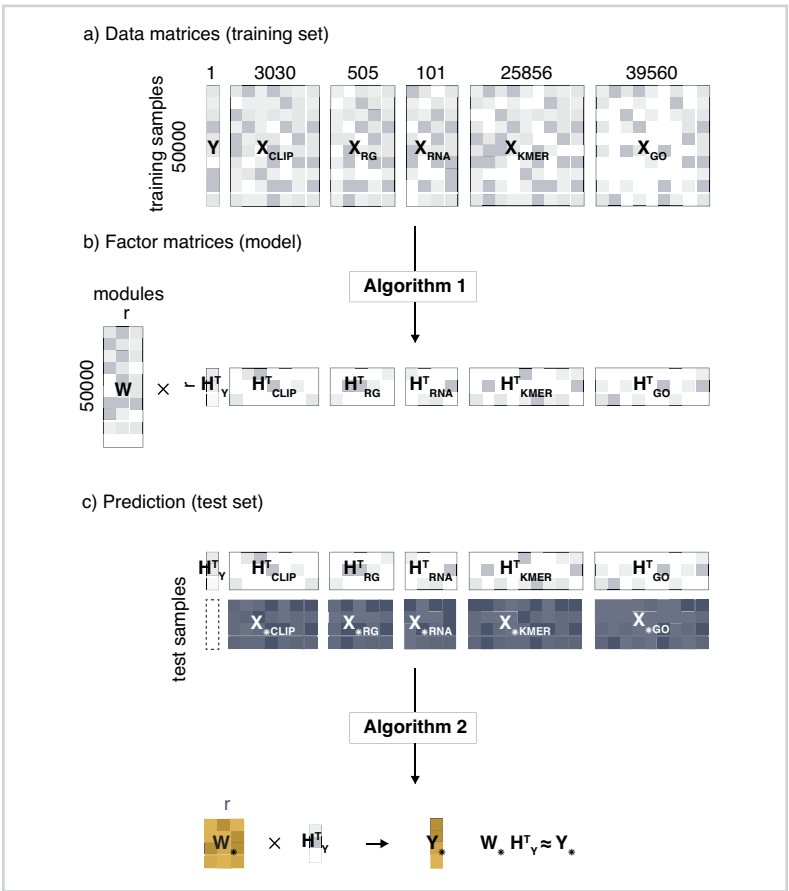


Figure 4.1

Overview of the analysis. a) The target vector $\mathbf{Y}$ and other data sources $\mathbf{X}_q$ are used for model inference. b) iONMF factorization (Algorithm 1) approximates the data sources with a factor model (common coefficient matrix $\mathbf{W}$ and a basis matrix $\mathbf{H}_q$ for each data source). c) Prediction of test samples (Algorithm 2) uses the basis matrices $\mathbf{H}_q, \mathbf{H}_Y$ and test sample data $\mathbf{X}_{*q}$ to estimate the coefficient matrix $\mathbf{W}_*$ and predict $\mathbf{Y}$. Given test data is shown in blue and the predicted variables are shown in orange.

SNMF and NMF-QNO to assume fixed $\mathbf{H}_q$). Samples were projected into the low dimensional space $\mathbf{W}_*$ to predict $\mathbf{Y}_*$. Empirically, algorithms converged in less than 100 iterations (change in cost function value $< 10^{-6}$). The factorization rank was set to $r = 10$ for all methods. Larger ranks did not significantly improve the predictive performance for the price of a higher running time (data not shown).

We used cross-validation (80%/20% sampling, repeated three times) to choose hyperparameters: orthogonality regularization α (iONMF), L_1 regularization (SNMF,

NMF-QNO). Hyperparameters were sampled from the set $\{10^{-3}, 10^{-2}, ...10^{3}\}$. The reported predictive performances are measured with the Area under ROC curve (AUC) on the prediction on the independent hold-out test set of size 1,000. Prediction using iONMF resulted in highest AUC in 24 out of 31 cases. The iONMF, NMF and NMF-QNO methods consistently outperformed SNMF. The critical distance diagram shown in Table 1 confirms the statistical significance ($P < 0.05$) of the observed differences in ranks of predictive models over multiple data sets [59]. This confirms the feasibility of orthogonality as a way to induce discriminative and parsimonious factor models.

The number of training examples critically affects the performance of most statistical models. In Fig. 4.2, we show how the performance of iONMF and NMF changes with increasing training set size. As the number of training examples increases, the gap between predictive performance of iONMF vs. NMF increases as well.

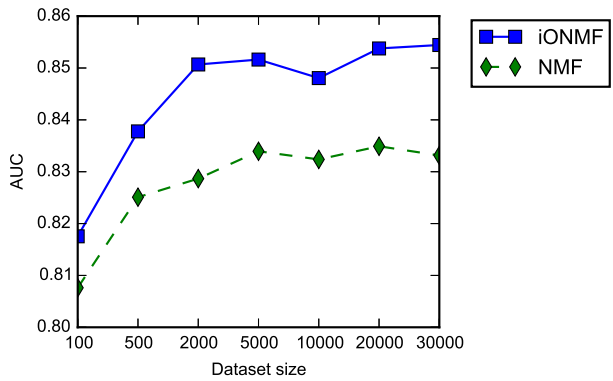


Figure 4.2

Average performance of iONMF and NMF over 31 RBP experiments, depending on the size of the training set. The test set contains 1000 samples with 20% positives.

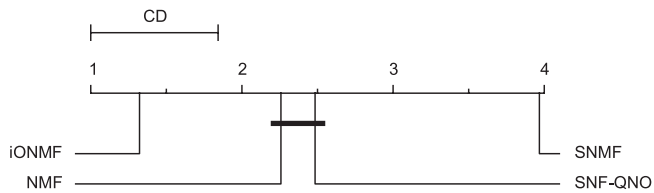
4.5 *Effect of orthogonality*

Next, we investigated the influence of orthogonality hyperparameter α on sparseness (Eq. 2.3) and angle between vectors in $\mathbf{H}_q$. Higher values of angle indicate greater degree of independence between the respective vectors (patterns in $\mathbf{H}_q$). Fig. 4.3 shows the average AUC across all 31 experiments with varying α. As α is increased, sparseness (Eq. 2.3) increases from 0.46 to 0.79, effectively halving the number of non-zero model

Table 4.1

Predictive performance as measured by the Area under ROC curve (AUC) on the hold-out test sets for the evaluated matrix factorization methods. A critical distance diagram of average rankings at significance level $P < 0.05$ is shown below.

Protein	iONMF	NMF	SNMF	QNO	Protein	iONMF	NMF	SNMF	QNO
1 Ago/EIF.	0.89	*0.89*	0.85	0.87	17 hnRNPC	*0.97*	0.96	0.48	0.70
2 Ago2M.	*0.71*	0.69	0.66	0.69	18 hnRNPL	0.74	0.73	0.70	*0.77*
3 Ago2	0.81	0.81	0.76	*0.83*	19 hnRNPL	*0.66*	0.62	0.56	0.61
4 Ago2	*0.84*	0.82	0.79	0.82	20 hnRNPLl.	*0.69*	0.67	0.63	0.68
5 Ago2	*0.73*	0.71	0.65	0.66	21 MOV10	*0.96*	*0.96*	0.89	0.92
6 eIF4AIII	0.92	0.91	0.78	*0.95*	22 Nsun2	0.81	0.80	0.69	*0.82*
7 eIF4AIII	0.93	*0.93*	0.67	0.64	23 PUM2	*0.93*	0.92	0.86	0.89
8 ELAVL1	*0.91*	0.89	0.71	0.80	24 QKI	*0.84*	0.77	0.52	0.62
9 ELAVL1M.	*0.71*	0.70	0.68	0.70	25 SRSF1	*0.85*	*0.85*	0.73	0.86
10 ELAVL1A	*0.94*	0.93	0.91	0.92	26 TAF15	*0.91*	0.89	0.82	*0.91*
11 ELAVL1	0.95	0.94	0.90	*0.95*	27 TDP-43	*0.84*	0.78	0.45	0.57
12 ESWR1	*0.87*	0.85	0.80	0.85	28 TIA1	*0.93*	0.92	0.86	0.90
13 FUS	*0.81*	0.73	0.55	0.65	29 TIAL1	*0.87*	0.86	0.73	0.85
14 Mut FUS	*0.96*	0.95	0.91	0.94	30 U2AF2	*0.82*	0.74	0.61	0.70
15 IGF2.1-3	*0.93*	0.92	0.89	0.91	31 U2AF2	*0.80*	0.74	0.60	0.74
16 hnRNPC	*0.95*	0.93	0.45	0.63					



parameters. Also, the average pairwise angle between vectors in $\mathbf{H}_q$ increases from 65° to 90°. Even for extreme values of α, AUC changes only slighthly, from 0.88 to 0.83. We compared the feature vectors found by all factorization methods in more detail, see Section 4.6.

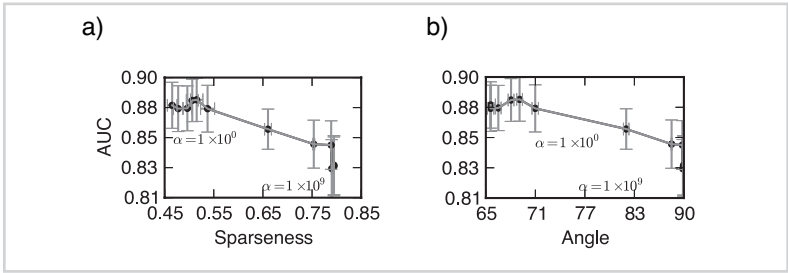


Figure 4.3

Effect of parameter α on performance (y axis) versus a) sparseness and b) angle of basis vectors $\mathbf{H}_q$ (right) of model obtained with iONMF.

4.6 Overlap between modules

To illustrate the advantage of orthogonal factorization, we examine the differences in feature vectors discovered by each matrix factorization method in more detail. Orthogonality is related to the phenomenon of *multicollinearity* in the context of linear regression, where multiple feature vectors in a model are highly correlated [60]. This may lead to suboptimal prediction performance and can have an effect on the magnitude of particular regression coefficients. Our framework can be seen as learning multiple regression models, where each row $\mathbf{x}_j \in \mathbf{X}$ is predicted by a (non-negative) linear combination of feature vectors in $\mathbf{H}$ given by coefficients in row $\mathbf{w}_j$.

For each RBP experiment and model, we examine the feature vectors in each $\mathbf{H}_q$. Let ρ_j be the maximal Pearson correlation of each feature vector $\mathbf{h}_j \in \mathbf{H}_q$ with any other feature vector in $\mathbf{h}_k \in \mathbf{H}_q$, $k \neq j$. The Fig. 4.4 shows the average maximal correlation $\bar{\rho}$ computed over all feature vectors in a particular model.

The models inferred with iONMF show the smallest average correlation in 30 out of 31 experiments. Also, the feature vectors are on average the least correlated in models found by iONMF when averaged over all 31 experiments (shown by horizontal lines in Fig. 4.4). This results support the claim that redundancy is alleviated and supports the improved predictive performance.

4.7 Estimated importance of data sources

We evaluated the predictive performance for each possible subset of data sources and each RBP experiment (Suppl. Tables C.3–C.5 for AUC of individual experiments). We then calculated the average AUC and standard error obtained with each data source subset across all selected RBP experiments, shown in Fig. 4.5 and Suppl. Table C.2.

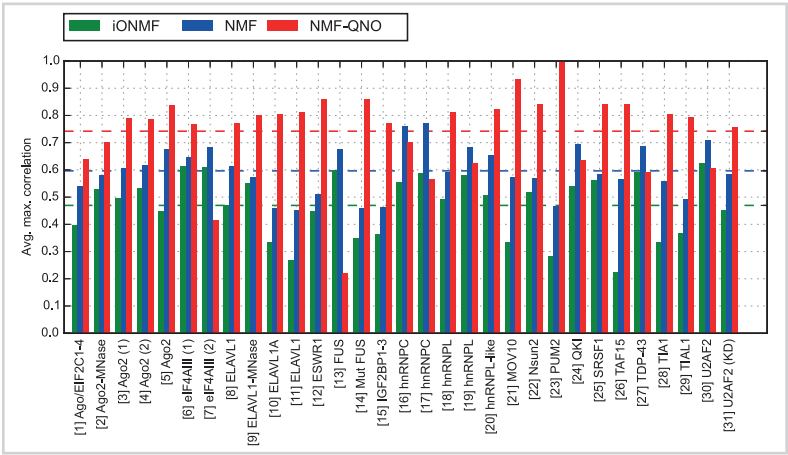


Figure 4.4

Comparison of average maximal pairwise correlation between low-rank components (rows in $\mathbf{H}$). The feature vectors obtained by SNMF report an $\bar{\rho} = 0.01$ and were not included into the figure due to inferior predictive performance resulting from overly sparse vectors.

To ensure fair comparison, the factorization rank r was selected such that the total number of model parameters (number of values in $\mathbf{W}$ and $\mathbf{H}_q$) was approximately equal for each subset of data sources (Suppl. Table C.2).

First, we discuss the importance of each single data source. According to AUC, the most informative data source is RNA structure (col. R, average AUC=0.744 ± 0.024, Fig. 4.5, Suppl. Table C.2). This agrees with previous observations about the importance of particular RNA structure interaction interfaces [61, 62], but may also reflect the need for RNA bases to be single stranded to allow UV crosslinking [63]. The second most informative data source is interaction with other proteins within the same region (col. C, average AUC=0.732 ± 0.018, Fig. 4.5, Suppl. Table C.2). This agrees with combinatorial protein-RNA interactions that compete or cooperate for RNA binding [64, 65], but may also indicate that many RNA nucleotides may have generally increased accessibility and crosslinking efficiency, that might be affected by variance in gene expression.

The most informative pair of data sources are RNA k-mers (K) and type of genomic region (T) with average AUC=0.860 ± 0.017 (col. KT Fig. 4.5, Suppl. Table C.2). These features describe the genomic organization and sequence content biases of functional subunits, e.g. exon, intron, untranslated regions (UTR), and exon-intron boundaries.

The poor performance when using $\mathbf{X}_{GO}$ alone (col. G, AUC=0.492±0.008, Fig. 4.5, Suppl. Table C.2) is likely due to sparse and incomplete gene function annotation, which is in great contrast to other data sources that are two orders of magnitude denser (density of $\mathbf{X}_{GO}$ is 0.01%, while density of $\mathbf{X}_{RG}$ is 16%). Inclusion of $\mathbf{X}_{GO}$ into the most informative subset (col. CKRT, AUC=0.920 ± 0.006, Fig. 4.5, Suppl. Table C.2) does not change the predictive accuracy significantly (col. CGKRT, AUC=0.886 ± 0.011, Fig. 4.5, Suppl. Table C.2). As $\mathbf{X}_{GO}$ is larger than all other data sources combined together, this result supports the claim that orthogonality regularization in the iONMF cost function acts as a subtle normalization with respect to data source size.

The average AUC correlates with the number of included data sources. The best accuracy was achieved on $\{\mathbf{X}_{CLIP}, \mathbf{X}_{KMER}, \mathbf{X}_{RNA}, \mathbf{X}_{RG}\}$, col. CKRT. Except for $\mathbf{X}_{GO}$, combining two or more data sources resulted in better accuracy than in models obtained on individual data sources, supporting the benefit of data integration. A more detailed, Spearman correlation-based comparison of all data source combinations confirms several binding preferences supported by the literature (Suppl. Section C.3.2 and Suppl. Fig. C.1).

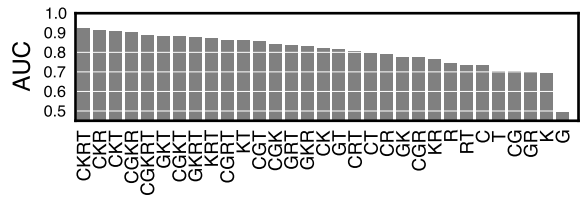


Figure 4.5

Average AUC for 31 experiments, using all possible subsets of data sources: CLIP experiments (C; $\mathbf{X}_{CLIP}$), RNA k-mers (K; $\mathbf{X}_{KMER}$), region type (T; $\mathbf{X}_{RG}$), Gene Ontology terms (G; $\mathbf{X}_{GO}$) and RNAfold structure prediction (R; $\mathbf{X}_{RNA}$).

4.8 *Identified factors associated with RBP binding*

A principal motivation for development of iONMF was to to identify the characteristic features associated with each discovered module. As explained in Section 3.4 and shown in Fig. 3.1d, each module reveals common feature values of crosslinked sites (samples) assigned to the module. These values are reflected in $\mathbf{H}_q$, one row for each of the r modules. The identified modules of crosslinked sites with common features were visualized and used to predict functionally relevant protein-RNA interactions.

Visualization of a complete set of RBP experiments is excluded from this book in favour of brevity, but can nevertheless be found in [66].

4.8.1 iONMF identifies biologically relevant binding patterns

We present the results and provide an explanation for an example RNA-binding protein U2AF2, a known splicing factor, where the most informative single data sources are ordered by predictive performance as follows: $\mathbf{X}_{\text{KMER}}$ (AUC=0.695), $\mathbf{X}_{\text{RG}}$ (AUC=0.673), $\mathbf{X}_{\text{CLIP}}$ (AUC=0.650), $\mathbf{X}_{\text{RNA}}$ (AUC=0.593), $\mathbf{X}_{\text{GO}}$ (AUC=0.505); see Suppl. Table C.5. The most informative data subset is $\{\mathbf{X}_{\text{CLIP}}, \mathbf{X}_{\text{KMER}}, \mathbf{X}_{\text{RG}}\}$ (AUC=0.925).

RNA secondary structure. In agreement with U2AF2 being a single-stranded RNA binding protein, the probability of double stranded RNA decreases around its crosslinked sites. Features in $\mathbf{H}_{\text{RNA}}$ are shown in Fig. 4.6. Hierarchical clustering of feature vectors in $\mathbf{H}_{\text{KMER}}$ are shown in Suppl. Fig. C.4, C.5.

RBP co-binding and k-mer composition. Examining features in $\mathbf{H}_{\text{CLIP}}$, one is able to discover factors associated with binding of individual or groups of RBPs. The features that are common to each module allow us to define hypotheses on cooperative or competitive binding of multiple proteins, which can then be experimentally tested. A global picture of RNA motifs associated to all proteins is shown on 4.7.

Fig. 4.6 shows results for U2AF2. It also shows that splicing factor hnRNPC interacts with the same RNA positions as U2AF2. Competition between the two is reported by [67]. The two factors also share similar binding motifs (Fig. 4.7). The relationship is further confirmed by recognition of U-rich motifs, appearing in the corresponding module in $\mathbf{X}_{\text{KMER}}$.

Region type. Fig. 4.6 shows $\mathbf{H}_{\text{RG}}$ features for U2AF2. The intron-exon boundary can be seen at ~30 nucleotides upstream from the crosslinked site. This is expected since U2AF2 is a splicing factor that generally crosslinks to a 3' splice site [67]. Protein similarity based on region types is shown in Fig. 4.8, confirming the ability of iONMF feature vectors to cluster the proteins into functionally related groups. Detailed data is shown in Suppl. Fig. C.7 and individual feature vectors.

Sequence motif content and positioning. Fig. 4.6 shows the sequence content and positions of RNA sequence k-mers (features in $\mathbf{H}_{\text{KMER}}$) for U2AF2. The most associated k-mers are U-rich and are similar to recognition sites of hnRNPC, an experimentally confirmed competitor for the same binding sites [67]. Co-binding of the two can be

seen in $\mathbf{H}_{\text{CLIP}}$ matrices.

Gene annotation. Due to extensive length of the Gene Ontology associations, and in favour of brevity, the relevant results can be found in [66].

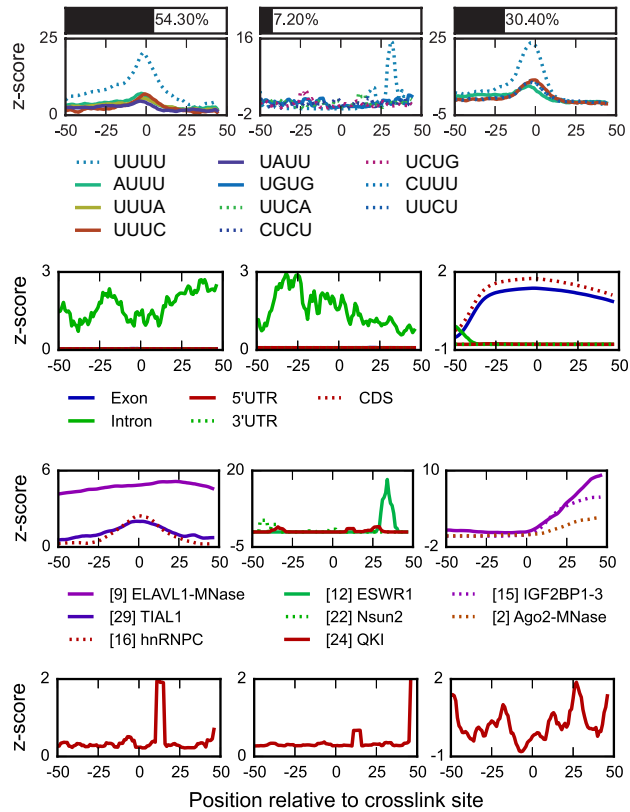


Figure 4.6

Three modules most associated with positions bound by U2AF2 are shown, top to bottom: $\mathbf{H}_{\text{KMER}}$, $\mathbf{H}_{\text{RG}}$, $\mathbf{H}_{\text{CLIP}}$, $\mathbf{H}_{\text{RNA}}$. Top bars show the percentage of samples described by the corresponding module.

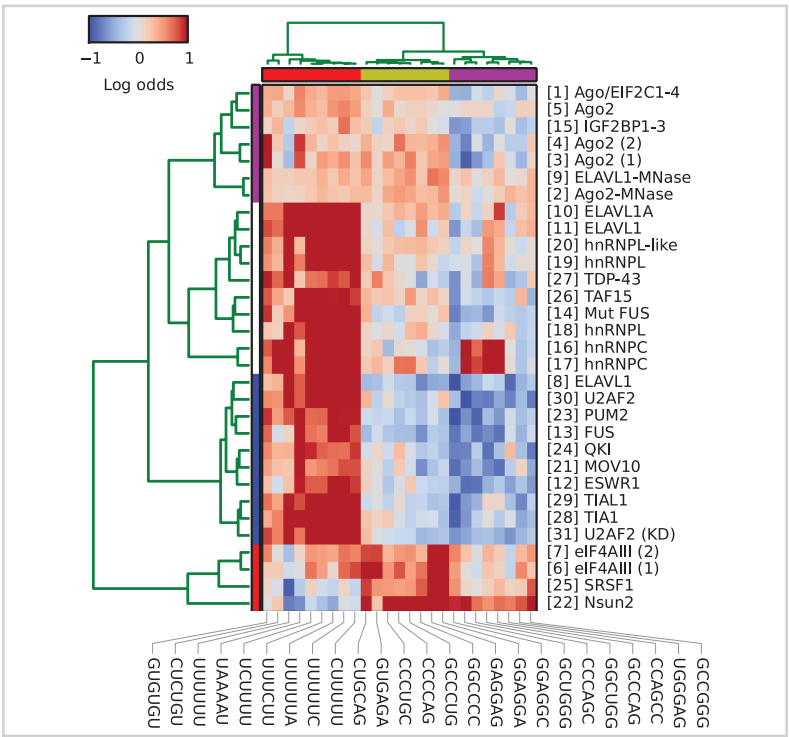


Figure 4.7

Hierarchical clustering (Ward's linkage) of proteins and 20 most common complex motifs, estimated from row vectors $\mathbf{H}_{\text{KMER}}$ obtained with iONMF. Heatmap shows log odds ratios of observed motif probability in sites proximal to crosslinked sites divided by the expected probability (at random positions).

4.8.2 Orthogonality constraints demultiplex binding patterns

Orthogonality regularization provides an advantage in model interpretation over NMF. CLIP-based protocols are subject to a U-rich sequence preference due to UV-C crosslinking. As reported previously [63], the detection of U-rich motifs may occur at crosslinks for RBPs not associated with U-rich tracts, such as TDP-43 (Fig. 4.9). The NMF method discovers both U-rich motif and known tandem UG repeats in a single module (column vector in $\mathbf{H}_{\text{KMER}}$), while iONMF successfully distinguishes the two. Assigning the data samples to corresponding modules (Section 3.4), 41.6% of positive samples are related to UG-rich component, while 80.3% are related to U-tracts. In summary, orthogonality regularization can disambiguate between different modalities of RBP binding.

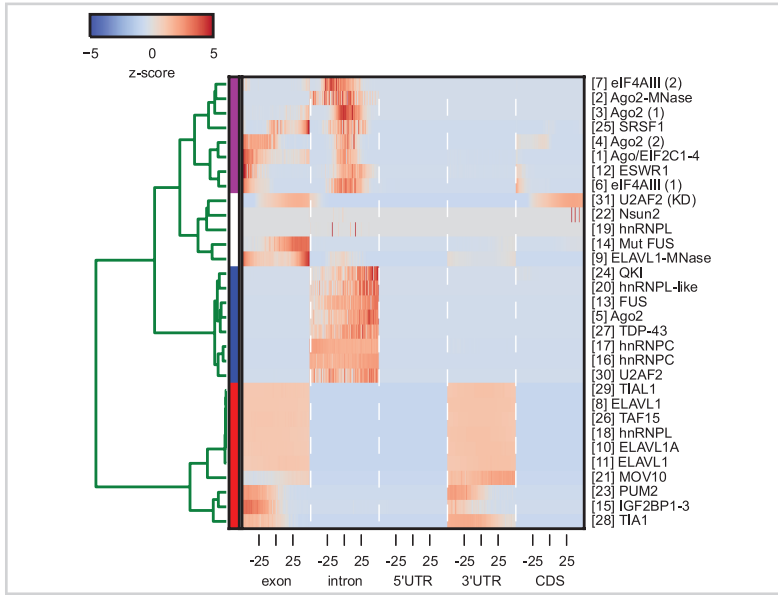


Figure 4.8

Protein similarity based on gene region types row vectors in $\mathbf{H}_{RG}$. For each region type, the interval [-50..50] relative to the crosslinked sites is shown.

4.9 Summary on biological results

Computational approaches already play a crucial role in protein-RNA interaction prediction by aiding experiment planning and interpretation of results. Genome-wide assays of protein-RNA interaction mapping [68] has identified close to a thousand human RNA-binding proteins. Data on RNA binding proteins is growing rapidly, emphasizing the need for integrative methods which jointly consider all available data sources.

An interesting finding of our study is that in addition to RNA structure and sequence, the position relative to genomic features (exons, etc.) and CLIP data of other RBPs is informative for predicting binding sites of a specific RBP. Genomic regions are informative as many proteins bind at specific positions relative to these features, e.g. U2AF2 generally binds upstream of exons (Fig. 4.6). We show that CLIP data are predictive, as subsets of examined RBPs exhibit similar binding patterns (Suppl. Fig. C.2). Importantly, overlap is only seen between a subset of RBPs, but we find no evidence that some sites or features are generally shared across all RBPs. While

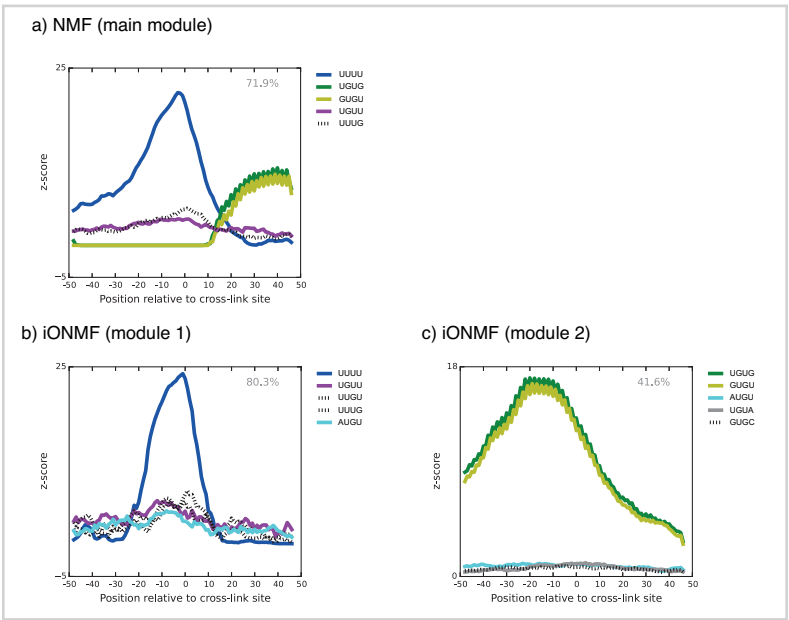


Figure 4.9

Comparison of modules in $\mathbf{H}_{\text{KMER}}$ related to crosslinks of TDP-43. a) Canonical NMF. b, c) Top two relevant components found by iONMF.

contribution of non-specific background should be considered, we find it most likely that co-binding profiles result from biologically relevant features. For example, many RBPs bind to similar RNA sequences or structures (Suppl. Fig. C.2-C.5).

An important consideration when analyzing CLIP-based data is the experimental bias, causing certain sequences to be non-specifically detected. One such example are the U-rich sequences that appear to be especially susceptible to crosslinking [69]. Orthogonal matrix factorization can be useful in this cases, disambiguating between multiple different patterns of varying strengths 4.9.

Several of the examined RBPs are known to bind similar motifs, such as the U-rich motifs bound by ELAVL, TIA, hnRNPC and U2AF2, which are also detected in our analyses (Fig. 4.7). Moreover, RBPs may interact at the protein level, either directly or indirectly via co-factors, which could stabilize their binding to proximal RNA sites. Few experimental studies have explored the impact of protein-protein interactions on coordinated RNA binding, but our analyses could be used to explore

such potential interactions in the future. Few experimental studies have explored the impact of protein-protein interactions on coordinated RNA binding, but our analyses could be used to explore such potential interactions in the future.

4.10 Summary on orthogonal matrix factorization

One of the challenges in the application of the matrix factorization-based models is the presence of multiple patterns of varying magnitude within multiple data sources, that are failed to be detected by the unconstrained NMF. We design a model based on orthogonal non-negative matrix factorization and the corresponding optimization algorithm.

Orthogonality regularization favors both non-overlapping and sparse solutions, providing class-specific descriptions and model interpretation. It favours independent patterns providing competent predictive performance. By evaluating the pairwise dependence of feature vectors and sparseness, we empirically show there exist predictive models with equivalent predictive performance but higher sparseness and degree of independence between feature vectors. In practice, it enables to disambiguate the patterns, hardly detectable by the canonical NMF.

Data integration in iONMF yields improvements in accuracy when compared to state-of-the-art approaches. The resulting predictions are in strong accordance with a published *in vitro* studies and identified a number of promising candidates for further investigation. Despite a highly specific aim of the presented experiments it is important to note that iONMF can be used for general purpose machine learning. Our experimental findings establish iONMF as the data integration technique of choice where sparse, modular models are desired.

5

Kernel methods

A principal assumption underlying the matrix factorization algorithms designed so far is that the data can be approximated as a combination of one or more linear models. Modeling output functions that can be defined by non-linear functions of the input variables can be achieved by kernel methods. In this second part of the work, we use kernel methods to design a similar model based on multiple data sources, represented as kernel matrices. In this chapter, we present the basic theory and related work in kernel regression, approximation and multiple kernel learning.

A central premise in machine learning is the trade-off between model complexity and pattern stability. Many scenarios include a set of output (target) variables related to the input variables through non-linear functions. The increase in allowed model complexity translates to increasing the space of candidate functions and enables modelling of a larger set of problems, but increases the probability of finding a good matching between the inputs and the outputs simply by chance. Kernel methods leverage the data representation into higher dimensional (possibly infinite) input spaces while enabling the inference using the existing algorithms. The central idea consists of thinking about input data as pair-wise similarities between data points, obtained through a *kernel function*, which is an inner product corresponding to a feature representation in a higher dimensional space. This allows one to have control over the both the class of output functions.

This chapter introduces a minimal set of necessary mathematical and modelling formalisms used to derive further results related to kernel methods. A short basic section on inner product spaces is provided in the Appendix B.3.1.

5.1 Kernel functions

Assume the existence of an arbitrary input space $\mathscr{X}$ and a map ϕ to a Hilbert space[1], $\phi : \mathscr{X} \mapsto \mathscr{H}$. In the space $\mathscr{H}$, the inner product $\langle \cdot, \cdot \rangle_{\mathscr{H}}$ corresponds to the inner product between feature mappings $\langle \phi(\mathbf{x}), \phi(\mathbf{x'}) \rangle_{\mathscr{H}}$ for any two $\mathbf{x}, \mathbf{x'} \in \mathscr{X}$. In practice, the knowledge of ϕ is seldom required if we are able to define a kernel function k that corresponds to the evaluation of the inner product between two feature mappings

$$k(\mathbf{x}, \mathbf{x'}) = \langle \phi(\mathbf{x}), \phi(\mathbf{x'}) \rangle_{\mathscr{H}}.$$

[1] A Hilbert space is an inner product space which is separable and complete. It allows for the computations of angles and length. Additionally, completeness implies that any Cauchy sequence in $\mathscr{H}$ converges to an element of $\mathscr{H}$. For an introduction to Hilbert spaces, see Kreyszig [70].

Let $\mathbf{x}, \mathbf{x}'$ be two elements of the space $\mathscr{X}$. Valid kernel functions are characterized by the positive-semidefiniteness property, defined as follows.

Positive semi-definite functions. Let $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ be a collection of n arbitrary elements. A symmetric matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ (or $\mathbb{C}^{n \times n}$) resulting from the pairwise evaluations of a kernel function $\mathbf{K}(i, j) = k(\mathbf{x}_i, \mathbf{x}_j)$, is positive semi-definite if

$$\text{for any vector } \mathbf{a} \in \mathbb{R}^n \text{ (or } \mathbb{C}^n\text{), we have } \mathbf{a}^T \mathbf{K} \mathbf{a} \geq 0.$$

Any function $k(\cdot, \cdot)$ satisfying this property is a positive semi-definite function and can be proven to correspond to an inner product in a Hilbert space $\mathscr{H}$ [71]. Note that with this definition, we need not explicitly construct a mapping $\phi(\mathbf{x})$ that would correspond to the kernel of interest. A general mapping that corresponds to any valid kernel is discussed in more detail in Section 5.2 below. Furthermore, the definition does not put any restrictions on the original space $\mathscr{X}$.

The practical consequence is the following. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a matrix of n data points in a feature space of dimension d. In the context of linear regression, the *Gram* matrix $\mathbf{X}\mathbf{X}^T$ in Eq. B.14 can be replaced with a kernel matrix $\mathbf{K}$ where $\mathbf{K}(i, j) = k(\mathbf{x}_i, \mathbf{x}_j)$ corresponds to the evaluation of the kernel function k. Thus, linear regression (and many other models) can be defined in the feature space induced by ϕ only using the knowledge on inner products by replacing $\mathbf{X}\mathbf{X}^T$ in Eq. B.14. with the kernel matrix $\mathbf{K}$. This substitution is colloquially referred to as the *kernel trick*. It enables learning non-linear output functions (with respect to the original input space $\mathbf{X}$) using linear systems of equations.

Polynomial kernel. We give a concrete example of a kernel where a feature mapping ϕ can be written and evaluated explicitly. Let the $\mathscr{X} = \mathbb{R}^2$ and feature map $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$. Then, the polynomial kernel between $\mathbf{x}, \mathbf{z} \in \mathscr{X}$ can be evaluated as

$$\begin{aligned} \langle \mathbf{x}, \mathbf{z} \rangle_{\mathscr{H}} = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T (z_1^2, \sqrt{2}z_1z_2, z_2^2) = \\ = x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2 = ((x_1, x_2)^T (z_1, z_2))^2 = \\ = \langle \mathbf{x}, \mathbf{z} \rangle^2 = k_{\text{poly}}(\mathbf{x}, \mathbf{z}). \end{aligned}$$

Hence, the evaluation of the polynomial kernel $k_{\text{poly}}(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$ is equivalent to the evaluation of inner products between feature maps $\phi(\mathbf{x})$ and $\phi(\mathbf{z})$. Importantly,

the feature mappings need never be evaluated explicitly to efficiently compute inner products.

More examples of common kernels are listed in Table 5.1. While linear regression was chosen as a worked example in Appendix B.3.2, the kernel trick can be used to *kernelize* many linear models used in machine learning, such as principal component analysis (PCA), linear discriminant analysis (LDA), support vector machines (SVM) and more. Choosing a kernel function represents the prior assumptions on the relations between the original input space and the target output functions. The kernel functions provide different ways of measuring similarities between data points, which is illustrated with images of kernels in Figure 5.1 on a subset of $\mathbb{R}^1$. More examples of kernels and the intuition on the resulting output functions are discussed in the following paragraphs.

Exponentiated-quadratic kernel. The exponentiated-quadratic kernel is one of the most common kernels used in machine learning, particularly with Gaussian Processes, where it has gained a nickname *smoothing device*. The latter fact is attributed to the infinite differentiability of the exponential function, resulting in infinitely smooth output functions. The functional form of the exponentiated-quadratic kernel is

$$k_{\text{exp}}(\mathbf{x}, \mathbf{x'}) = \sigma^2 \exp(-\frac{||\mathbf{x} - \mathbf{x'}||^2}{\ell^2}). \tag{5.1}$$

It is stated as a function of the difference $\mathbf{x} - \mathbf{x}$, and thus an example of a *stationary* or *translation-invariant* kernel. The length scale hyperparameter ℓ determines the distance (in terms of norm $||\mathbf{x} - \mathbf{x'}||$) at which the output functions $f(\mathbf{x})$ and $f(\mathbf{x'})$ covary (see the next Section, 5.2 ,for more discussion on output functions). Larger length scales thus imply long-range effects between data points. Similarly, when stationary kernels are treated as one-dimensional functions, they are particularly useful in spectral analysis, which is a principal tool for stationary kernel decompositions. An alternative but equivalent form of the kernel is

$$k_{\text{exp}}(\mathbf{x}, \mathbf{x'}) = \sigma^2 \exp(-\gamma ||\mathbf{x} - \mathbf{x'}||^2), \tag{5.2}$$

where the length scale is absorbed in the frequency hyperparameter γ. This last form is included to comply with existing literature later in the text.

The Matérn kernel. The Matérn kernel kernel is another example of a stationary

kernel, where both the smoothness and the length scale of the output functions can be controlled explicitly [72]. The general form of the Matérn kernel is defined as follows:

$$k_{\text{Mat}}(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\|\mathbf{x}-\mathbf{x}'\|}{\ell}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}\|\mathbf{x}-\mathbf{x}'\|}{\ell}\right), \tag{5.3}$$

where K_ν is a modified Bessel function and $\Gamma(\nu)$ is the Gamma function [73]. The hyperparameters control the smoothness ν, and the length scale ℓ. The latter plays a very similar role to the length scale ℓ in the exponentiated-quadratic kernel; indeed, as the smoothness parameter $\nu \to \infty$, the output functions are infinitely smooth. Conversely, small values of ν result in very rough functions. In case ν is of form $p + 1/2$, where p is an integer, the output functions are p times differentiable. The two common kernels used in machine learning, depending on ν, are the Matérn 3/2 and Matérn 5/2 kernels:

$$k_{\text{Mat},3/2}(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\sqrt{3}\|\mathbf{x}-\mathbf{x}'\|}{\ell}\right)\exp\left(-\frac{\sqrt{3}\|\mathbf{x}-\mathbf{x}'\|}{\ell}\right), \tag{5.4}$$

$$k_{\text{Mat},5/2}(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\sqrt{5}\|\mathbf{x}-\mathbf{x}'\|}{\ell} + \frac{\sqrt{5}\|\mathbf{x}-\mathbf{x}'\|^2}{\ell}\right)\exp\left(-\frac{\sqrt{3}\|\mathbf{x}-\mathbf{x}'\|}{\ell}\right). \tag{5.5}$$

Hence, with ν of form $p + 1/2$, the Matérn kernel is composed of a product of a p-degree polynomial and an exponential when treated as a one-dimensional function of the distance $\|\mathbf{x} - \mathbf{x}'\|$.

String kernels. An example where the kernel methods enable modeling of data which is not straightforwardly represented in vector spaces is presented next. In this case, the data points are discrete, finite length, finite character set strings. This is a common scenario in text processing or bioinformatics, where kernels exposing different properties and patterns exists [74].

The data points are strings of length at most L, derived from a finite alphabet $\mathscr{B}$, defining the space of data points as $\mathscr{B}^L$ (including the empty character). The string-type data can be transformed into a numerical form using e.g. manual preprocessing, string kernels, or deep learning.

Arguably the simplest string kernel is the *spectrum kernel*, called also the n-gram bag-of-words kernel. The inner product between two strings $\mathbf{s}, \mathbf{s}' \in \mathscr{B}^L$ is defined as a product of occurrences of each substring $\mathbf{u}$ of length $\ell < L$:

$$k_{\text{SPEC}}(\mathbf{s}, \mathbf{s'}) = \sum_{\mathbf{u} \in \mathscr{B}^{\ell}} \text{count}(\mathbf{u}, \mathbf{s}) \times \text{count}(\mathbf{u}, \mathbf{s'}) \tag{5.6}$$

with the substring length ℓ as a hyperparameter and count$(\mathbf{u}, \mathbf{s})$ equals the number occurrences of $\mathbf{u}$ in $\mathbf{s}$. Note that this kernel does not regard any positional information of $\mathbf{u}$ within strings. The feature space explicitly representing the spectrum kernel could be simply constructed by listing all substrings and the respective counts, but such construction becomes unpractical as the size of the feature space grows exponentially as the size of the alphabet $|\mathscr{B}|^{\ell}$. A kernel, dependent on the positional information is the *substring* kernel[2], which assumes that the same length L for all strings and compares substrings of length ℓ.

$$k_{\text{SUB}}(\mathbf{s}, \mathbf{s'}) = \sum_{j=1}^{L-\ell+1} I(\mathbf{s}[j : j + \ell] == \mathbf{s'}[j : j + \ell]), \tag{5.7}$$

where the indicator function $I(\text{expr})$ is equal to 1 if the expression in the argument is true. The substring kernel maps the data into an even larger feature space of dimension $(L-\ell+1)\times|\mathscr{B}|^{\ell}$. An example of a kernel that is not trivial to represent in a vector space is the *substring kernel with mismatches*, permitting ℓ_0 mismatches within substrings of size ℓ:

$$k_{\text{SUB-MIS}}(\mathbf{s}, \mathbf{s'}) = \sum_{j=1}^{L-\ell+1} I\Big(\text{match}(\mathbf{s}[j : j + \ell], \mathbf{s'}[j : j + \ell]) > (\ell - \ell_0)\Big), \tag{5.8}$$

where the match$(\mathbf{t}, \mathbf{t'})$ function counts the number of exact positional matches for a pair of strings $\mathbf{t}, \mathbf{t'}$. The feature space implied by this kernel is not representable as a vector space due to a particular substring now matching multiple other substrings equally, implying non-independent components of a space.

[2] In the original publication, Sonnenburg et al. [74] refer to this kernel as *weighted degree kernel*, where different lengths of substrings $\ell = 1, 2..., \ell_{\max}$ can be weighted according to parameters $\beta_1, \beta_2..., \beta_{\max}$. This definition includes elements of multiple kernel learning, which is treated later in this book.

Table 5.1

Common kernels used in this work.

Label	Name	Input space	Expression	Hyperparameters ($\boldsymbol{\theta}$)
$k_{\text{lin}}(\mathbf{x}, \mathbf{x}')$	Linear	$\mathbb{R}^d$	$\mathbf{x}^T\mathbf{x}'$	
$k_{\text{poly}}(\mathbf{x}, \mathbf{x}')$	Polynomial (of degree D)	$\mathbb{R}^d$	$(\mathbf{x}^T\mathbf{x}')^D$	D
$k_{\text{exp}}(\mathbf{x}, \mathbf{x}')$	Exponentiated-quadratic	$\mathbb{R}^d$	Eq. 5.1-5.2	σ, ℓ
$k_{\text{sig}}(\mathbf{x}, \mathbf{x}')$	Sigmoid	$\mathbb{R}^d$	$\tanh(\gamma\mathbf{x}^T\mathbf{x}' + b)$	γ, b
$k_{\text{Mat},3/2}(\mathbf{x}, \mathbf{x}')$	Matérn 3/2	$\mathbb{R}^d$	Eq. 5.4	ℓ
$k_{\text{Mat},5/2}(\mathbf{x}, \mathbf{x}')$	Matérn 5/2	$\mathbb{R}^d$	Eq. 5.5	ℓ
$k_{\text{SPEC}}(\mathbf{s}, \mathbf{s}')$	Spectrum	$\mathscr{B}^L$	Eq. 5.6	ℓ
$k_{\text{SUB}}(\mathbf{s}, \mathbf{s}')$	Substring	$\mathscr{B}^L$	Eq. 5.7	ℓ
$k_{\text{SUB-MIS}}(\mathbf{s}, \mathbf{s}')$	Substring mismatch	$\mathscr{B}^L$	Eq. 5.8	ℓ, ℓ_0

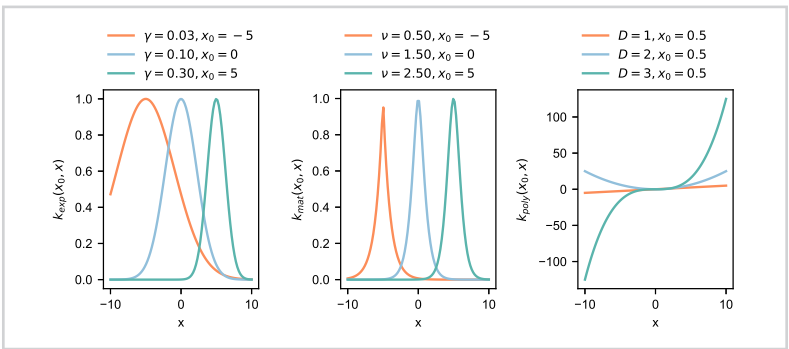


Figure 5.1

Basis functions of exponentiated-quadratic, Matérn and polynomial kernel for different hyperparameter values. The kernel $k(x, x_0)$ is computed for $x \in [-10, 10] \subset \mathbb{R}^1$, with the inducing points x_0 selected to avoid clutter.

5.2 Output function spaces

It is interesting to examine the form of output regression functions when a kernel is used in place of the inner products in linear regression (Eq. B.14, page 131). The functional form gives a direct insight into the properties of functions induced by a particular kernel. To achieve this, we must introduce the *Reproducing kernel Hilbert space* (RKHS). Assume that $\mathscr{X}$ is a general vector space. The RKHS is a space of functions defined by a point in the original input space $\mathscr{X}$ with the inner product

equal to the kernel. In general inner product spaces, a linear functional[3] is defined by an inner product and an element $\mathbf{x} \in \mathscr{X}$ [70]. Similarly, in Hilbert spaces, the linear functionals can be defined by a point in $\mathscr{X}$ and a kernel k.

Concretely, let $\mathbf{x} \in \mathscr{X}$ be a point in the original vector space and k a kernel function. Then, a function $f \in \mathscr{H}$ can be defined by the point $\mathbf{x}$ as follows

$$f = k(\mathbf{x}, \cdot),$$

where $\cdot$ indicates the function argument. A *evaluation* of f at a point $\mathbf{x'}$ is then defined as

$$f(\mathbf{x'}) = k(\mathbf{x}, \mathbf{x'}).$$

The evaluation of $f(\mathbf{x'})$ is itself defined in the form of an inner product $\langle \cdot, \cdot \rangle_{\mathscr{H}}$. Hence, linear combinations of elements in $\mathscr{H}$ give rise to other elements of $\mathscr{H}$. For example, a linear combination of elements $\boldsymbol{\alpha} \in \mathbb{R}^n$ gives rise to the function

$$g = \sum_i^n \alpha_i k(\mathbf{x}_i, \cdot),$$

which is also an element of $\mathscr{H}$. The function evaluation for an arbitrary element in $\mathscr{X}$ is thus

$$g(\mathbf{x'}) = \sum_i^n \alpha_i k(\mathbf{x}_i, \mathbf{x'}). \tag{5.9}$$

Note the similarity of Eq. 5.9 with Eq. B.10 where the kernel k was used in place of the canonical inner product. The output functions — inferred given a training set $\mathscr{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ and a kernel k — are linear combinations of functions in $\mathscr{H}$, centered at the points in $\mathscr{D}$.

5.3 Multiple kernel learning

Combining different representations of the same set of objects is considerably facilitated with kernels. There even exist situations where the original input space can not

[3] A linear functional is a function from a vector space to its field of scalars and are of form $f : \mathscr{X} \mapsto \mathbb{R}$.

be associated to a vector space, but for which kernels can be defined, leading to an implicit definition of the corresponding input space. Common representations of objects include real vector spaces, discrete spaces, discrete character strings, graphs, trees, and more. Multiple kernel learning (MKL) not only facilitates selection of kernels from predefined sets, but also enables seamless operation with different representations of the same set of objects [31].

5.3.1 *Making new kernels from old*

Following the properties of kernels, various rules exists to combine two or more different kernels and obtain new valid kernels (Bishop [29], ch. 6). For example, the *sum rule* for combining two kernels can be straightforwardly shown. Given two kernels k_1 and k_2, the sum

$$k_{1+2}(\mathbf{x}, \mathbf{x'}) = k_1(\mathbf{x}, \mathbf{x'}) + k_2(\mathbf{x}, \mathbf{x'})$$

is also a valid kernel. Letting $\mathbf{K}_1$ and $\mathbf{K}_2$ be the corresponding kernel matrices, both of size $\mathbb{R}^{n \times n}$, the positive-definiteness property is preserved by kernel sum

$$\mathbf{a}^T \mathbf{K}_{1+2} \mathbf{a} = \mathbf{a}^T \mathbf{K}_1 \mathbf{a} + \mathbf{a}^T \mathbf{K}_2 \mathbf{a} \geq 0$$

for all vectors $\mathbf{a} \in \mathbb{R}^n$ and the symmetry property can be shown trivially. All possible kernel matrices span a special subspace of $\mathbb{R}^{n \times n}$, referred to as the positive semi-definite cone and denoted as $\mathbb{R}^{n \times n}_{++}$. Although more rules to combine kernels exist (e.g. the product rule), the sum rule is the most common rule used in MKL.

Let $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ be a set of input data points associated with target (output) values $\mathbf{y} \in \mathbb{R}^n$. Let $\mathscr{X}_1, \mathscr{X}_2, ..., \mathscr{X}_p$ be inner product spaces, each containing $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ and endowed with respective inner product (kernel) functions $k_1, k_2, ...k_p$. The spaces $\mathscr{X}_1, \mathscr{X}_2, ..., \mathscr{X}_p$ are isomorphic and can contain different representations of the same set of objects. The kernels k_q are positive definite mappings from $\mathscr{X}_q \times \mathscr{X}_q$ to $\mathbb{R}$. In multiple kernel learning (MKL), the domain of f is the union (concatenation) of $\mathscr{X}_1, \mathscr{X}_2, ..., \mathscr{X}_p$ spaces. The functions in a MKL setting are often defined as linear combinations of base kernels, centered at the input points:

$$f = \sum_{q=1}^{p} \sum_{i=1}^{n} \mu_q \alpha_i k_q(\mathbf{x}_i, \cdot). \tag{5.10}$$

Here, the vector $\boldsymbol{\mu} \in \mathbb{R}^p$ represents the *kernel weights*, revealing the importance of each kernel for the problem at hand, while $\boldsymbol{\alpha}$ plays the same role as in single kernel regression. More generally, the weights of kernels and input points can be optimized jointly as a kernel-input point weight matrix $\mathbf{A} \in \mathbb{R}^{p \times n}$:

$$f = \sum_{q=1}^{p} \sum_{i=1}^{n} a_{qi} k_q(\mathbf{x}_i, \cdot).$$

Common MKL algorithms solve for $\boldsymbol{\mu}, \boldsymbol{\alpha}$ or both. In this work, we will examine the general form of different MKL algorithms and low-rank kernel matrix approximations. Importantly, we will highlight the differences from the resulting output function spaces induced by different algorithms.

5.3.2 *Multiple kernel learning algorithms*

The MKL algorithms optimize the kernel weights $\boldsymbol{\mu}$ to combine multiple kernel matrices into a single kernel matrix optimized for supervised learning. Often, it is assumed as a two-stage process, first optimizing the kernel matrix via $\boldsymbol{\mu}$ in 5.10 and later inferring the remaining model parameters, e.g. $\boldsymbol{\alpha}$ in Eq. 5.10. Below, we present baseline multiple kernel learning algorithms that assume the availability of full kernel matrices. A thorough review of MKL algorithms is presented in [31].

Algorithms based on centered alignment. The following algorithms proposed by Cortes et al. [75] are based on kernel centered alignment in a regression setting and differ in computational complexity and constraints on the solution $\boldsymbol{\mu}$. All are based on aligning the kernels with target values, when the space of kernel matrices is treated as a vector space $\mathbb{R}^{n \times n}_{++}$, defined above in Section 5.3.1. The subject to optimization in the following algorithms are the kernel weights $\boldsymbol{\mu} \in \mathbb{R}^p$, resulting in a combined kernel matrix

$$\mathbf{K}_\mu = \sum_{q=1}^{p} \mu_q \mathbf{K}_q. \tag{5.11}$$

We now introduce a number of tools upon which the algorithms are implemented. The *Frobenius product* is an inner product between two kernel matrices $\mathbf{K}_1$ and $\mathbf{K}_2$,

defined as:

$$\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F = \sum_i^n \sum_j^n \mathbf{K}_1(i,j)\mathbf{K}_2(i,j).$$

To compute the alignment, the feature map associated to the kernel is centered, i.e. the mapping associated to a data point $\mathbf{x}$ is scaled by the expectation $\phi_c(\mathbf{x}) = \phi(\mathbf{x}) - \mathbb{E}_{\mathbf{x}}[\phi(\mathbf{x})]$ where $\mathbf{x}$ is sampled from the empirical data distribution. In this case, the *centered kernel matrix* is obtained from any kernel matrix $\mathbf{K}$ by applying the following matrix function:

$$C(\mathbf{K}) = (\mathbf{I} - \frac{\mathbf{1}\mathbf{1}^T}{n})\mathbf{K}(\mathbf{I} - \frac{\mathbf{1}\mathbf{1}^T}{n}),$$

where $\mathbf{1}$ is a vector size n with all values equal to 1. The idempotent linear operator $\mathbf{P} = (\mathbf{I} - \frac{\mathbf{1}\mathbf{1}^T}{n})$ is the centering projection, from which follows:

$$\langle \mathbf{K}_1, C(\mathbf{K}_2) \rangle_F = \langle C(\mathbf{K}_1), \mathbf{K}_2 \rangle_F = \langle C(\mathbf{K}_1), C(\mathbf{K}_2) \rangle_F.$$

Finally, the *centered kernel alignment* is defined as the cosine of the angle between the kernel matrices:

$$\hat{\rho}(\mathbf{K}_1, \mathbf{K}_2) = \frac{\langle C(\mathbf{K}_1), C(\mathbf{K}_2) \rangle_F}{\|C(\mathbf{K}_1)\|_F \|C(\mathbf{K}_2)\|_F}$$

where the matrix norm $\|\cdot\|$ is the Frobenius norm defined in Eq. 2.1. Note that the definitions are consistent with the canonical inner product and the normalized projection (angle cosine) if $\mathbf{K}_1$ and $\mathbf{K}_2$ are unrolled into vectors.

Assume p kernel matrices $\{\mathbf{K}_1, \mathbf{K}_2, ..., \mathbf{K}_p\}$ and the target values $\mathbf{y} \in \mathbb{R}^n$. The associated *ideal kernel* is defined as $\mathbf{K}_y = \mathbf{y}\mathbf{y}^T$ and represents the assumed true underlying kernel. Arguably the simplest algorithm is the independent centered alignment (Align). The kernel weights $\boldsymbol{\mu}$ are defined as:

$$\mu_q = \hat{\rho}(\mathbf{K}_q, \mathbf{K}_y).$$

This algorithm is equivalent of p independent linear regressions (if kernel matrices are treated as one-dimensional vectors). Its principal advantages are simplicity and low computational complexity in the number of kernels. Therefore the choice is suitable for a small number of data points n and arbitrary number of kernels. Additionally, the optimization does not take into account dependencies between kernels, which is discussed next.

The bounded linear combination solves the optimization problem (Alignf):

$$\max_{\boldsymbol{\mu}\in\mathscr{M}} \quad \frac{\langle C(\mathbf{K}_\mu), \mathbf{K}_y\rangle_F}{\|C(\mathbf{K}_\mu)\|_F}, \tag{5.12}$$

where the feasible set is a hypersphere $\mathscr{M} = \{\boldsymbol{\mu} : \|\boldsymbol{\mu}\| = 1\}$ and $\boldsymbol{\mu}$ defines $\mathbf{K}_\mu$ as in Eq. 5.11. Let the matrix $\mathbf{M}$ denote the inner products between the kernels, $m_{ij} = \langle C(\mathbf{K}_i), C(\mathbf{K}_j)\rangle_F$ and the vector $\mathbf{a}$ the inner product with the ideal kernel $a_q = \langle C(\mathbf{K}_q), \mathbf{K}_y\rangle_F$. Then the solution to the problem in Eq. 5.12 is given by $\boldsymbol{\mu}^\star = \frac{\mathbf{M}^{-1}\mathbf{a}}{\|\mathbf{M}^{-1}\mathbf{a}\|}$. The solution admits a simple interpretation; it is the solution of a linear regression in a vector space of kernels which is subsequently scaled. The practical role of scaling is for interpretation purposes, as the primary goal of MKL is to learn a combined kernel and not a predictive model. Note that in this case the weighted sum $\sum_q \mu_q \mathbf{K}_q$ is not guaranteed to result in a positive semi-definite kernel matrix.

Finally, to ensure a valid kernel matrix and improve interpretation, $\boldsymbol{\mu}$ can be limited to a convex subset, $\mathscr{M} = \{\boldsymbol{\mu} : \sum_q \mu_q = 1 \text{ and } \mu_q \geq 0\}$ — a half-sphere. The solution involves solving a linearly constrained quadratic problem (QP) and no longer assumes a closed-form solution (Alignfc):

$$\min_{\mathbf{v}\geq 0} \quad \mathbf{v}^T\mathbf{M}\mathbf{v} - 2\mathbf{v}^T\mathbf{a} \tag{5.13}$$

and the optimal solution $\boldsymbol{\mu}^\star = \frac{\mathbf{v}^\star}{\|\mathbf{v}^\star\|}$. Both solutions to Eq. 5.12-5.13 require at least quadratic computational complexity in the number of kernels and are infeasible for problems with large number of kernels, but might be preferred due to interpretability.

Simultaneous solution to MKL and linear regression. A natural extension of MKL algorithms is to simultaneously solve for the kernel weights and other model parameters, as the two sets of parameters are often dependent. One such case that builds on Ridge regression and MKL is the L_2-regularized Kernel Ridge Regression (L2-KRR) model [76]. Recalling the form of the primal Ridge regression,

$$\min_{\mathbf{w}} \quad \|\mathbf{w}\| + c\sum_{i=1}^{n}(\mathbf{w}^T\phi(\mathbf{x}_i) - y_i)^2,$$

where c is the trade-off parameter, the dual form can be obtained by replacing the inner products of feature mappings $\phi(\mathbf{x}_i)$ with the kernel matrix:

$$\max_{\boldsymbol{\alpha}} \quad -\lambda \boldsymbol{\alpha}^T \boldsymbol{\alpha} - \boldsymbol{\alpha}^T (\mathbf{K}\boldsymbol{\alpha} - 2\mathbf{y}), \tag{5.14}$$

where the left-hand term is the model capacity and the right-hand term represents the data fitting term. When the kernel matrix $\mathbf{K}$ is replaced with a sum of base kernels $\mathbf{K}_\mu = \sum_{q=1}^{p} \mathbf{K}_q$, the problem can be stated as:

$$\min_{\boldsymbol{\mu} \in \mathscr{M}} \max_{\boldsymbol{\alpha}} \quad -\lambda \boldsymbol{\alpha}^T \boldsymbol{\alpha} - \sum_{q=1}^{p} \mu_q \boldsymbol{\alpha}^T \mathbf{K}_q \boldsymbol{\alpha} + 2\boldsymbol{\alpha}^T \mathbf{y}, \tag{5.15}$$

where the feasible set $\mathscr{M} = \{\boldsymbol{\mu} \geq \mathbf{0}$ and $\|\boldsymbol{\mu} - \boldsymbol{\mu}_0\|\}$ and $\boldsymbol{\mu}_0$ is a generalization of bounding the norm of $\boldsymbol{\mu}$.

The optimization problem in Eq. 5.14 is deliberately posed as a maximization instead of equivalent minimization, due to the consequent min-max form in Eq. 5.15. Applying the von Neumann generalized minimax theorem [77], the problem in Eq. 5.15 can be stated analogously as:

$$\max_{\boldsymbol{\alpha}} \; -\lambda \boldsymbol{\alpha}^T \boldsymbol{\alpha} + 2\boldsymbol{\alpha}^T \mathbf{y} \quad + \quad \min_{\boldsymbol{\mu} \in \mathscr{M}} \; -\boldsymbol{\mu}^T \mathbf{v},$$

with $\mathbf{v} = (\boldsymbol{\alpha}^T \mathbf{K}_1 \boldsymbol{\alpha}, \boldsymbol{\alpha}^T \mathbf{K}_2 \boldsymbol{\alpha}, \ldots \boldsymbol{\alpha}^T \mathbf{K}_p \boldsymbol{\alpha})$. This results in a convex optimization problem as the point-wise maximum preserves convexity. Similarly as to Eq. 5.13, the problem admits a closed-form solution $\boldsymbol{\mu} = \boldsymbol{\mu}_0 + \frac{\mathbf{v}}{\|\mathbf{v}\|}$. The regression weights are then obtained as the solution to the canonical dual Ridge regression problem, $\boldsymbol{\alpha} = (\mathbf{K}_\mu + \lambda \mathbf{I})^{-1} \mathbf{y}$. Now, as the two parts of the solution are mutually dependent, the authors propose a simple iterative algorithm that alternatively solves for $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$, in accordance with the solutions stated above.

5.4 Gaussian processes

Gaussian processes (GP) are a common name for non-parametric Bayesian regression [72]. An advantage over kernel Ridge regression, presented in Section 5.2 is a principled way to treat uncertainty in the model, which depends on the location of the inputs and the assumed covariance structure. A definition of a GP follows.

Multivariate normal distribution. The domain of a multivariate normal distribution $\mathscr{N}$ are vectors in an real vector space of finite dimension $\mathbb{R}^n$. It is parametrized by two sets of parameters: the mean vector $\mathbf{m} \in \mathbb{R}^n$ and a positive semi-definite covariance

matrix $\mathbf{K} \in \mathbb{R}^{n \times n}_{++}$. Any kernel is a valid covariance function and can be used to compute $\mathbf{K}$.

Gaussian process. A Gaussian Process is set of real number variables, any finite number of which are jointly distributed according to a multivariate normal (Gaussian) distribution.

As before, let $\mathscr{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ be a set of input data points associated with target (output) values $\mathbf{y} \in \mathbb{R}^n$. A kernel function k is used in place of the inner product and determines a kernel matrix $\mathbf{K}$. In a GP setting, a *latent* function $f(\mathbf{x})$ with covariance function k is assumed. A finite sample of values of f corresponding to $\mathscr{D}$ are represented as $\mathbf{f} \in \mathbb{R}^n$ and distributed according to a GP:

$$p(\mathbf{f}|\mathscr{D}) = \mathscr{N}(\mathbf{0}, \mathbf{K}),$$

where we assume $\mathbf{m} = \mathbf{0}$ for later convenience. In practice, this assumption means the target data is centered. Given the latent values $\mathbf{f}$, the actual observed values $\mathbf{y}$ will be distributed according to another GP:

$$\begin{aligned} p(\mathbf{y}|\mathbf{f}) &= \mathscr{N}(\mathbf{f}, \sigma^2\mathbf{I}), \\ p(\mathbf{y}|\mathscr{D}) &= \mathscr{N}(\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I}). \end{aligned} \tag{5.16}$$

In other words, the variables $\mathbf{y}$ are independent given $\mathbf{f}$ and associated with noise variance governed by σ^2. Here, σ^2 plays a similar role as the regularization parameter in Kernel Ridge Regression. In practice, one is interested in predicting the value y_* for an arbitrary element of the original input space $\mathbf{x}_*$. The predictive distribution is inferred using Bayesian inference and basic linear algebra:

$$\begin{aligned} p(y_*|\mathbf{x}_*, \mathscr{D}, \mathbf{y}) &= \mathscr{N}(\mu_*, \sigma_*^2) \\ &= \mathscr{N}(\mathbf{k}_*^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y},\ k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_* + \sigma^2), \end{aligned}$$

where the vector $\mathbf{k}_*$ represents the evaluation of the kernel function between the new input point $\mathbf{x}_*$ and the training data $\mathbf{k}_* = (k(\mathbf{x}_1, \mathbf{x}_*), k(\mathbf{x}_2, \mathbf{x}_*), ..., k(\mathbf{x}_n, \mathbf{x}_*))$. The input data is again accessed solely through the evaluations of the kernel and the main computational burdens are the computation and inversion of the kernel matrix with complexities $O(n^2)$ and $O(n^3)$, respectively.

Samples from Gaussian Processes where the covariance structure is defined by different kernels is shown on Figure 5.2. Note how different hyperparameters influence the shape of the sampled functions f. For instance, in the case of exponentiated-quadratic covariance, the parameter γ governs the length scale, which can be interpreted as the frequency content of the signal. The parameter ν of the Matérn kernel induces jagged functions for small values and increasingly smooth functions for large values. Finally, for the polynomial kernel the degree parameter determines the degree of the polynomials.

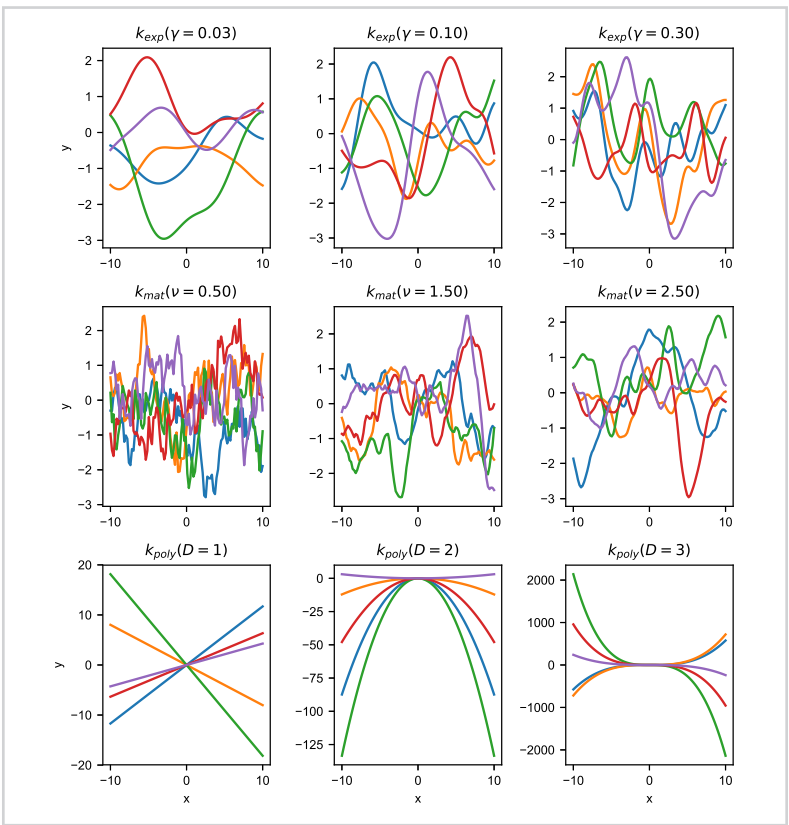


Figure 5.2

Samples from Gaussian processes with covariance structure defined by exponentiated-quadratic, Matérn and polynomial kernel for different values of hyperparameters. With polynomial kernel of degree $D = 1$, the linear kernel is recovered. The noise parameter σ^2 is set to 0, so the observed values $\mathbf{y}$ are equal to latent function values $\mathbf{f}$.

5.5 Kernel matrix approximations

The kernel-based models and related algorithms are naturally bounded by the quadratic complexity in the number of data points and usually assume the computation of the kernel matrix. This property is severely limiting the applicability of kernel methods on large datasets. The methods in approximate kernel learning can very broadly be classified in *approximations of the kernel function* and *approximation of the kernel matrix* and share a common goal in achieving linear complexity in the number of data points.

In this section, we assume a data set of n data points $\mathscr{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$. A kernel k generates the kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$. Note that in order to achieve sub-quadratic complexity, $\mathbf{K}$ should not be computed in its entirety. Instead, different approximation schemes compute an approximation to $\mathbf{K}$, commonly preserving properties related to the actual machine learning task. By definition, $\mathbf{K}$ is a symmetric, positive semi-definite matrix. Consequently, there exist infinitely many factors $\mathbf{G}$ of $\mathbf{K}$, i.e. $\mathbf{K} = \mathbf{G}\mathbf{G}^T$. Approximations to the kernel matrix typically involve a low-rank matrix $\mathbf{G} \in \mathbb{R}^{n \times r}$ where the rank $r < n$ is usually chosen based on practical constraints.

The following methods are based on a subset of the original data points, referred to as the *active set* (inducing set, inducing inputs, pivots):

$$\mathscr{A} \subset \{1, 2, ..., n\} \text{ and } |\mathscr{A}| = r.$$

The selection of $\mathscr{A}$ critically influences both the approximation accuracy and performance of the downstream models. It is important to note that the inducing points need not be exact instances selected from $\mathscr{D}$, but can be arbitrary points in the corresponding space. The methods able to optimize inducing points over the whole domain pose additional assumption on the kernels and are not the treated later in the text.

The methods below will result in the same approximation $\mathbf{L} = \mathbf{G}\mathbf{G}^T$ for the same set $\mathscr{A}$ and differ only in the way inducing points are selected. This is guaranteed by the lemma on the uniqueness of the kernel approximation for a fixed active set:

Proposition. There exists a unique matrix $\mathbf{L}$ that is *(i) symmetric, (ii) has the column space spanned by* $\mathbf{K}(:, \mathscr{A})$ *and (iii)* $\mathbf{L}(:, \mathscr{A}) = \mathbf{K}(:, \mathscr{A})$. The proof can be found in Bach and Jordan [34], Proposition 1.

The Nyström method. In principle, the Nyström method provides a basic kernel approximation without specifying the way of selecting the active set $\mathscr{A}$ (Rasmussen

[72], ch. 8). Given a fixed $\mathscr{A}$ the kernel matrix approximation $\mathbf{L}$ is obtained as:

$$\mathbf{L} = \mathbf{K}(:, \mathscr{A})\mathbf{K}(\mathscr{A}, \mathscr{A})^{-1}\mathbf{K}(:, \mathscr{A}). \tag{5.17}$$

Here, $\mathbf{L}$ represents a projection of values of the kernel to the subspace spanned by the vectors in the $n \times r$ submatrix $\mathbf{K}(:, \mathscr{A})$, with the active set submatrix approximated exactly $\mathbf{L}(\mathscr{A}, \mathscr{A}) = \mathbf{K}(\mathscr{A}, \mathscr{A})$. The active set $\mathscr{A}$ can be either selected randomly or optimized with respect to the target task.

Recent developments select $\mathscr{A}$ based on leverage scores [78]. The leverage scores $\boldsymbol{\ell} \in \mathbb{R}^n$ provide an importance sampling distribution for creating random low-rank approximations (sketches) of $\mathbf{K}$:

$$\boldsymbol{\ell}(\lambda) = \operatorname{diag}(\mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1}).$$

This definition can be interpreted as follows: as $\lambda \to 0$, the scores are equal for all i. Otherwise, as $\lambda \to \infty$, the leverage scores are proportional to $\mathbf{K}(i, i) = k(\mathbf{x}_i, \mathbf{x}_i) = \|\phi(\mathbf{x}_i)\|^2$ where ϕ is a mapping to the Hilbert space defined in Section 5.1. In other words, the points are scored in proportion to the norm in the implicit inner product space induced by the kernel. The leverage scores are approximated via another random sketch of $\mathbf{K}$, which is selected such that $\mathbf{L} = \mathbf{G}\mathbf{G}^T$ and the active set is selected randomly,

$$\hat{\boldsymbol{\ell}}(\lambda) = \operatorname{diag}(\mathbf{G}(\mathbf{G}^T\mathbf{G} + n\lambda\mathbf{I})^{-1}\mathbf{G}^T). \tag{5.18}$$

It can be observed that the applicability of $\boldsymbol{\ell}$ is limited to inner product spaces where points can have different norms (e.g. induced by linear or polynomial kernel), while they will be equal for exponentiated-quadratic kernel as $k_{\exp}(\mathbf{x}_i, \mathbf{x}_i) = 1$ for all $\mathbf{x}_i$. Additionally, the approximate leverage scores will be biased towards points that were selected for the initial matrix sketch $\mathbf{G}$. Many alternative selection methods exist and are based on approximating the distribution of inducing points. Recently, it was suggested that inducing points can be initialized via the Kmeans++ initialization algorithm proposed by Oglic and Gärtner [79], where the inducing points need not be in $\mathscr{D}$. Empirically, this method yielded very similar results to the approximate leverage scores and improved performance on three out of five datasets for a moderate increase in time (see ref [79]). Next, we present two iterative, greedy approaches based on an alternative matrix decomposition.

Incomplete Cholesky decomposition (ICD). A natural iterative approach to compute approximations to symmetric semi-definite matrices is the Cholesky decomposition, which computes a triangular matrix approximation to $\mathbf{K}$ and therefore presents an efficient way to solve linear systems. The Incomplete Cholesky decomposition employs an efficient selection of pivot columns and stops after a predetermined number of iterations, which is equal to the target approximation rank.

A kernel matrix $\mathbf{K}$ is approximated with a low-rank Cholesky factor $\mathbf{G}$. The ICD is a family of methods that produce a finite sequence of matrices $\mathbf{G}_1, \mathbf{G}_2, ..., \mathbf{G}_r$, such that $\mathbf{G}_r\mathbf{G}_r^T \rightarrow \mathbf{K}$ as $r \rightarrow n$

Initially, $\mathbf{G}$ is set to $\mathbf{0}$. A diagonal vector representing the lower-bound on the approximation gain is initialized as $\mathbf{d} = \text{diag}(\mathbf{K})$ and the active set $\mathscr{A} = \emptyset$, keeping track of selected pivot columns. At iteration j, a pivot i is selected from the remaining set $\mathscr{J} = \{1, 2, ..., n\} \setminus \mathscr{A}$ and its pivot column $\mathbf{G}(:, j) \leftarrow \mathbf{g}_i$ is computed as

$$\begin{aligned} \mathbf{G}(i, j) &= \sqrt{\mathbf{d}(i)}, \\ \mathbf{G}(\mathscr{J}, j) &= \frac{1}{\mathbf{G}(i, j)}\Big(\mathbf{K}(\mathscr{J}, i) - \sum_{t=1}^{j-1} \mathbf{G}(\mathscr{J}, t)\mathbf{G}(i, t)\Big). \end{aligned} \tag{5.19}$$

A pivot i is selected in each iteration according to the maximal value in $\mathbf{d}$. The pivot columns $\mathbf{G}(:, j)$ can also be seen as evaluations of *basis functions* when treated as a function $k(\mathbf{x}_i, \cdot)$ with fixed $\mathbf{x}_i$. Importantly, only the information on one column of $\mathbf{K}$ is required at each iteration and $\mathbf{G}\mathbf{G}^T$ need never be computed explicitly. The selected pivot is added to the active set, and the counter j and the diagonal vector are updated:

$$\begin{aligned} \mathbf{d} &\leftarrow \mathbf{d} - \mathbf{g}_i^2, \\ \mathscr{A} &\leftarrow \mathscr{A} \cup \{i\}, \\ j &\leftarrow j + 1. \end{aligned}$$

The selection of pivots can be optimized further in scenarios with a target task, e.g. linear regression. This type of improvement is presented next.

Cholesky with side-information (CSI). A supervised version of the ICD method, where the corresponding target values $\mathbf{y} \in \mathbb{R}^n$ are assumed [34]. The method is based

on the ICD, except for the way pivot columns are selected. The global objective function J to be minimized is defined as:

$$J(\mathbf{G}) = \kappa_1 \|\mathbf{K} - \mathbf{G}\mathbf{G}^T\|_1 + \kappa_2 \min_{\mathbf{w}\in\mathbb{R}^r} \|\mathbf{y} - \mathbf{G}\mathbf{w}\|.$$

The gradient of J is naturally split into two parts, the gain with respect to the approximation error (computed exactly as in ICD) and the gain with respect to the target values $\mathbf{y}$, where the trade-off is controlled by the parameters κ_1 and κ_2. The gains are computed efficiently with look-ahead decompositions, where an additional sketch of the kernel matrix is used to evaluate the gains of each potential pivot column.

The principal advantage of kernel matrix approximations is the independence of any particular kernel function as the data is always accessed only through the values of the kernel. Hence, these are the most general methods applicable to any valid kernel. In the same vein, the resulting approximations are not always optimal as a predefined set of kernels and hyperparameters must be specified.

5.6 Kernel-specific approximations

Since the primary interest of this work is the approximation of the *kernel matrix*, we present details of approximation of kernel functions in Appendix B.3.3. Briefly, works in kernel-specific approximation tend to focus on translation-invariant kernels, that can be written as a function of one argument (e.g. the difference between data points). A natural idea then is to generate an explicit form of feature map with the kernel equal to the corresponding inner product. This can be achieved with using the basis function decomposition, such as the Fourier transform.

6

Approximate multiple kernel learning

The related work in kernel learning, presented in Chapter 5 and Appendix B.3.3, seldom focuses on simultaneous approximation of multiple kernel matrices. In particular, the related work does not address the following concerns, which we address with the Mklaren algorithm.

- The kernel matrix approximation methods, such as Incomplete Cholesky Decomposition, Cholesky with Side Information or the Nyström method, allow for a custom basis function selection criterion for one kernel, but the existing criteria do not address a multiple kernel scenario.
- The range (column space) of a sum of the kernel matrices is equivalent to the range of the concatenation of their respective implicit feature spaces. However, it is easy to show that this does not hold if a kernel matrix approximation based on inducing points is used. This means that multiple kernel-aware methods are needed.
- The feature selection methods in classic linear regression assume access to all possible basis functions. This is not feasible in approximate kernel learning, as the former aims to achieve sub-quadratic complexity. A trade-off between optimality and computation is required.
- The currently most time-efficient methods are approximations of certain classes of kernel functions. Significant limitations are posed on both the kernels and input data (e.g. translation invariance and continuous input spaces). This limits applications, for example in bioinformatics using string kernels.

In this section, we present a kernel matrix factorization algorithm, where the input data are accessed only through the kernels, yielding multiple kernel matrices. The central idea builds on orthogonality in the Least-angle regression (LAR; Appendix B.2.2), which is used to select the basis functions in an attempt to maximize the predictive accuracy of the resulting model. Again, the principal motivation to develop a simultaneous approximation of multiple kernel matrices is the increased model capacity compared to classic matrix factorization algorithms.

We start by presenting the design and the technical details of the Mklaren model and algorithm, which rely on Incomplete Cholesky Decomposition (Section 5.5), multiple kernel learning (Section 5.3) and Least-angle regression. Technical derivations

cover out-of-sample prediction (Section 6.6), model interpretation using the relation between primal and dual regression coefficients (Section 6.7), L_2 norm regularization (Section 6.8), and computational complexity (Section 6.9). Finally, we explain the relation between multiple kernel learning and kernel matrix approximation (Section 6.10).

6.1 Initial definitions and overview

Let $\mathscr{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ be a set of input data points associated with target values $\mathbf{y} \in \mathbb{R}^n$. Let $\mathscr{X}_1, \mathscr{X}_2, ..., \mathscr{X}_p$ be inner product spaces, each containing $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ and endowed with respective kernel functions $k_1, k_2, ...k_p$. The spaces $\mathscr{X}_1, \mathscr{X}_2, ..., \mathscr{X}_p$ can contain different representations of the same set of objects (e.g. real vector spaces, discrete vector spaces, discrete character strings, etc.). The kernels k_q are positive definite maps from $\mathscr{X}_q \times \mathscr{X}_q$ to $\mathbb{R}$. The role of kernels in regression is to determine the covariance structure of the output functions $f(\mathbf{x})$ used to approximate $\mathbf{y}$ given the inputs. In multiple kernel learning (MKL), the domain of f is the union of $\mathscr{X}_1, \mathscr{X}_2, ..., \mathscr{X}_p$ input spaces.

Evaluating $k_q(\mathbf{x}_i, \mathbf{x}_j)$ for each pair $\mathbf{x}_i, \mathbf{x}_j$ in the training set determines kernel matrices $\mathbf{K}_q \in \mathbb{R}^{n \times n}$. The goal of a kernel matrix approximation algorithm is to construct the matrices $\mathbf{G}_1, \mathbf{G}_2, ..., \mathbf{G}_p$, where $\mathbf{G}_q \in \mathbb{R}^{n \times r_q}$ and the total rank $r = \sum_q r_q < n$. The given target values or any other additional information can be exploited to focus on the parts of the input space that are relevant to the task at hand.

The Mklaren algorithm simultaneously learns low-rank approximations of kernel matrices $\mathbf{K}_q$ associated to kernels k_q based on active sets $\mathscr{A}_q$ — subsets of training points of cardinality $|\mathscr{A}_q| = r_q$. Incomplete Cholesky decomposition (ICD) is used to selectively update some of the $\mathbf{G}_q$ and the regression estimate $\mathbf{f} \in \mathbb{R}^n$. At each step, a kernel k_q and an inducing point (pivot) $\mathbf{x}_i$ are chosen based on a heuristic that evaluates the explained information on the current residual (error) $\mathbf{e} = \mathbf{y} - \mathbf{f}$. This is achieved with the least-angle regression strategy in the space spanned by the current approximations $\mathbf{G}_q$ and computed in time $O(nr^2)$ per step. Finally, the coefficients of a general function estimator $\hat{f}(\mathbf{x})$ are inferred to predict the output of any point in the input space.

The high-level pseudo code of the Mklaren algorithm is given in Algorithm 3, and its steps are described in detail in the following subsections. Further implications of simultaneous approximation of multiple kernels are discussed in Section 6.10.

6.2 Simultaneous Incomplete Cholesky decompositions

Recall the description of Incomplete Cholesky decomposition (ICD) of a single kernel matrix 5.5. In case of p kernels, each kernel function k_q, $q \in 1, 2, ..., p$ determines a corresponding $\mathbf{K}_q$, which is approximated with Cholesky factors $\mathbf{G}_q$. An example scenario is depicted in Fig. 6.1. The set of all Cholesky factors $\mathbf{G}_q$ is used to construct a *combined feature matrix* as follows. In each step, we assume the existence of a regression estimate $\mathbf{f} \in \mathbb{R}^n$, initially set to $\mathbf{0}$, and the residual $\mathbf{e} = \mathbf{y} - \mathbf{f}$.

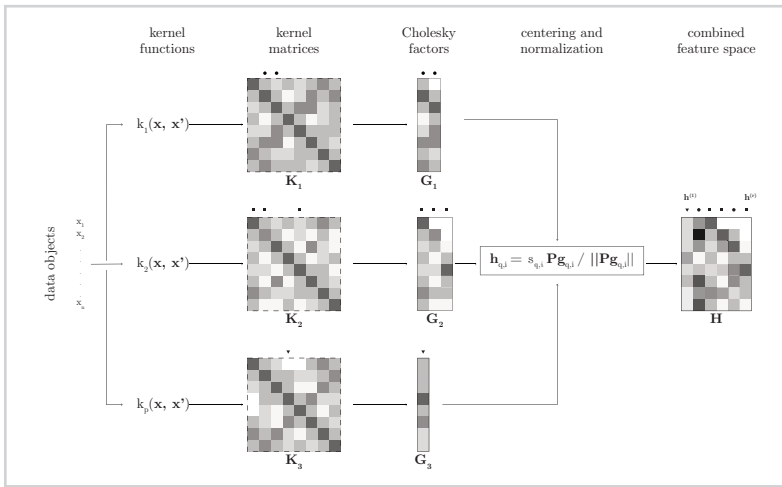


Figure 6.1

Overview of variables included in the hypothetical model using three kernels, $q \in \{1, 2, 3\}$. Kernel matrices in dashed values are never computed explicitly. The markers *circle*, *rectangle* and *triangle* represent the selected pivot columns for kernels 1, 2 and 3, respectively.

As least-angle regression is defined for centered and scaled data, this condition has to be ensured for the basis functions in out approximation. For each selected pivot column $\mathbf{g}_{q,i}$ for kernel k_q, define the centering and scaling transformation:

$$\mathbf{h}_{q,i} \leftarrow s_{q,i}\mathbf{P}\mathbf{g}_{q,i}/\|\mathbf{P}\mathbf{g}_{q,i}\|, \tag{6.1}$$

where the operator $\mathbf{P}$ is the centering projection $\mathbf{P} = (\mathbf{I} - \frac{\mathbf{1}\mathbf{1}^T}{n})$ and $s_{q,i}$ is the sign of the correlation $(\mathbf{P}\mathbf{g}_i)^T\mathbf{e}$. Each $\mathbf{h}_{q,i}$ is normalized and makes an angle of at most 90 degrees with the residual $\mathbf{e}$. The set of columns $\mathbf{h}_{q,i}$ span the *combined feature space*, equivalent to any matrix $\mathbf{H} \in \mathbb{R}^{n \times r}$ containing this same set of columns (in any order):

$$\text{span}(\mathbf{H}) = \text{span}\Big(\{\mathbf{h}_{1,1}, \mathbf{h}_{1,2}..., \mathbf{h}_{1,r_1}, \mathbf{h}_{2,1}, ..., \mathbf{h}_{2,r_2}, \mathbf{h}_{p,1}, ..., \mathbf{h}_{p,r_p}\}\Big). \tag{6.2}$$

Applying the operator $\mathbf{P}$ is equivalent to centering the positive semi-definite matrix $\mathbf{H}\mathbf{H}^T$ — the approximation of the *combined Gram matrix* — and is used for selection of new pivot columns independent of their respective length.

Least-angle regression is used to iteratively select the next pivot column and thus determine $\mathbf{H}$ and update the regression estimate $\mathbf{f}$. The next kernel q and pivot j are selected from all remaining sets $\mathcal{J}_q$, based on the current residual $\mathbf{e} = \mathbf{y} - \mathbf{f}$. The corresponding pivot column $\mathbf{g}_{q,i}$ is computed using the Cholesky step in Eq. 5.19 and added to $\mathbf{G}_q$. At any iteration, each $\mathbf{G}_q$ may contain a different number of columns r_q as the selection depends on the relevance of k_q for explaining the residual.

6.3 Pivot selection based on Least-angle regression

Least-angle regression (LAR) is an active set-based feature selection in linear regression (see Appendix and Efron and Hastie [36] for a thorough description). Here, we propose an idea based on the LAR column selection to determine the next pivot column to be added to any of the $\mathbf{G}_q$ and consequently to combined feature matrix $\mathbf{H}$.

The original LAR method assumes availability of all variables representing the covariates (feature vectors) in the sample data matrix. In our case, however, this matrix is $\mathbf{H}$ and is constructed iteratively. The adaptation of the LAR-based column selection is non-trivial since the exact values of the new columns $\mathbf{g}_{q,i}$ and $\mathbf{h}_{q,i}$ are *unknown at the time of selection*. This section describes a method to construct $\mathbf{H}$ given the columns $\mathbf{h}_{q,i}$ and learn $\mathbf{f} \in \text{span}(\mathbf{H})$. In favor of clarity we assume (only in this section) that the values of all $\mathbf{h}_{q,i}$ are known and describe the ordering of $\mathbf{h}_{q,i}$ in $\mathbf{H}$. The problem of unknown candidate pivot column values is postponed to Section 6.4.

The matrix $\mathbf{H}$ is initialized to $\mathbf{0}$. The regression estimate $\mathbf{f}$ and the residual $\mathbf{e}$ are initialized as

$$\mathbf{f} = \mathbf{0} \text{ and } \mathbf{e} = \mathbf{y}, \text{ assuming w.l.g. } \mathbf{1}^T\mathbf{y} = 0. \tag{6.3}$$

By construction, $\|\mathbf{h}_{q,i}\| = 1$ and $\mathbf{1}^T\mathbf{h}_{q,i} = 0$ for all q, i. The $\mathbf{h}_{q,i}$ will be added to $\mathbf{H}$ in a defined order

$$\mathbf{H}(:,t) \leftarrow \mathbf{h}_{q,i} = \mathbf{h}^{(t)} \text{ for } t = 1, 2, ..., r,$$

where a unique kernel, pivot pair q, i is selected for each position t. The ordering depends on the correlation with the residual $c_t = \mathbf{e}^T\mathbf{h}^{(t)}$. Therefore, Cholesky factors $\mathbf{G}_q$ containing pivot columns with more information on the current residual are preferably selected.

The column selection procedure is depicted in Fig. 6.2a and is defined as follows. At iteration $t = 1$ the first vector is chosen to maximize correlation

$$\max_{\{m\in 1,2,\ldots,r\}} \quad c_m = \mathbf{e}^T\mathbf{h}^{(m)}.$$

This $\mathbf{h}^{(m)}$ is then added to $\mathbf{H}(:,1) = \mathbf{h}^{(m)}$. At each iteration t, $\mathbf{H}$ contains t columns. There exist the *bisector* $\mathbf{u}$, having $\|\mathbf{u}\| = 1$ and making equal angles, less than 90 degrees, between the residual $\mathbf{e}$ and vectors currently in $\mathbf{H}$. Updating the regression estimate $\mathbf{f}$ along direction $\mathbf{u}$ and the residual $\mathbf{e}$:

$$\mathbf{f}^{\text{new}} = \mathbf{f} + \gamma\mathbf{u}, \qquad \mathbf{e}^{\text{new}} = \mathbf{e} - \gamma\mathbf{u}, \tag{6.4}$$

causes the correlations $c_t = \mathbf{e}^T\mathbf{h}^{(t)}$ to change equally for all $\mathbf{h}^{(t)}$ in $\mathbf{H}$, for an arbitrary step size $\gamma \in \mathbb{R}$. The value γ is set such that some new column $\mathbf{h}^{(t+1)}$ not in $\mathbf{H}$ will have the same correlation with $\mathbf{e}^{\text{new}}$ as all the columns already in $\mathbf{H}$:

$$\measuredangle(\mathbf{e}^{\text{new}}, \mathbf{h}^{(1)}) = \ldots = \measuredangle(\mathbf{e}^{\text{new}}, \mathbf{h}^{(t)}) = \measuredangle(\mathbf{e}^{\text{new}}, \mathbf{h}^{(t+1)}).$$

The step size γ and $\mathbf{h}^{(t+1)}$ are selected as follows. Define the following quantities

$$C = \max_{\{t|h^{(t)}\in\mathbf{H}\}} \quad c_t \qquad\qquad A = (\mathbf{1}^T\mathbf{H}\mathbf{1})^{-1/2}.$$

Then,

$$\begin{aligned} \gamma &= \min^{+}_{\{m|\mathbf{h}^{(m)}\notin\mathbf{H}\}} \quad \left\{\frac{C - c_m}{A - a_m}, \frac{C + c_m}{A + a_m}\right\}, \text{where} \\ c_m &= \mathbf{e}^T\mathbf{h}^{(m)} \\ a_m &= \mathbf{u}^T\mathbf{h}^{(m)}. \end{aligned} \tag{6.5}$$

Here, $\min^+$ is the minimum over positive arguments for each choice of m. The selected column vector $\mathbf{h}^{(m)}$ is the minimizer of Eq. 6.5 and is inserted at $\mathbf{H}(:, t+1) = \mathbf{h}^{(m)}$. For the last column vector (as there are no further column vectors to chose from) the step size simplifies to $\gamma = C/A$, yielding the least-squares solution for $\mathbf{H}$ and $\mathbf{y}$.

The mentioned problem in our case is that the exact values of all potential pivot columns $\mathbf{g}_{q,i}$ not in $\mathbf{G}_q$ and its corresponding $\mathbf{h}_{q,i}$ are unknown. Explicit calculation of all columns using the Cholesky step in Eq. 5.19 would yield quadratic computational complexity in n, as their values depend on all previously selected pivots. The issue is addressed by using approximations $\hat{\mathbf{g}}_{q,i}$ and $\hat{\mathbf{h}}_{q,i}$ that are less expensive to compute, as described in the following section.

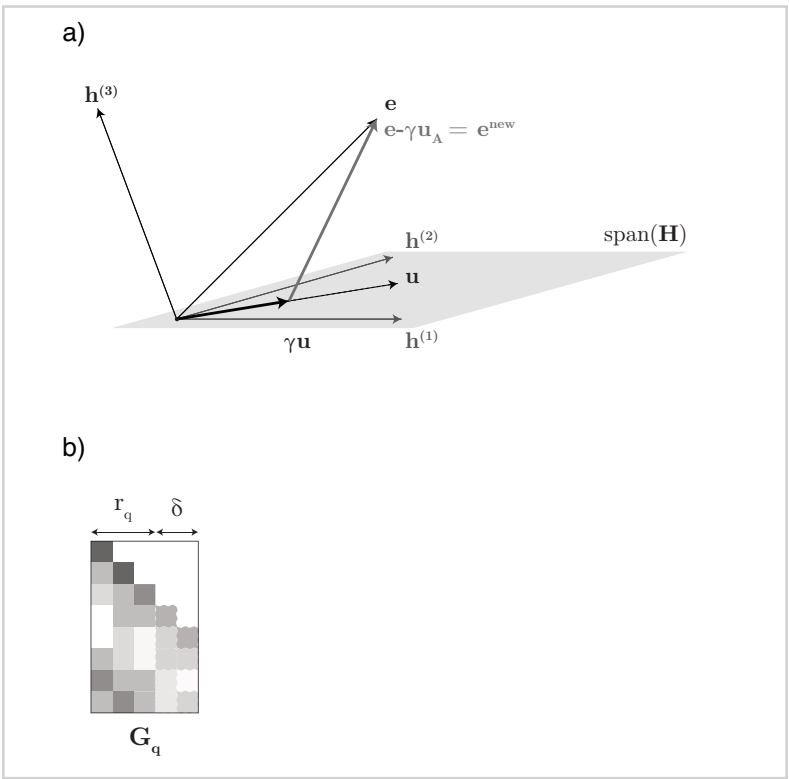


Figure 6.2

a) Updating the regression estimate within the combined feature matrix $\mathbf{H}$ containing two vectors $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$. The residual is $\mathbf{e} = \mathbf{y} - \mathbf{f}$, where $\mathbf{f} \in \text{span}\{\mathbf{h}^{(1)}\}$ and $\measuredangle(\mathbf{e}, \mathbf{h}^{(2)}) = \measuredangle(\mathbf{e}, \mathbf{h}^{(1)})$. The new residual $\mathbf{e}^{\text{new}}$ upon selection of $\mathbf{h}^{(2)}$ is obtained by adding $\gamma\mathbf{u}$ to $\mathbf{f}$ and updating $\mathbf{e}$ accordingly. The step size γ is increased until some new vector $\mathbf{h}^{(3)}$ has the same correlation (angle) with $\mathbf{e}^{\text{new}}$ as both $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$, i.e., $\measuredangle(\mathbf{e}^{\text{new}}, \mathbf{h}^{(3)}) = \measuredangle(\mathbf{e}^{\text{new}}, \mathbf{h}^{(2)}) = \measuredangle(\mathbf{e}^{\text{new}}, \mathbf{h}^{(1)})$. b) Schematic representation of selected r_q pivot columns and δ look-ahead columns.

6.4 Look-ahead decompositions

The selection of a new column vector $\mathbf{h}^{(m)}$ to be added to the combined feature matrix $\mathbf{H}$ and its corresponding $\mathbf{h}_{q,i}$, $\mathbf{g}_{q,i}$ is based only on the values a_m, c_m in Eq. 6.5.

Instead of explicitly calculating each candidate $\mathbf{g}_{q,i}$ for all q, i at each iteration, we use an approximate column vector $\hat{\mathbf{g}}_{q,i}$. The approach uses a similar idea to look-ahead columns and is based on selecting δ look-ahead pivot columns, spanning a *look-ahead approximation* to the full kernel matrix [34, 80]. As it will be clear from the description below, increasing δ monotonically increases the (look-ahead) approximation accuracy and is limited only by the available computational resources. See Section 6.9 for more discussion.

By definition of ICD in Eq. 5.19, the values of a candidate pivot column $\mathbf{g}_{q,i}$ at step $j = r_q$ and pivot $i \notin \mathscr{A}_q$ are:

$$\mathbf{g}_{q,i} = \frac{(\mathbf{K}_q - \sum_{t=1}^{r_q-1} \mathbf{G}_q(:,t)\mathbf{G}_q(:,t)^T)(:,i)}{\sqrt{\mathbf{d}_q(i)}}.$$

The main computational cost is the computation of a rank n kernel matrix $\mathbf{K}_q$. Instead, δ look-ahead columns define a look-ahead approximation $\mathbf{L}_q = \mathbf{G}_q(:, r_q + \delta)\mathbf{G}_q(:, r_q + \delta)^T$, depicted in Fig. 6.2b. This defines approximate values $\hat{\mathbf{g}}_{q,i}$:

$$\begin{aligned}\hat{\mathbf{g}}_{q,i} &= \frac{(\mathbf{L}_q - \sum_{t=1}^{r_q-1} \mathbf{G}_q(:,t)\mathbf{G}_q(:,t)^T)(:,i)}{\sqrt{\mathbf{d}_q(i)}} = \\ &= \frac{\mathbf{G}_q(:,\ r_q{+}1{:}r_q{+}\delta)\ \mathbf{G}_q^T(r_q{+}1{:}r_q{+}\delta,\ i)}{\sqrt{\mathbf{d}_q(i)}}.\end{aligned} \tag{6.6}$$

Given $\hat{\mathbf{g}}_{q,i}$ and consequently $\hat{\mathbf{h}}_{q,i}$, consider the computation of approximate $\hat{c}_{q,i}$:

$$\hat{c}_{q,i} = \mathbf{e}^T\hat{\mathbf{h}}_{q,i} = \frac{|(\mathbf{P}\hat{\mathbf{g}}_{q,i})^T\mathbf{e}|}{\|\mathbf{P}\hat{\mathbf{g}}_{q,i}\|}. \tag{6.7}$$

Inserting $\hat{\mathbf{g}}_{q,i}$ as in Eq. 6.6, the denominator $1/\sqrt{\mathbf{d}_q(i)}$ cancels out. The norm $\|\mathbf{P}\hat{\mathbf{g}}_{q,i}\|$ can be computed as:

$$\begin{aligned}&\|\mathbf{P}\hat{\mathbf{g}}_{q,i}\|^2 = \|\mathbf{P}\,\mathbf{G}_q(:,\ r_q{+}1{:}r_q{+}\delta)\ \mathbf{G}_q^T(r_q{+}1{:}r_q{+}\delta,\ i)\|^2 = \\ &= \mathbf{G}_q(i,:)\Big(\mathbf{G}_q^T\mathbf{G}_{q(r_q+1:r_q+\delta,\ r_q+1:r_q+\delta)} - \mathbf{G}_q^T\mathbf{1}^T\mathbf{G}_{q(r_q+1:r_q+\delta,\ r_q+1:r_q+\delta)}\Big)\mathbf{G}_q(i,:)^T.\end{aligned}$$

Similarly, dot product with the residual is computed as:

$$\begin{aligned}(\mathbf{P}\hat{\mathbf{g}}_{q,i})^T\mathbf{e} &= \mathbf{e}^T\Big(\mathbf{P}\,\mathbf{G}_q(:,\ r_q{+}1{:}r_q{+}\delta)\mathbf{G}_q^T(r_q{+}1{:}r_q{+}\delta,\ i)\Big)^T = \\ &= \Big(\mathbf{e}^T\mathbf{G}_q(:,\ r_q{+}1{:}r_q{+}\delta) - \mathbf{e}^T\mathbf{1}\mathbf{1}^T\mathbf{G}_q(:,\ r_q{+}1{:}r_q{+}\delta)\Big)\mathbf{G}_q(r_q{+}1{:}r_q{+}\delta,\ i).\end{aligned} \tag{6.8}$$

Computation of $\hat{a}_{q,i}$ is analogous. Correctly ordering the computations yields a computational complexity of $O(\delta^2)$ per column. Note that matrices $\mathbf{G}_q^T\mathbf{G}_q$, $\mathbf{G}_q^T\mathbf{1}\mathbf{1}^T\mathbf{G}_q$, $\mathbf{e}^T\mathbf{G}_q$, $\mathbf{e}^T\mathbf{1}\mathbf{1}^T\mathbf{G}_q$ are the same for all columns (independent of i) and need to be computed only once per iteration.

The values $\hat{a}_{q,i}$ and $\hat{c}_{q,i}$ can be computed efficiently for all kernel matrices and enable the selection of the next kernel, pivot column pair q, i to be added to $\mathbf{G}_q$ and consequently $\mathbf{H}$. After selecting q, i a Cholesky step is performed (Eq. 5.19) to compute the exact $\mathbf{g}_{q,i}$ and

$$\mathbf{G}_q(:, j) \leftarrow \mathbf{g}_{q,i}. \tag{6.9}$$

The computation of a new column renders the look-ahead columns in $\mathbf{G}_q$ at indices $r_q{+}1{:}r_q{+}\delta$ invalid. After applying Eq. 6.9, all columns at indices $r_q{+}1{:}r_q{+}\delta$ are recomputed using the standard Cholesky step with pivot selection based on the current maximal value in $\mathbf{d}_q$ at a cost $O(n\delta^2)$. The exact values of $\mathbf{g}_{q,i}$ and $\mathbf{h}_{q,i}$ determine $\mathbf{h}^{(m)}$ to be added to $\mathbf{H}$ and enables the correct computation of a_m, c_m and step size γ in Eq. 6.5. The regression estimate $\mathbf{f}$ and the residual $\mathbf{e}$ can be correctly updated according to Eq. 6.4.

6.5 The Mklaren algorithm

The steps described in the previous sections complete the Mklaren algorithm (Algorithm 3). Given a sample of n data points with targets $\mathbf{y}$ and p kernel functions, the model hyperparameters are: the maximum rank r of the combined feature matrix, the number of look-ahead columns δ and L_2 norm regularization parameter λ (constraining $\mathbf{f}$, discussed in Section 6.8).

The variables related to regression estimate ($\mathbf{f}$, residual $\mathbf{e}$ and bisector $\mathbf{u}$) and individual decompositions $\mathbf{G}_q$ (active sets $\mathscr{A}$, column counters r_q) are initialized in lines 1-2. Each $\mathbf{G}_p$ is initialized using standard ICD with δ look-ahead columns, as described in Section 6.4 (line 3).

The main loop is executed for r iterations, until the sum of selected pivot columns equals $\sum_q r_q = r$, where at each iteration a kernel k_q and a pivot column $i \notin \mathscr{A}_q$ are selected and added to $\mathbf{G}_q$ and consequently the combined feature matrix $\mathbf{H}$. For each kernel k_q and each pivot $i \notin \mathscr{A}_q$, $\hat{a}_{q,i}$ and $\hat{c}_{q,i}$ are computed. Based on these approximate values, the kernel k_q and pivot i are selected. Given the optimal k_q and pivot i, the pivot column $\mathbf{g}_{q,i}$ and $\mathbf{h}_{q,i}$ are computed. The new pivot column $\mathbf{g}_{q,i}$ is added to $\mathbf{G}_q(:, r_q)$, r_q is incremented and the δ columns at $\mathbf{G}_q(:, r_q{+}1{:}r_q{+}\delta)$ are recomputed using standard ICD (lines 5-11). Having computed the exact $\mathbf{g}_{q,i}$, the true values $a_{q,i}$ and $c_{q,i}$ can be computed and the regression estimate $\mathbf{f}$ and the residual $\mathbf{e}$ are updated (lines 12-16).

The regression coefficients $\mathbf{w}$ solving for $\mathbf{Hw} = \mathbf{f}$ and required for out-of-sample prediction, are obtained by constructing $\mathbf{H}$ and solving the linear system discussed in Section 6.6 (line 17).

6.6 Out-of-sample prediction

The coefficients $\mathbf{w} \in \mathbb{R}^r$ determine a general function estimator $\hat{f}$, enabling prediction for any point in the input space. The values of $\mathbf{w}$ are found by solving a simple linear system relating the combined feature space $\mathbf{H}$ to training set regression estimate $\mathbf{f}$:

$$\mathbf{Hw} = \mathbf{f} \implies \mathbf{w} = (\mathbf{R}^T\mathbf{R})^{-1}\mathbf{Q}^T\mathbf{f}, \tag{6.10}$$

where $\mathbf{H} = \mathbf{QR}$ is the thin QR decomposition [43].

Inference of Cholesky factors for arbitrary data points is possible without explicitly repeating the Cholesky steps. The domain of $\hat{f}$ is the representation of data points in the combined feature space, constructed in Section 6.2, and is practically defined by the selected pairs of inducing points and kernels. To simplify notation, we show the approach assuming one kernel matrix, while the computation with multiple kernels is analogous.

Let $\mathscr{A} \subset \{1, 2, ..., n\}$ be an active set of pivot indices. The proposition in Section 5.5 guarantees that the approximation $\mathbf{GG}^T$ equals the Nyström approximation, given the same $\mathscr{A}$:

$$\mathbf{L} = \mathbf{GG}^T = \mathbf{K}(:, \mathscr{A})\mathbf{K}(\mathscr{A}, \mathscr{A})^{-1}\mathbf{K}(:, \mathscr{A})^T.$$

Algorithm 3: The Mklaren algorithm pseudocode.

Input:

$\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ set of points in $\mathscr{X}$,
$k_1, k_2, ...k_p$ kernel functions on $\mathscr{X} \times \mathscr{X}$,
$\mathbf{y} \in \mathbb{R}^n$ regression targets, with $\mathbf{1}^T\mathbf{y} = 0$,
r maximum total rank,
δ number of look-ahead columns,
λ regularization parameter.

Result:

$\mathbf{G}_1 \in \mathbb{R}^{n \times r_1}, \mathbf{G}_2 \in \mathbb{R}^{n \times r_2}, ...\mathbf{G}_p \in \mathbb{R}^{n \times r_p}$,
Cholesky factors,
$\mathbf{H} \in \mathbb{R}^{n \times r}$ combined feature space,
$\mathscr{A}_1, \mathscr{A}_2, ..., \mathscr{A}_p$ active sets of pivot indices,
$\mathbf{f} \in \mathbb{R}^n$ regression estimate on the training set,
$\mathbf{w} \in \mathbb{R}^r$ regression coefficients determining function estimator $\hat{f}$.

Initialize:
  $\mathbf{H} = \mathbf{0}$, residual $\mathbf{e} = \mathbf{y}$, bisector $\mathbf{u} = \mathbf{0}$, regression estimate $\mathbf{f} = \mathbf{0}$, active sets $\mathscr{A}_q = \emptyset$ and counters $r_q = 0$ for $q \in \{1, ..., p\}$.
Compute standard Cholesky Decompositions with δ look-ahead columns for $\mathbf{G}_1, \mathbf{G}_2, ..., \mathbf{G}_p$.
while $\sum_q r_q < r$ *do*
  Compute $\hat{a}_{q,i}$ and $\hat{c}_{q,i}$ for each kernel q and pivot $i \notin A_q$ (Eq. 6.7)
  Select q, i based on the minimum in Eq. 6.5
  Compute $\mathbf{g}_{q,i}$ (Eq. 5.19) and $\mathbf{h}_{q,i}$ (Eq. 6.1)
    $\mathbf{G}_q(:, r_q) \leftarrow \mathbf{g}_{q,i}$
    $\mathbf{H}(:, \sum_q r_q) \leftarrow \mathbf{h}_{q,i}$
    $r_q \leftarrow r_q + 1, \mathscr{A}_q \leftarrow \mathscr{A}_q \cup \{i\}$
  Recompute $\mathbf{G}_q(:, r_q{+}1{:}r_q{+}\delta)$ using standard ICD
  Compute true $a_{q,i}$ and $c_{q,i}$ (Eq. 6.5)
  Compute the bisector $\mathbf{u}$ of columns in $\mathbf{H}$ except $\mathbf{h}_{q,i}$ (Eq. B.5)
  Compute γ for $\mathbf{h}^{(m)} = \mathbf{h}_{q,i}$ (Eq. 6.5) and update
    $\mathbf{f} \leftarrow \mathbf{f} + \gamma\mathbf{u}$
    $\mathbf{e} \leftarrow \mathbf{e} - \gamma\mathbf{u}$
Solve linear system $\mathbf{Hw} = \mathbf{f}$ for $\mathbf{w}$ using Eq. 6.10.

This relation is used to infer the Cholesky factors for new data. Let $\mathbf{K}_*$ be the values of the kernel function $k(\mathbf{x}_*, \mathbf{x}_i)$ evaluated for n_t data points $\{\mathbf{x}_*\}$ and the active set $\{\mathbf{x}_i\}$, for $i \in \mathscr{A}$. The Cholesky factors $\mathbf{G}_* \in \mathbb{R}^{n_t \times r}$ for test samples $\{\mathbf{x}_*\}$ are inferred via the linear transform $\mathbf{T} = \mathbf{K}(\mathscr{A}, \mathscr{A})^{-1}\mathbf{K}(\mathscr{A}, :)\mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}$. That is,

$$\begin{aligned} \mathbf{G}_*\mathbf{G}^T &= \mathbf{K}_*\mathbf{K}(\mathscr{A}, \mathscr{A})^{-1}\mathbf{K}(\mathscr{A}, :) \\ &\Longrightarrow \\ \mathbf{G}_* &= \mathbf{K}_*\mathbf{K}(\mathscr{A}, \mathscr{A})^{-1}\mathbf{K}(\mathscr{A}, :)\mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1} \\ &= \mathbf{K}_*\mathbf{T}. \end{aligned} \tag{6.11}$$

The matrix $\mathbf{T} \in \mathbb{R}^{r \times r}$ is inexpensive to compute and store. The combined feature matrix $\mathbf{H}_* \in \mathbb{R}^{n_t \times r}$ is obtained after centering and normalization as in Eq. 6.1 and the predicted outputs are $\mathbf{f}_* = \mathbf{H}_*\mathbf{w}$.

6.7 Computing dual coefficients

Regardless of using kernel approximation, a limited form of model interpretation is still possible for certain classes of kernels. Again, we show the approach for one kernel matrix and the combined feature matrix $\mathbf{H}$ while the computation for multiple kernels is analogous. Kernel Ridge regression is often stated in terms of dual coefficients $\boldsymbol{\alpha} \in \mathbb{R}^n$, satisfying the relation:

$$\mathbf{H}^T\boldsymbol{\alpha} = \mathbf{w}.$$

This is an overdetermined system of equations. The vector $\boldsymbol{\alpha}$ with minimal norm can be obtained by solving the following least-norm problem:

$$\begin{aligned} &\text{minimize } \|\boldsymbol{\alpha}\|_2 \\ &\text{subject to } \mathbf{H}^T\boldsymbol{\alpha} = \mathbf{w}. \end{aligned}$$

The problem has an analytical solution equal to

$$\boldsymbol{\alpha} = \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{w}.$$

Obtaining dual coefficients $\boldsymbol{\alpha}$ can be useful if the range of the explicit feature map induced by a kernel k is of finite dimension, such that $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})\phi(\mathbf{x})$, $\phi : \mathscr{X} \mapsto \mathbb{R}^d$,

which is the case for e.g. linear, polynomial, and various string kernels. The values of regression coefficients in the range of ϕ, $\mathbf{w}_\phi \in \mathbb{R}^d$ are obtained by computing the matrix $\boldsymbol{\phi} \in \mathbb{R}^{n \times d}$ for the training set and considering

$$\mathbf{w}_\phi = \boldsymbol{\phi}^T \boldsymbol{\alpha}.$$

Moreover, if the vector $\boldsymbol{\alpha}$ is sparse, only the relevant portions of $\boldsymbol{\phi}$ need to be computed. This condition can be enforced by using techniques such as matching pursuit when solving for $\boldsymbol{\alpha}$ [81].

6.8 L_2 norm regularization

Increasing the model bias by bounding the norm $\|\mathbf{w}\|$ or equivalently $\|\mathbf{f}\|$ is realized as follows. Zou and Hastie [82] prove the following lemma, which shows that the L_2-regularized regression problem can be stated as ordinary least squares using appropriate augmentation of the data $\mathbf{X}, \mathbf{y}$. The lemma assumes for all t, $\|\mathbf{X}(:,t)\| = 1, \mathbf{1}^T\mathbf{X}(:,t) = 0$ and $\mathbf{1}^T\mathbf{y} = 0$.

Lemma. Define the augmented data set $\mathbf{X}^\lambda, \mathbf{y}^\lambda$ to equal

$$\mathbf{X}^\lambda = \sqrt{(1+\lambda)}\begin{pmatrix}\mathbf{X} \\ \sqrt{\lambda}\mathbf{I}\end{pmatrix}$$

$$\mathbf{y}^\lambda = \begin{pmatrix}\mathbf{y} \\ \mathbf{0}\end{pmatrix}.$$

The least-squares solution of $\mathbf{X}^\lambda\mathbf{w} = \mathbf{y}^\lambda$ is then equivalent to L_2-regularized solution of the original problem with data $\mathbf{X}, \mathbf{y}$ and regularization parameter λ (see Eq. B.12).

The lemma allows us to define simple modification of the combined feature matrix in Eq. 6.2:

$$\mathbf{h}^\lambda_{q,i} = \mathbf{P}\begin{pmatrix}\mathbf{P}\mathbf{g}_{q,i} \\ 0 \\ 0 \\ \dots \\ \lambda \\ \dots \\ 0\end{pmatrix} / \|\mathbf{P}\begin{pmatrix}\mathbf{P}\mathbf{g}_{q,i} \\ 0 \\ 0 \\ \dots \\ \lambda \\ \dots \\ 0\end{pmatrix}\|.$$

This definition is now equivalent to performing LAR in augmented space $\mathbf{H}^{\lambda}$, resulting in an L_2 regularized solution for $\mathbf{f}$ after r steps of the approximation. It is straightforward to modify Eq. 6.5, and Eq. 6.7-6.8 for $\mathbf{h}^{\lambda}_{q,i}$.

6.9 Computational complexity

The Mklaren algorithm scales linearly both in the number of data points n and kernels p. The computational complexity is

$$O(n\delta^2 + r(r^2 + np\delta^2 + n\delta^2) + nr^2 + r^3) = O(r^3 + nr^2 + npr\delta^2). \tag{6.12}$$

The look-ahead Cholesky decompositions are standard Cholesky decompositions with δ pivots and complexity $O(n\delta^2)$. The main loop is executed r times. The selection of kernel and pivot pairs includes inverting $\mathbf{H}_A^T\mathbf{H}_A$ of size $r \times r$, thus having a complexity of $O(r^3)$. However, as each step is a rank-one modification to $\mathbf{H}_A^T\mathbf{H}_A$, the Morrison-Sherman-Woodbury lemma on matrix inversion can be used to achieve complexity $O(r^2)$ per update [83]. The computation of correlations with the bisector in Eq. 6.7 and residuals are computed for p kernels in $O(np\delta^2)$. Recomputation of δ Cholesky factors requires standard Cholesky steps of complexity $O(n\delta^2)$. The computation of the gradient step and updating the regression estimate is $O(n)$. The QR decomposition in Eq. 6.10 takes $O(nr^2)$, and the computation of linear transform $\mathbf{T}$ in Eq. 6.11 is of $O(r^3 + nr^2)$ complexity. The computational complexity bound in Eq. 6.12 also provides natural parameter bounds — for total complexity lower than $O(n^2)$ — which equal $r < \sqrt{n}$ and $\delta < \sqrt{\frac{n}{rp}}$.

6.10 A function space view

This section discusses the difference between multiple kernel learning of Mklaren and approximating the unweighted sum of kernel matrices by inducing point-based approximation methods. It is well-known that learning with Kernel Ridge Regression (KRR) with the input of a sum of the kernels (in the dual space) is equivalent to the concatenation of features induced by the kernel (in the primal space) with regard to the obtained regression estimates.

A general MKL setting is defined as follows. Given a sample of data points $\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n$, and multiple kernels $k_1, k_2, ..., k_p$ as inner products, the output functions f are sampled from a space spanned by

$$f \in \text{span}\{k_1(\mathbf{x}_1, \cdot), k_1(\mathbf{x}_2, \cdot), ..., k_1(\mathbf{x}_n, \cdot), \\ k_2(\mathbf{x}_1, \cdot), k_2(\mathbf{x}_2, \cdot), ..., k_2(\mathbf{x}_n, \cdot), \\ k_p(\mathbf{x}_1, \cdot), k_p(\mathbf{x}_2, \cdot), ..., k_p(\mathbf{x}_n, \cdot)\}\,, \tag{6.13}$$

with $k_q(\mathbf{x}_i, \cdot)$ a basis function of kernel q centered at data point $\mathbf{x}_i$. Evaluating $k_q(\mathbf{x}_i, \mathbf{x}_j)$ for each kernel k_q and pair $\mathbf{x}_i, \mathbf{x}_j$ in the training set gives rise to a $n \times np$ design matrix. Given n associated target values $y_1, y_2, ..., y_n$, a solution to the overdetermined linear system is the np coefficients of the function estimator $\hat{f}$, which in turn defines training set estimates $\mathbf{f} \in \mathbb{R}^n$. Note that the same estimates $\mathbf{f}$ are obtained if the basis functions are sums of the p kernels, giving rise to a similar, $n \times n$ linear system. Summing the kernels in the first case occurs simultaneously with optimizing the coefficients determining $\hat{f}$.

Introducing non-zero prior *kernel weights* - real number coefficients associated to each of the p kernels - does not effect neither the output function span, nor the optimal solution $\hat{f}$, as it accounts for a simple rescaling of the basis functions listed in Eq. 6.13. This explains the equivalent performance of KRR regardless of learning the kernel weights [84]. By definition, Mklaren selects the basis functions from a subspace of Eq. 6.13.

The situation is very different when a kernel matrix sum $\mathbf{K} = \sum_{q=1}^{p} \mathbf{K}_q$ is used with an inducing point-based approximation (Cholesky, Nyström). Let $\mathbf{G} \in \mathbb{R}^{n \times r}$ be a rank-r approximation of $\mathbf{K}$ obtained by selecting an active set $\mathscr{A}$ of size r and constructed using Eq. 5.19. Then, as shown in Section 6.6, the column space of $\mathbf{G}$ equals the span of $\mathbf{K}(:, \mathscr{A})$. Consequently, the output functions f are sampled from a space:

$$f \in \text{span}\{\sum_q k_q(\mathbf{x}_i, \cdot), \forall i \in \mathscr{A}\}. \tag{6.14}$$

Note that this space does not equal that in Eq. 6.13, and is different in two main aspects:

- evaluating all $\sum_q k_q(\mathbf{x}_i, \mathbf{x}_j)$ results in an $n \times r$ design matrix - an *underdetermined* linear system.

- the basis functions are now not individual kernels centered at input points. Instead, they are *unweighted sums of all input kernels*, centered at inducing points. Summing the kernels precedes the optimization of $\hat{f}$, further restricting the output function space.

The single (practically unrealistic) exception is the case where $r = n$ and $\mathbf{K}$ is exactly approximated by $\mathbf{G}$. These facts motivate an algorithm to select both the basis functions and the corresponding inducing points separately, as multiple kernel learning is not readily achieved by passing the sum of the kernel matrices to a single kernel matrix approximation algorithm such as ICD, CSI or Nyström. Moreover, one is then able to use different kernels in different regions/subsets of the input space.

In practical terms, if the p kernels are exponentiated-quadratic kernels with different length scales, each basis function at point $\mathbf{x}_i$ in Eq. 6.14 is constrained to an unweighted sum of basis function centered at $\mathbf{x}_i$, forcing the resulting output function to be distorted by all initial length scales. Clearly, the space in Eq. 6.13 contains, but is in general not equal to the space in Eq. 6.14. The consequences of the differences in function space spans are investigated experimentally in Section 7.3.

Experiments with Mklaren

7

This part provides an empirical evaluation of the proposed methods. The selection of methods and their settings is argued in Section 7.1. Empirical results are provided on robust inducing point selection (Section 7.2), effect of different function spaces on one-dimensional time series (Section 7.3), and experiments with discrete input spaces (Section 7.4). Furthermore, we evaluate the dimensionality of approximations (Section 7.5), multiple kernel learning (Section 7.6) and empirical execution time (Section 7.7).

7.1 A note on compared methods

All the compared methods are used with Kernel Ridge Regression and are listed in the order of increasing number of assumptions concerning the used kernels. The experiments in the following subsections were designed to elucidate the differences and trade-offs in terms of applicability, scalability and prediction error. All the experiments assume a data set with $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ points in p inner product spaces $\mathcal{X}_q$ with corresponding kernel functions $k_q(\cdot, \cdot)$ and real number targets $\mathbf{y} = (y_1, y_2, ..., y_n)$. The dimension of the resulting approximate feature space is equal to rank $r < n$ and is equal for all compared methods.

The methods Mklaren, CSI, ICD and Nyström approximate the kernel matrix using r inducing points defining the basis functions $k_q(x_i, \cdot)$. The methods ICD and Nyström do not consider target values $\mathbf{y}$, while CSI uses a matching pursuit-based heuristic[1]. The ICD selects each inducing point based on a lower-bound on approximation error gain. A relatively recent version of the Nyström method uses approximate leverage scores to determine the sampling distribution of inducing points within the training set. The methods CSI, ICD and Nyström do not assume multiple kernels as opposed to Mklaren. To implement MKL for the former three methods and ensure comparable ranks, we either i) approximate an unweighted sum of all p kernels up to rank r or ii) approximate each of the p kernels up to rank $\lceil \frac{r}{p} \rceil$. The versions using the approach ii) are denoted CSI*, ICD* and Nyström*. All methods in this paragraph can be seen as discrete optimization; the basis functions are selected from provided kernels with fixed hyperparameters and the inducing points are constrained to the training set. No further restrictions are posed on neither the input spaces $\mathcal{X}_q$ nor kernels $k_q(\cdot, \cdot)$.

[1] With or *matching pursuit*, we refer to the approach which chooses the basis functions in a greedy manner and solves for the full least-squares estimate at each step.

Random Fourier Features (RFF) approximate the frequency components of the basis functions. Here, the *matching pursuit* optimization is used; at each step number of look-ahead basis functions δ is used to generate basis functions for k_q, greedily selecting the best-aligned basis to the current residual, up to rank r [85]. Multiple kernel learning is realized by sampling the features from different spectral densities, which are governed by the bandwidth associated to each kernel. To model non-stationary effects, we implement the spectral densities recently proposed by Ton et al. [86], denoting this method as RFF-NS. More details on both stationary and non-stationary RFF can be found in Appendix B.3.3.

Sparse Pseudo Input Gaussian Processes (SPGPs) approximate the kernel matrix by continuous, maximum likelihood optimization. This method explores a space of models with the largest capacity, as it optimizes both the locations of the inducing points (in the span of the training set), as well as kernel hyperparameters. Consequently, it is limited to real vector spaces $\mathscr{X}_q = \mathbb{R}^d$ of finite dimension d and continuous, differentiable kernels. The method is described in more detail in Appendix B.3.3.

As an empirical lower bound on prediction error, we used the L_2-regularized Kernel learning (L2-KRR), learning both the kernel weights and the regression coefficients given the full kernel matrix [76]. To evaluate the ability to learn with multiple kernels independent of the approximation, we used the MKL methods Align (independently learned weights for each kernel), AlignF (linear combination) and AlignFC (convex combination) from Cortes et al. [75].

7.2 Robust selection of inducing points

The methods compared in this section are based on selecting inducing points in the input space $\mathscr{X}$, but not necessarily coinciding with the training set. The motivation to compare the robustness of inducing point selection is based on the premise that the variance in the data is generated by two different sources: the placement of inducing points and noise. In this section, we explore how difference regimes of variance affect the selection of inducing points. Even though the Gaussian Processes represent functions over the whole domain $\mathscr{X}$, we require a finite sample of data points $\{x_1, x_2, ..., x_n\}$ and function evaluations $\mathbf{f} \in \mathbb{R}^n$ to apply GPs in practice. In the same way a subset of data is an approximation to the underlying function, the set of inducing points $\mathscr{A}$ is an approximation to the subset of data $\{x_1, x_2, ..., x_n\}$, possibly influenced by regions where $\mathbf{f}$ varies.

We use simulated data and assume the input space to be the real line $\mathbb{R}$. The training set is composed of $n = 100$ equally-spaced points within an interval $[0, 20]$ and the exponentiated-quadratic kernel is used as a covariance function (Eq. 5.2). The $r < n$ true inducing points are sampled without replacement from the training set and define the active set $\mathscr{A}$. The inducing points are sampled in two different regimes, defined by a discrete probability distribution. The *uniform* regime assumes an uniform distribution. The *biased* regime assumes an increasing probability towards higher indices in $x_1, x_2, ..., x_n$:

$$P(i \in \mathscr{A}) \propto ([0, 20]_n(i))^2,$$

where $[a, b]_n$ is an equally-spaced, discrete interval of n real numbers from a to b, and (i) represents its i-th element. Thus, the inducing points are sampled non-uniformly within the input region. The resulting rank-deficient kernel matrix $\mathbf{L}$ is computed as the Nyström approximation:

$$\mathbf{L} = \mathbf{K}(:, \mathscr{A})\mathbf{K}(\mathscr{A}, \mathscr{A})^{-1}\mathbf{K}(\mathscr{A}, :).$$

The target values are sampled from a zero-mean Gaussian Process (GP) with covariance matrix $\mathbf{L}$ and at input-dependent noise represented by a vector $\boldsymbol{\sigma}^2 \in \mathbb{R}^n$.

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{L} + \mathbf{I}\boldsymbol{\sigma}^2).$$

The noise vector is also set according to two different regimes:

$$\sigma_i^2 = 1 \quad \text{(the } \textit{fixed} \text{ regime)}.$$

and

$$\sigma_i^2 = 10^{[-2,2]_n(i)} \quad \text{(the } \textit{increasing} \text{ regime)}.$$

Note that with the *increasing* regime, noise increases towards the same region in the input space as does the selection of inducing points with *biased* sampling regime.

Each inducing point $\mathbf{x}_i$ determines a Gaussian basis function centered at $\mathbf{x}_i$ with the spread governed by γ. This is a particular realization of the function space presented in Section 6.10. Thus, including any data point in the low-rank approximation of $\mathbf{K}$ amounts to adding a corresponding basis function to the current function approximation $\hat{f}(\mathbf{x})$.

7.2.1 *Inducing points location distributions*

We evaluate inducing point-based methods Mklaren, CSI, ICD, SPGP, and Nyström for a fixed setting of rank $|\mathscr{A}| = r = 5$. This value is relatively low, as the most notable differences between the methods are manifested in the first steps of approximation, with all methods converging in performance above a certain value of r (see Sections 7.5 and 7.6 below).

For each combination of γ, sampling and noise regimes, we generate 500 sample data sets (vectors $\mathbf{y}$). The empirical distribution of inducing points is then compared to the true distribution. Table 7.1 shows the Kullback-Leibler divergence (KL, discrete entropy) between the true and approximating distributions. The visual comparison of distributions shown as cumulative histograms for one setting of γ is shown on Fig. 7.1.

The ordering of methods for different regimes appears consistent for various values of γ. The main differences are caused by the inducing points sampling regime. While Mklaren, CSI and SPGP are supervised methods and depend on $\mathbf{y}$, the Nyström method samples roughly uniformly while ICD selects fixed inducing points due to constant inputs $\mathbf{X}$.

With uniform sampling regime, the best performance is expectedly achieved by the Nyström method, as both the true and the sampling distribution are uniform (e.g. KL=0.002 for intermediate value $\gamma = 0.3$). Mklaren performs second best for uniform/fixed regime (KL=0.009). In the uniform/increasing regime, both Mklaren and CSI are clearly affected by noise, with Mklaren achieving a slightly better fit on 3 out of 4 cases (Fig. 7.1b).

Unsurprisingly, for the biased sampling regimes, supervised methods Mklaren, CSI and SPGP perform consistently better than the two unsupervised methods, which is reflected in lower KL and can be explained by the information on targets improving inducing point selection. At fixed noise, the inducing point distribution of Mklaren is closest to the true distribution (Fig. 7.1c). Mklaren performs best in 6 out of 8 tested cases with biased regime. At an extremely short length scales / high frequencies (γ=3.0), all the information on inducing points is lost in the noise, and the uniform sampling performed by the Nyström method is the best option. Interestingly, as can be seen on the cumulative histograms, the SPGP method places the inducing points uniformly across the input domain regardless of heteroskedasticity and/or biased inducing point placement. This is attributable to the uniform prior on inducing point

Table 7.1

Kullback-Leibler divergence of fitting distributions with low-rank kernel matrix approximation methods. Number of samples $n = 100$, rank $r = 5$ and the length scale hyperparameter $\gamma \in \{0.1, 0.3, 1, 3\}$ (top to bottom).

	Mklaren	CSI	ICD	Nyström	SPGP
$\gamma = 0.1$					
uniform/fixed	0.026	0.066	3.580	*0.001*	0.063
uniform/increasing	0.142	0.270	3.580	*0.001*	0.032
biased/fixed	*0.247*	0.289	4.137	0.420	0.469
biased/increasing	*0.079*	0.252	4.137	0.420	0.446
$\gamma = 0.3$					
uniform/fixed	0.009	0.045	3.583	*0.002*	0.024
uniform/increasing	0.157	0.355	3.583	*0.002*	0.018
biased/fixed	*0.191*	0.230	4.978	0.430	0.452
biased/increasing	*0.024*	0.082	4.978	0.430	0.461
$\gamma = 1.0$					
uniform/fixed	0.003	0.059	3.583	*0.002*	0.005
uniform/increasing	0.314	0.060	3.583	*0.002*	0.012
biased/fixed	*0.193*	0.451	4.205	0.424	0.477
biased/increasing	*0.013*	0.153	4.205	0.424	0.471
$\gamma = 3.0$					
uniform/fixed	0.592	2.193	3.639	*0.002*	*0.002*
uniform/increasing	1.020	2.160	3.639	*0.002*	0.007
biased/fixed	2.151	5.482	6.764	*0.422*	0.455
biased/increasing	1.590	5.371	6.764	*0.422*	0.453

locations, which is further discussed below in Section 7.3 and Appendix B.3.5.

The apparent accuracy of CSI and Mklaren in the biased/increasing regime can be explained by high function variance in the regions where the inducing points are also present. This is seen on Fig. 7.1b, where the shift in noise heavily influences the inducing point selection. However, the shift caused by true change in location

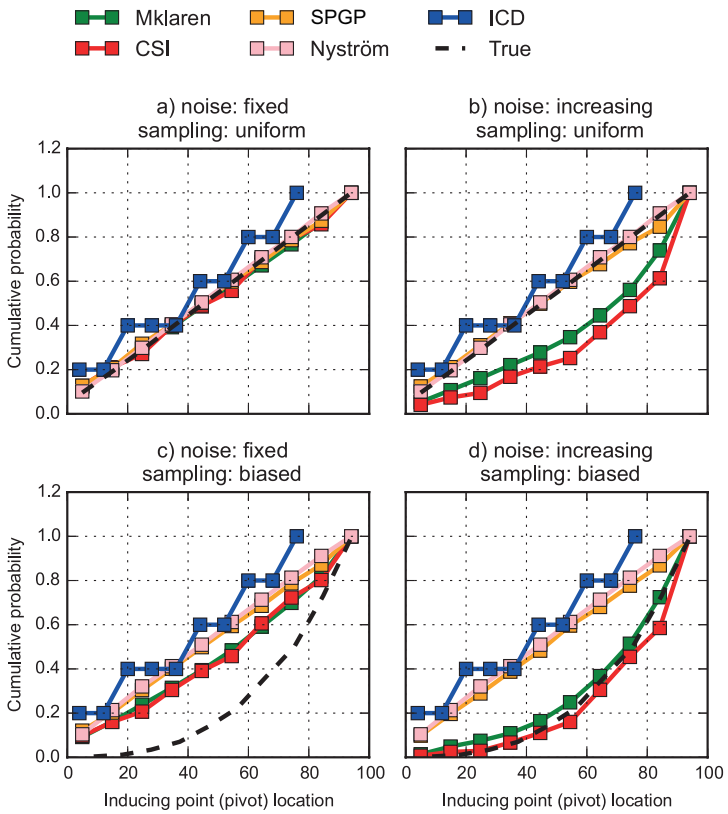


Figure 7.1

Cumulative histograms for selecting inducing points (at $\gamma = 0.3$) obtained with 500 replicate data sets. The true cumulative distribution is shown in black.

of the inducing points also somewhat contributes to the total shift (Fig. 7.1c). As seen on Fig. 7.1d, both methods tend to select the inducing points with a significant bias towards the boundary of the input domain. Finally, the Mklaren fit for the biased/increasing case (KL=0.013) is substantially better than that of CSI (KL=0.153). In summary, a non-uniform probability of inducing points within the input domain favours supervised pivot selection methods Mklaren and CSI, with a slight advantage of Mklaren for noisy data, outperforming CSI in 15 out of 16 comparisons.

7.2.2 *Matching pursuit versus Least-angle regression*

To elucidate the differences between heuristics used by Mklaren and CSI, we visualize the solution paths as an increasing number of inducing points are added to the model. Both methods are based on a current regression estimate and the gain associated to each unused inducing point. In both cases, this gain is estimated by means of look-ahead decompositions. To make the scenarios as comparable as possible, we set the CSI kernel approximation / fitting trade-off parameter $\frac{\kappa_2}{\kappa_1+\kappa_2} = 0.99999$, allowing only 0.001% of gain to be influenced by kernel approximation.

The subtle difference in gain estimation is in the way the regression estimate is computed. For CSI, the regression estimate at each step is taken to be the full least-squares solution. Mklaren employs the LAR criterion, where intermediate regression estimates are more conservative; the information available by including a basis function is used only up to the point at which there is another, unused basis function equally correlated with the current residual. This point is illustrated in Fig. 7.2. Observe that once CSI selects an inducing point, the regression estimate passes almost exactly through that point - it is the optimal least-squares fit at that iteration. Mklaren, on the other hand, will not maximally increase the magnitude of the associated basis function, but first select all the pivots before computing the least-squares fit on the last step.

The choice of the inducing points appears to be more appropriate with the use of the LAR criterion, particularly in the cases where the locations of the true inducing points are sampled non-uniformly. Quantitatively, Mklaren achieved a 0.94 ± 0.073 Pearson correlation with the true signal and $71\% \pm 15\%$ explained variance (average over 30 trials). In comparison, CSI scored 0.75 ± 0.23 Pearson correlation and $50\% \pm 24\%$ explained variance. In summary, the conservative selection of inducing points defined by the LAR criterion can lead to low-rank approximations that are closer to the true inducing points.

7.3 *Time series*

To provide intuition on the behaviour of the compared methods, we first show examples on simple data sets with regularly sampled, one-dimensional inputs. The *Appliances energy prediction* data set[2] contains time series measurements of temperature in nine building compartments (labelled T_1-T_9). The measurements are recorded at 10

[2] https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction

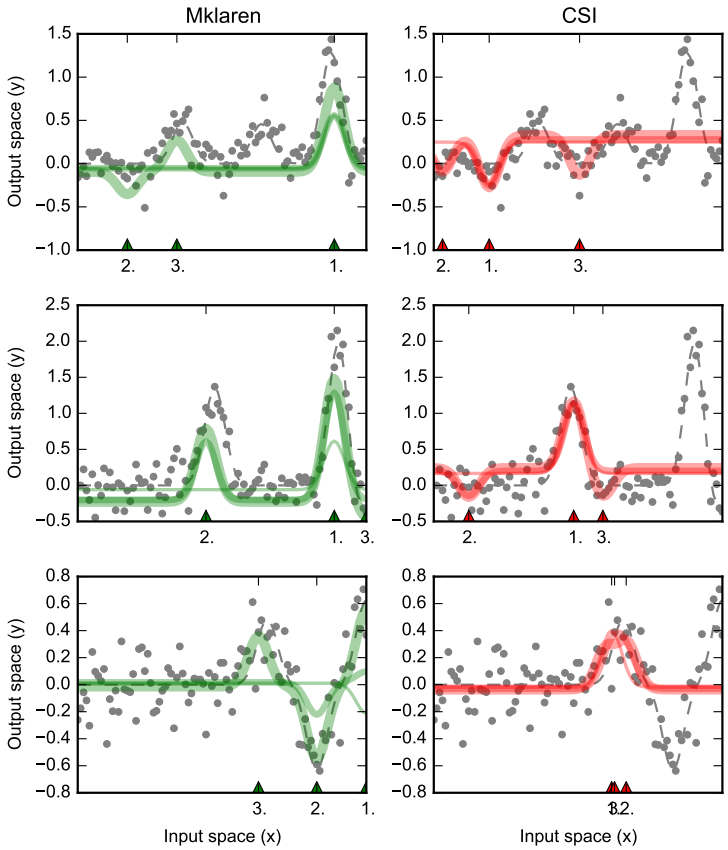


Figure 7.2

Three artificial data sets (in rows) and corresponding regression estimates Mklaren (green) and CSI (red). The functions are samples from a degenerate Gaussian Process with exponentiated-quadratic covariance and fixed noise variance. The true (noiseless) function is plotted with a dashed line. Intermediate regression estimates for ranks $r = 1, 2, 3$ are obtained by adding a basis function centered at the inducing point. The location and the order of inducing point selection is marked with the corresponding number on the x-axis. Thicker lines represent later regression estimates. Note that Mklaren only proceeds to the full least-squares solution at the last step (for $r = 3$).

minute intervals over 20 weeks. Each week, beginning with Monday, was treated as a separate time series of length 1008; to make the signals comparable across weeks, each signal T_i is mapped as $T_i' = T_i - T_{\text{out}}$, where T_{out} represents the corresponding outside temperature.

Signals were split into training, validation and test sets by assigning each consecutive time point in a different set. The one-dimensional input space was thus represented as the time index of each measurement in a week. The training set was used to fit

the data on a) exponentiated-quadratic kernels (Eq. 5.2) with the hyperparameter γ in $\{10^{-4}, 10^{-2}, 1, 10^{2}, 10^{4}\}$ and b) the Matérn 3/2 kernel (Eq. 5.4) with the length scale hyperparameter assuming equivalent values. The validation set was used to tune the regularization hyperparameter λ within $\{0, 10^{-1}, 10^{-0.5}, 1, 10^{0.5}, 10^{1}\}$. The look-ahead parameter for Mklaren, CSI, RFF and RFF-NS was set to $\delta = 10$, while rank was set to the number of days in a week ($r = 7$). The performance evaluation metric was the average root mean-squared error (RMSE) on the test sets, averaged over 20 signals (weeks).

Summarized results over the nine data sets are shown in Table 7.2. Unsurprisingly, supervised methods selecting components from different length scales — Mklaren, SPGP, RFF and RFF-NS — performed best. The difference between Mklaren and the kernel matrix approximations ICD, CSI and Nyström are statistically significant at $P < 0.05$ (Friedman rank test, Fig 7.3, ref. [59]). With the exponentiated-quadratic kernel, Mklaren performed best in 4 out of 9 data sets, and 8 out of 9 cases with the Matérn kernel. With both kernels and on all time series, Mklaren outperformed the kernel matrix approximation methods ICD, CSI and Nyström. The supervised method CSI expectedly outperformed unsupervised ICD and Nyström.

Surprisingly, Mklaren slightly outperformed SPGP, which is the most flexible of the methods as the inducing point locations can be optimized in a continuous manner. However, as Mklaren imposes no prior on inducing point locations, it can sometimes appear more flexible by placing the inducing points non-uniformly, while SPGP will favour uniform placement (see Appendix B.3.5). This is again confirmed on Fig. 7.4, which highlights the differences in output function spaces generated by each of the approximation methods.

Low-rank approximations of a single kernel matrix (unweighted sum of kernel matrices) — CSI, ICD and Nyström — all suffer from the same effect: the basis functions are sums of basis functions at individual length scales. This results in a combination of low- and high- frequency signals, caused by summing the kernels prior to kernel matrix approximation (Eq. 6.14, Section 6.10). Constraining the output function space in this way expectedly leads to sub-optimal performance, regardless of the supervised selection of the inducing input points. Conversely, Mklaren selected the basis functions and kernels from a larger function space (Eq. 6.13, Section 6.10).

Most similar to Mklaren, the basis functions defined by RFF were also sampled from functions with defined length scales, which are approximated with periodic compo-

Table 7.2

RMSE on time series models with the exponentiated-quadratic and Matérn kernels for data sets with temperature measurements in nine building compartments (T_1-T_9). The best score for each compartment is shown in red.

k_{exp}	Mklaren	CSI	ICD	Nyström	RFF	RFF-NS	SPGP
T_1	2.04±0.67	2.48±0.82	2.48±0.66	2.57±0.74	2.37±0.84	*2.00±0.74*	2.26±0.72
T_2	*1.63±0.42*	2.00±0.63	2.00±0.44	2.20±0.48	1.77±0.59	1.74±0.32	1.79±0.44
T_3	*2.08±0.80*	2.55±0.89	2.56±0.76	2.80±0.78	2.17±0.93	2.20±1.09	2.35±0.83
T_4	2.00±0.70	2.46±0.81	2.47±0.68	2.68±0.92	2.11±0.76	*1.99±0.62*	2.28±0.70
T_5	2.14±0.73	2.66±0.82	2.61±0.70	2.79±0.87	*2.11±0.86*	2.55±0.73	2.37±0.78
T_6	1.05±0.51	1.17±0.54	1.19±0.55	1.18±0.56	*1.00±0.47*	1.07±0.50	1.12±0.60
T_7	1.95±0.79	2.56±0.88	2.46±0.72	2.86±0.75	*1.91±0.96*	2.14±0.79	2.17±0.83
T_8	*2.09±0.77*	2.68±0.98	2.61±0.73	2.83±0.74	2.59±1.47	2.25±0.89	2.33±0.80
T_9	*2.06±0.81*	2.55±0.91	2.54±0.77	2.75±0.73	2.26±0.90	2.43±0.96	2.31±0.88
$k_{\text{Mat},3/2}$							
T_1	*1.99±0.66*	2.55±0.82	2.55±0.65	2.81±0.74	-	-	2.20±0.71
T_2	*1.58±0.39*	2.12±0.64	2.07±0.45	2.08±0.48	-	-	1.75±0.44
T_3	*2.04±0.76*	2.53±0.92	2.61±0.75	2.80±0.79	-	-	2.26±0.86
T_4	*1.91±0.69*	2.41±0.72	2.54±0.66	2.72±0.80	-	-	2.15±0.73
T_5	*2.06±0.73*	2.60±0.95	2.68±0.71	2.87±0.88	-	-	2.32±0.77
T_6	1.04±0.51	1.14±0.59	1.17±0.55	1.18±0.53	-	-	*1.03±0.57*
T_7	*1.92±0.76*	2.53±0.91	2.57±0.71	2.61±0.78	-	-	2.16±0.80
T_8	*2.06±0.75*	2.59±0.77	2.70±0.71	2.92±0.79	-	-	2.27±0.80
T_9	*1.97±0.80*	2.58±0.95	2.62±0.74	2.84±0.84	-	-	2.24±0.85

nents rather than inducing points. This way, Mklaren is able to model data dependencies at different length scales in regions of the input space defined by inducing points, which is not possible by the stationary RFF. The improved performance by Mklaren is most likely attributable to direct exploitation of the target signal, rather than the two-step approach of initially generating random features and subsequently selecting them using matching pursuit.

The non-stationary RFF-NS, somewhat unexpectedly, performed slightly worse than stationary RFF. We speculate that this can be attributed to the specific way the optimization is implemented; as both approaches sample periodic basis functions and

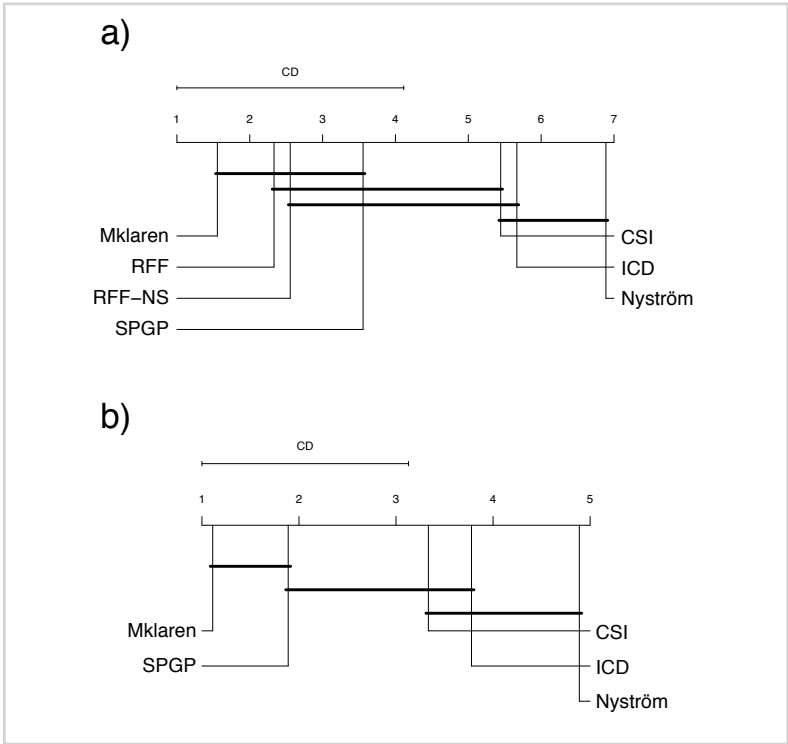


Figure 7.3

Critical distance diagrams, displaying average rankings accross data sets in Table 7.2 at significance level $P < 0.05$. a) Exponentiated-quadratic kernel. b) Matérn 3/2 kernel.

the signal variance appears constant across the input region, stationary features are not at a notable disadvantage. Since both methods are allowed the same number of δ look-ahead draws, and as RFF samples from a smaller space, the probability of finding useful basis functions appears slightly larger. These type of claims, however, need to be based on more analytical foundations, and are out of scope of the present work.

On that note, an interesting future challenge would be to couple the efficient sampling from different spectral densities in a supervised, Mklaren-like manner. This type of approach could be based on conditioning the sampling from the spectral densities (Eq. B.16) on some information on the target signal. Additionally, since the spectral densities are continuous functions in both hyperparameters and frequencies $\boldsymbol{\omega}$, a maximum-likelihood based optimization would likely enable such a hybrid approach.

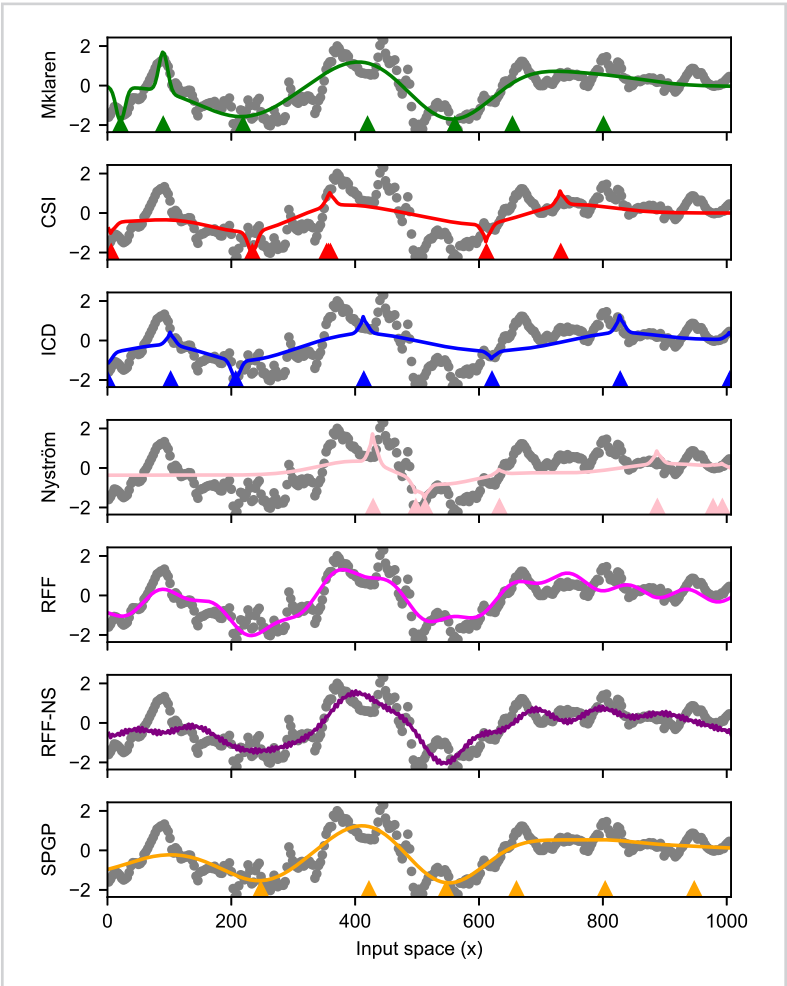


Figure 7.4

Selected example of a time series fit with the exponentiated-quadratic kernels (T_6, first week, $\lambda = 0$). The predictions for each model are compared to the test data points (in gray). The locations of selected inducing points are marked with triangles near the bottom edge (where applicable).

7.4 String kernels

In this section, we evaluate the performance in a domain of discrete, finite length character strings. This is a common scenario in text processing or bioinformatics, and

different kernels exists for this task [74]. First, we examine the behaviour on synthetic data sets with known generating parameters and finally present experiments on a real biological data set.

The data points are n strings of length L, derived from a finite alphabet representing DNA bases, $\mathscr{B} = \{A, C, G, T\}$. We assume a real number target y_i associated to each data point $\mathbf{s}_i$. The string-type data can be transformed into a numerical form using e.g. manual preprocessing, string kernels, or deep learning. In this scenario, we assume the data to only be available through evaluation of kernels between data points.

We focus on the *spectrum kernel*, called also the n-gram bag-of-words kernel. Recall the definition from Eq. 5.6; The spectrum kernel between two strings is defined as a product of occurrences of each substring $\mathbf{u}$ of length ℓ. The substring length ℓ is a hyperparameter and count($\mathbf{u}$, $\mathbf{s_i}$) the number occurrences of $\mathbf{u}$ in $\mathbf{s_i}$. The already mentioned exponential space complexity of explicitly constructing the feature space renders SPGP-like continuous optimization of inducing inputs locations unfeasible.

More interestingly, there exists a similarity of hyperparameter ℓ with the bandwidth/length scale hyperparameter $\gamma \propto \frac{1}{\ell}$) of the exponential kernel. Decreasing γ results in the larger distance (length scale) at which the points in a vector space are conditionally dependent (in terms of output values). The same effect is achieved by decreasing ℓ, as the smaller ℓ causes more points to be conditionally dependent. This concept will be explored in a similar way as the length scales of continuous functions in the previous section.

7.4.1 Experiments on synthetic data

First, we evaluate the performance on synthetic data sets with random strings of length $L = 30$, generated by uniformly sampling from $\mathscr{B}^L$. The sizes of training, validation and test sets were set to $n_{\text{tr}} = 500$, $n_{\text{val}} = n_{\text{te}} = 5000$ respectively. We used a sum of 10 spectrum kernels with ℓ in the integer interval $[1, 10]$. We randomly sampled an active set $\mathscr{A}$ of $r = 7$ inducing points from the training set. The underlying true kernel matrix was $\mathbf{L} = \mathbf{K}_{\ell=4}(:, \mathscr{A})\mathbf{K}_{\ell=4}(\mathscr{A}, \mathscr{A})^{-1}\mathbf{K}_{\ell=4}(:, \mathscr{A})^T$ with $\mathbf{K}_{\ell=4}$ the spectrum kernel matrix with a fixed $\ell = 4$. The true function was sampled from a degenerate Gaussian Process:

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{L}). \tag{7.1}$$

The sampling, training and validation (optimizing λ) was repeated 30 times. The methods evaluated methods were ICD, Nyström, CSI (using an unweighted sum of

kernel matrices), and Mklaren (approximating the kernel matrices individually). Out of 30 comparisons, Mklaren achieved lowest RMSE in 23 (76.6 %), while CSI in the remaining 7 (23.3%) cases ($P < 0.0017$, Wilcoxon paired signed rank test). The differences are shown as a critical distance plots on Fig. 7.5a, showing the differences in ranks at $P < 0.051$. Difference in the results can be explained by the effect of approximating the kernel sum as outlined in the Sections 6.10 and 7.3. We illustrate this point next.

To visualize the model fit for a given method, we devise the following strategy. Here, each method is examined through evaluations of its inferred output function $\hat{f}(\mathbf{s})$. First, we compute the distance between strings $\mathbf{s}_i, \mathbf{s}_j$, induced by a kernel k_ℓ:

$$\Delta_\ell(\mathbf{s}_i, \mathbf{s}_j) = \sqrt{k_\ell(\mathbf{s}_i, \mathbf{s}_i) + k_\ell(\mathbf{s}_j, \mathbf{s}_j) - 2k_\ell(\mathbf{s}_i, \mathbf{s}_j)}. \tag{7.2}$$

This is to be compared with the difference

$$\Delta_{\hat{f}}(i, j) = |\hat{f}(\mathbf{s}_i) - \hat{f}(\mathbf{s}_j)|$$

where $\hat{f}(\mathbf{s}_i), \hat{f}(\mathbf{s}_j)$ are the predicted outputs for strings $\mathbf{s}_i, \mathbf{s}_j$, respectively. We then calculate the *empirical alignment* between each kernel (parametrized by ℓ) and the function estimate $\hat{f}$ as

$$\text{alignment}(\ell, \hat{f}) = \rho(\Delta_\ell(i, j), \Delta_{\hat{f}}(i, j)), \tag{7.3}$$

where ρ is the Pearson correlation, and i, j are 1000 bootstrap pairs randomly sampled from the test set. Thus, Eq. 7.3 estimates the importance of a substring length ℓ for the output function $\hat{f}$. The alignment computed between each ℓ and the true signal $\mathbf{y}$ is considered as ground truth.

Fig. 7.5b shows a typical example where Mklaren outperforms the kernel matrix sum approximations and most closely matches the profile of the true output. The methods approximating the kernel sum are prone to the short length substrings (i.e. $\ell = 1, 2, 3$), which is analogous to high frequency signals in Section 7.3. These results confirm Mklaren is suitable for modelling discrete string data, where selection of individual basis functions outperforms the unweighted kernel matrix sum.

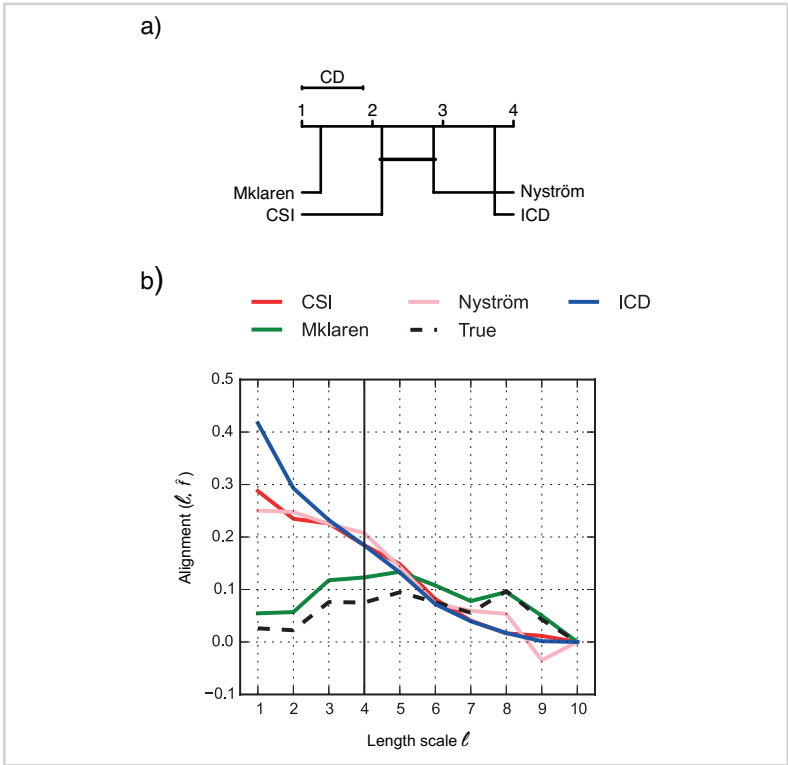


Figure 7.5

a) Critical distance diagram of average ranks at significance level $P < 0.05$. b) Visualization of different model fits for a sample data set. The y-axis represents the alignment defined in Eq. 7.3. The true substring length $\ell = 4$ used to generate the data is marked with a vertical line.

7.4.2 *Predicting RNA-binding protein binding affinities*

Finally, we evaluated the methods on a real biological data with thousands of RNA sequences of length $L = 30$, along with (real number) affinities of nine different RNA-binding proteins (data sets), similar to the original setup [61]. In total, there are 18 data sets, 2 replicates for each protein, with the total number of sequences n on the order of tens of thousands. We used $n_{\text{tr}} = n_{\text{va}} = 3000$ randomly selected sequences for training and validation. All the remaining sequences were used for testing. The training-validation-testing process was repeated 30 times. For all methods, the rank was set to $r = 10$ and the look-ahead parameter was set to $\delta = 10$ (Mklaren and CSI).

The results are shown in Table 7.3. Expectedly, the supervised methods Mklaren

and CSI achieve lowest average RMSE on all data sets. The differences are slight, but statistically significant. Mklaren achieves the minimal average RMSE in 16 out of 18 data sets. By ranking the results, the RMSE achieved by Mklaren is significantly lower than that of CSI ($P = 0.0028$, Wilcoxon paired signed-rank test). In summary, low-rank approximations are suitable choices for data sets where the objects are described only in terms of the kernel matrix, where Mklaren achieves competitive performance.

Table 7.3

Root-mean squared error (RMSE) on the RBP binding affinity prediction data sets using low-rank matrix reconstruction on string kernels. The lowest RMSE in each data set is shown in red.

	n	Mklaren	CSI	Nyström	ICD
U1A (set B)	15192	*0.836±0.010*	*0.836±0.012*	0.839±0.013	0.847±0.012
U1A (set A)	15294	*0.818±0.009*	*0.818±0.012*	0.825±0.010	0.830±0.012
VTS1 (set B)	28115	*0.965±0.008*	*0.965±0.007*	0.968±0.006	0.970±0.006
VTS1 (set A)	28542	*0.965±0.007*	0.966±0.007	0.967±0.006	0.969±0.006
Fusip (set B)	31148	*0.927±0.011*	0.929±0.008	0.939±0.007	0.943±0.010
Fusip (set A)	31300	*0.927±0.009*	0.932±0.009	0.940±0.007	0.941±0.010
RBM4 (set B)	34288	*0.790±0.024*	0.811±0.038	0.834±0.021	0.878±0.032
RBM4 (set A)	34628	*0.794±0.022*	0.812±0.024	0.834±0.015	0.877±0.022
SLM2 (set B)	37588	*0.871±0.010*	*0.871±0.011*	0.881±0.008	0.887±0.011
SLM2 (set A)	38002	*0.875±0.011*	0.884±0.012	0.891±0.010	0.899±0.012
PTB (set B)	39536	*0.859±0.004*	0.861±0.006	0.864±0.005	0.866±0.006
PTB (set A)	39822	*0.862±0.004*	*0.862±0.004*	0.866±0.004	0.869±0.006
SF2 (set B)	40774	*0.812±0.021*	0.818±0.017	0.839±0.011	0.854±0.024
SF2 (set A)	41197	*0.808±0.014*	0.823±0.019	0.829±0.011	0.851±0.011
HuR (set B)	41625	*0.758±0.007*	0.761±0.006	0.779±0.008	0.775±0.012
HuR (set A)	41864	0.756±0.009	*0.753±0.007*	0.772±0.010	0.769±0.012
YB1 (set B)	45397	*0.957±0.006*	0.958±0.004	0.960±0.004	0.964±0.004
YB1 (set A)	45857	0.964±0.005	*0.962±0.005*	0.965±0.005	0.967±0.005

7.5 Compactness of approximations

The main difference of Mklaren compared to the established kernel matrix approximation methods is simultaneous approximation of multiple kernels and the selection

criterion, which guarantees that all the basis functions are equally correlated with the current residual at each each step. In this section, we evaluate the minimal rank, necessary to achieve equivalent performance to full-rank multiple kernel learning method L2-KRR, optimizing both the kernel weights and regression coefficients. We therefore seek compact approximations, in the sense of finding the minimal approximation rank such to achieve the performance equivalent to a full-rank, highly optimized method. We performed the comparison on 28 public regression data sets provided with *Knowledge Extraction Evolutionary Learning* regression tool[3] [87]. The experimental setup mimics the one of Bach and Jordan [34].

We have used $p = 10$ exponentiated-quadratic kernels (Eq. 5.2) with varying length scale/bandwidth hyperparameters γ within the $\{10^{-3}, 10^{-2}, ..., 10^{5}, 10^{6}\}$. The performance was assessed using 5-fold cross-validation as follows. Up to 3000 data points were selected randomly from each data set. For each random split of the data set, a training set containing 60% of the data was used for kernel matrix approximation and regression model parameters inference. A *validation set* containing 20% of the data was used to select the regularization parameter λ from $\{0, 10^{-5}, 10^{-4}, ..., 10^{1}\}$. All variables were standardized and the targets $\mathbf{y}$ were centered. The look-ahead parameter δ was set to 10 for Mklaren, CSI, RFF and RFF-NS. The final performance was measured with root mean square error (RMSE), obtained on the test set with the remaining 20% of the data.

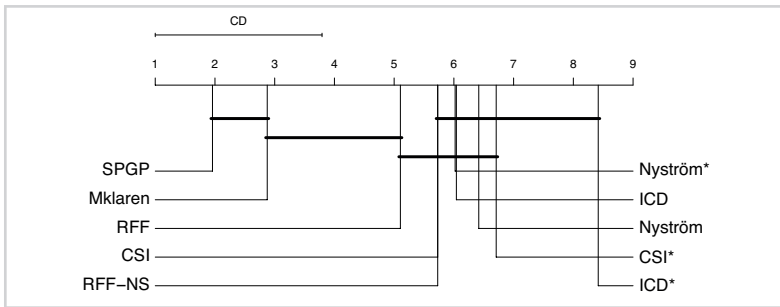


Figure 7.6

A critical distance plot. The positioning on the line denotes the average ranking of the methods in Table 7.4. The thick black line represents the critical distance at significance level $P < 0.05$.

Table 7.4 shows the minimal setting of r where the performance is at most one standard deviation away from the performance obtained by L2-KRR. The differences

[3] Available at http://sci2s.ugr.es/keel/category.php?cat=reg

Table 7.4

Minimal setting of rank r at which the performance of each method is at most one standard deviation away from the target performance of L2-KRR. The data sets (rows) where none of the methods achieved the target were removed from the comparison (data sets *mv*, *pole*, *stock*, *concrete*).

n	Mklaren	CSI	ICD	Nyström	CSI*	ICD*	Nyström*	RFF	RFF-NS	SPGP
diabetes 43	*2*	*2*	*2*	*2*	10	10	10	2	2	2
machineCPU 209	*2*	*2*	*2*	*2*	10	10	10	2	4	2
baseball 337	*9*	49	44	30	60	-	30	-	-	2
dee 365	*4*	7	7	21	10	40	10	4	25	2
autoMPG6 392	*2*	9	10	12	10	30	10	3	3	2
autoMPG8 392	*2*	8	12	14	10	20	10	7	8	2
ele-1 495	*2*	*2*	4	*2*	10	20	10	2	2	2
forestFires 517	*2*	*2*	*2*	*2*	10	10	10	2	2	2
laser 993	*19*	29	29	45	30	50	30	23	19	5
mortgage 1049	*43*	-	-	-	70	80	70	-	-	8
treasury 1049	*11*	29	27	46	30	30	20	-	-	3
ele-2 1056	*9*	46	69	-	20	40	20	8	20	3
friedman 1200	*96*	-	-	-	-	-	-	-	-	14
wizmir 1461	*16*	-	-	-	40	-	40	-	-	2
wankara 1609	*25*	-	-	-	50	-	50	-	-	2
plastic 1650	16	11	12	23	20	30	*10*	2	2	3
quake 2178	*2*	*2*	*2*	*2*	10	10	10	2	2	2
anacalt 3000	*2*	*2*	*2*	*2*	10	10	10	2	2	2
abalone 3000	*14*	44	38	44	30	60	30	10	15	3
california 3000	*46*	-	-	-	90	-	70	-	-	10
compactiv 3000	*32*	-	-	-	100	-	80	-	-	2
elevators 3000	*49*	-	-	-	-	-	-	-	-	2
house 3000	48	-	-	*45*	-	-	70	-	-	3
tic 3000	*2*	*2*	*2*	*2*	10	10	10	2	2	2

in ranks among all evaluated methods are shown in Table 7.4 and are statistically significant ($P < 2.2 \times 10^{-16}$, Friedman rank-sum test). The results are summarized on

Fig. 7.6 using the average ranking critical distance plot [59].

The best two methods are SPGP and Mklaren, whose rankings are statistically indistinguishable at statistical significance threshold of $P < 0.05$ (Friedman test), although SPGP consistently achieves the minimal rank. Expectedly, Mklaren, RFF and RFF-NS perform comparably, due to selection of basis functions from a given set of kernels centered on the training set. In all the examples, only Mklaren and SPGP achieve the goal within the given rank constraints. There is a larger gap between Mklaren and other kernel matrix approximation methods, attributable to supervised selection of inducing points from multiple kernels, and the difference in function spaces, discussed above in Sections 6.10 and 7.3.

The difference in rankings of Mklaren and CSI (next best kernel matrix approximation method) is statistically significant when examined independently (Wilcoxon paired signed-rank test, $P = 0.009$). In general, the methods using the unweighted sum outperform the methods using individual kernel approximation (CSI*, ICD*, Nyström*). Adaptable decision on which kernel to approximate next, used by Mklaren, outperforms fixed decompositions of all p kernels when provided with comparable rank constraints.

In summary, Mklaren performs best out of general kernel matrix approximation methods based on inducing point selection and is comparable to highly optimized methods for exponentiated-quadratic kernels. Importantly, it retains the general applicability for arbitrary kernels. Overall, the results confirm the utility of the LAR criterion in selecting not only the inducing points, but also the kernels to be approximated and suggest Mklaren as the method of choice when competitive performance in very low dimensional feature spaces is desired. Furthermore, the kernels that are not included in the decomposition can be discarded. This point is discussed further in the next subsection.

7.6 Comparison of MKL methods on rank-one kernels

The comparison of Mklaren to multiple kernel learning methods using the full kernel matrix is challenging as it is unrealistic to expect improved predictive performance with low-rank approximation methods. Although the restriction to low-rank feature spaces may result in implicit regularization and improved performance, the difference in implicit dimension of the feature space makes the comparison difficult [39].

We focus on the ability of Mklaren to select from a set of kernels and taking into ac-

count the implicit correlations between the kernels. We build on the empirical analysis of Cortes et al. [75], who used four sentiment analysis data sets compiled by Blitzer et al. [88]. In each data set, the examples are user reviews of products and the target is the product rating in the integer interval 1..5. The features are counts of the 4000 most frequent unigrams and bigrams in each data set. Each feature was represented by a rank-one kernel, thus enabling the use of multiple kernel learning for feature selection and explicit control over the feature space dimension. The data sets contain a moderate number of examples: *books* ($n = 5501$), *electronics* ($n = 5901$), *kitchen* ($n = 5149$) and *dvd* ($n = 5118$). The splits into training and test set were provided by the data set authors.

We compared Mklaren with state-of-the-art multiple kernel learning methods, based on maximizing centered kernel alignment [75]. The Align method infers the kernel weights independently, while AlignF and AlignFC consider the between-kernel correlations when maximizing the alignment. The combined kernel learned by all three methods was used with the kernel Ridge regression model. The Align method is linear in the number of kernels (p), while AlignF and AlignFC are cubic as they include solving an unconstrained (AlignF) or a constrained QP (AlignFC).

When testing for different ranks r, the features were first filtered according to the descending centered alignment metric for Align, AlignF, AlignFC prior to optimization. When using Mklaren, the r pivot columns were selected from the complete set of 4000 features. The parameter δ was set to 1. Note that in this scenario Mklaren is equivalent to the original LAR algorithm, thus excluding the effect of low-rank approximation and comparing only the kernel selection part. This way, the same dimension of the feature space was ensured.

The performance was measured via 5-fold cross-validation. At each step, 80% of the training set was used for kernel matrix approximation (Mklaren) or determining kernel weights (Align, AlignF, AlignFC). The remaining 20% of the training set was used for selecting regularization parameter λ from the set $0, 10^{-3}, 10^{-2}, ...10^{3}$, and the final performance was reported on the test set as RMSE.

The results for different settings of r are shown in Table 7.5. For the lowest settings of r, Mklaren outperforms all four other MKL methods that assume full kernel matrices, as the RMSE is more than one standard deviation away from that of any other methods. Mklaren outperforms the all full-rank multiple kernel methods at ranks $r = 10, 20, 40$, and performs best in on *dvd* data set at $r = 80$, *kitchen* and *dvd* at

Table 7.5

Mean RMSE on the test set for evaluated MKL methods. The columns represent increasing rank r, which is equal to the number of kernels/features included.

books	10	20	40	80	160
Uniform	1.568±0.002	1.570±0.001	1.546±0.007	1.509±0.021	1.457±0.009
Align	1.520±0.009	1.503±0.009	1.465±0.007	1.420±0.012	1.394±0.007
AlignF	1.505±0.017	1.472±0.009	1.446±0.009	1.418±0.005	1.405±0.006
AlignFC	1.501±0.013	1.464±0.008	1.427±0.007	1.403±0.003	1.388±0.006
Mklaren	*1.467±0.008*	*1.429±0.016*	*1.403±0.014*	*1.401±0.020*	*1.368±0.007*
dvd	10	20	40	80	160
Uniform	1.582±0.002	1.583±0.003	1.561±0.007	1.551±0.015	1.511±0.005
Align	1.533±0.017	1.489±0.006	1.475±0.015	1.428±0.011	1.416±0.010
AlignF	1.487±0.005	1.455±0.005	1.410±0.009	1.392±0.008	1.377±0.008
AlignFC	1.476±0.011	1.444±0.010	1.401±0.003	*1.382±0.012*	1.376±0.010
Mklaren	*1.447±0.013*	*1.405±0.009*	*1.387±0.006*	1.385±0.007	*1.373±0.002*
electronics	10	20	40	80	160
Uniform	1.573±0.003	1.564±0.006	1.553±0.010	1.517±0.007	1.395±0.004
Align	1.463±0.016	1.428±0.039	1.384±0.060	1.327±0.006	1.327±0.005
AlignF	1.460±0.003	1.424±0.017	1.345±0.010	1.330±0.011	1.313±0.009
AlignFC	1.458±0.004	1.381±0.019	1.338±0.006	*1.308±0.007*	*1.302±0.008*
Mklaren	*1.394±0.011*	*1.352±0.017*	*1.321±0.008*	1.325±0.014	1.306±0.003
kitchen	10	20	40	80	160
Uniform	1.580±0.001	1.561±0.009	1.545±0.001	1.494±0.002	1.395±0.018
Align	1.432±0.014	1.412±0.013	1.350±0.006	1.320±0.011	1.308±0.021
AlignF	1.448±0.009	1.385±0.021	1.320±0.005	1.298±0.006	1.280±0.005
AlignFC	1.446±0.009	1.371±0.022	1.319±0.009	*1.291±0.005*	*1.269±0.005*
Mklaren	*1.363±0.024*	*1.318±0.005*	*1.295±0.014*	1.302±0.008	1.285±0.005

$r = 160$. On all four data sets, the performance appears to saturate at $r = 160$ and does not improve significantly at $r = 320$ (not shown due to space limitations). The

greedy kernel and pivot selection used by Mklaren criterion considers implicit correlations between kernels. However, there is an important difference in computational complexity. Note that Mklaren is linear in the number of kernels p, which presents a practical advantage when dealing with a large number of kernels. A comparison of run times is performed in Section 7.6.

Finally, we compare the methods with respect to feature selection on the *kitchen* data set. Each of the methods Mklaren, Align, AlignF, and AlignFC infers the ordering of kernels. With Mklaren, the ordering is obtained as the pivot columns corresponding to kernels are iteratively added to the model. With alignment-based methods, we use the order indicated by the kernel weight vector. In Fig. 7.7, we display the top 40 features obtained from each ordering, shown as words on the vertical axis. As each feature is added, we infer the ordinary least-squares solution $\mathbf{w}_{\mathrm{OLS},i}$, which uses all features up to i. The arrows below each word at step i indicate the sign of the corresponding weight in $\mathbf{w}_{\mathrm{OLS},i}$. Intuitively, the slope (change in explained variance) is higher for features corresponding to words associated to strong sentiments. This is most notable for words such as *great*, *good*, *love*, etc. Not surprisingly, the order in which features are added to the model critically influences the explained variance. Here, Mklaren outperforms the alignment-based methods. Due to the similarity of LAR and Lasso (L_1) regression , the features strongly correlated to the response are identified early, irrespective of their magnitude [36]. On the other hand, the centered alignment appears to be biased towards words with a high number of nonzero entries in the data set, such as propositions. Moreover, the words associated to negative or positive sentiments are approximately balanced, according to the signs in $\mathbf{w}_{\mathrm{OLS},i}$. These results confirm Mklaren can also be used for model interpretation.

7.7 Empirical execution times

Finally, we measured the training time of the evaluated low-rank kernel approximation and MKL algorithms. We generated random data sets as described in Section 7.2, using exponentiated-quadratic kernels, rank $r \in \{5, 10, 30\}$, number of kernels $p \in \{1, 10\}$, and the number of data points $n \in \{10^2, 10^{2.5}, 10^3, ..., 10^{5.5}, 10^6\}$. The regularization parameter was set to $\lambda = 0.1$ and the look-ahead parameter was set to $\delta = 10$ for Mklaren, CSI and RFF. All methods were presented with an $\mathbf{X} \in \mathbb{R}^{n \times d}$ input matrix, a $\mathbf{y} \in \mathbb{R}^n$ target vector and $k_1, k_2, ..., k_p$ kernel functions. The time to compute the full-kernel matrix was included in the total training time for all full-rank methods and

CSI (see Section 7.1 for explanation). The processes for each tested algorithm were allowed memory resources of 512 GB RAM and a time budget of 3600 seconds before being stopped.

Timings of different scenarios for varying n are shown in Fig. 7.8. For the most resource-consuming scenario, $d = 100$ and $r = 30$, the exact times are shown in Table 7.6. For up to $10^{2.5} = 316$ examples, the full-rank MKL methods are comparable to low-rank approximations. SPGP is least efficient in this case due to optimization of inducing points overhead. For large data sets, only low-rank matrix approximations met the time constraints, with RFF being most efficient due to the absence of explicit computation of the kernel function. Mklaren runs in linear time complexity and an takes 3.9-5.1 times more time in comparison to Nyström due to look-ahead steps associated to each of the $p = 10$ kernels. Note that the run time associated to CSI could be improved by a more efficient implementation, as that provided by the authors is defined for the full kernel matrix input, which must be computed prior to approximation.

Similarly, we evaluate the performance of the methods in multiple kernel learning context with $n = 1000$, $d = 100$ and $r = 30$, varying number of kernels p. The exact timings are shown in Table 7.7. The methods with quadratic complexity in the number of kernels (AlignF, AlignFC) or SPGP exceeded the given or memory constraints when the number of kernels exceeded 100. Mklaren is second most efficient up to $p = 31$, after which the overhead in selectively evaluating the kernel function favours pre-computation of the full kernel matrix.

With these results, we empirically demonstrate the scalability of Mklaren to large data sets in comparison to related methods. In practice, the savings could be shown more explicitly by computing effective ranks (related to L2-KRR performance, similar to Section 7.5), however such experiment is severely limited by time and memory constraints associated to computing the full-kernel matrix for large n.

7.8 Summary of the results on approximate MKL

Subquadratic complexity in the number of data points is essential in large-scale application of kernel methods. Learning the kernel matrix efficiently from the data and the selection of relevant portions on the data early reduces the time and storage requirements. Using a greedy low-rank approximation to multiple kernels, we achieve linear complexity in the number of kernels and data points without sacrificing the

Table 7.6

Timings (in seconds) for rank $r = 30$, input space dimension $d = 100$, number of kernels $p = 10$ and number of data points n from 100-1000000 (in columns). Cases where processes exceeded 1 hour or memory limits are marked with a dash (-). The rows are sorted by criteria: 1) number of cases not exceeding one hour and 2) average rank in remaining cases.

method / n	100	316	1000	3162	10000	31622	100000	316227	1000000
RFF	0.03	0.08	0.12	0.26	0.73	2.45	8.86	32.7	105.22
ICD	0.08	0.17	0.39	1.08	3.53	10.63	36.4	117.23	370.54
Nyström	0.09	0.18	0.41	1.17	3.59	11.53	39.73	136.57	433.78
Mklaren	0.35	0.73	2.11	5.87	17.05	53.69	183.74	637.17	2070.78
SPGP	3.47	4.06	9.13	14.76	21.32	90.28	387.06	1011.26	3202.28
CSI	0.58	1.31	3.79	17.19	101.27	246.11	-	-	-
Uniform	0.07	0.44	4.73	44.31	593.77	-	-	-	-
L2-KRR	0.11	0.55	8.2	115.83	-	-	-	-	-
Align	0.09	0.59	9.66	158.17	-	-	-	-	-
AlignF	0.29	2.78	52	1461.83	-	-	-	-	-
AlignFC	0.31	2.83	52.46	1463.16	-	-	-	-	-

Table 7.7

Timings (in seconds) for rank $r = 30$, input space dimension $d = 100$, and number of data points $n = 1000$ and number of kernels p from 100-316 (in columns). Cases where processes exceeded 1 hour or memory resources are marked with a dash (-).

method / p	10	17	31	56	100	177	316
RFF	0.11	0.16	0.25	0.35	0.85	1.29	2.27
Uniform	4.59	6.09	5.57	7.36	10.54	16.34	27.07
Mklaren	1.97	2.86	4.93	8.42	14.27	23.98	41.4
L2-KRR	7.58	7.99	8.95	10.61	13.68	19.75	32.84
Align	9.3	13.08	18.91	30.06	46.66	79.9	142.59
SPGP	8.92	13.91	29.48	69.56	-	-	-
AlignF	51.96	128.55	384.29	1194.68	-	-	-
AlignFC	52.4	129.33	384.72	1190.79	-	-	-

consideration of between-kernel correlations.

When used with low-rank approximations, the unweighted sum of kernel matrices is not equal to concatenation of feature spaces induced by the kernels. Sampling the inducing point-centered basis functions only from the relevant kernels enables using

different hyperparameters in different regions of the input space. This property proved important both in matching the output functions in one-dimensional, continuous signals, string data sets, as well as general multidimensional regression data sets.

We show how the least-angle criterion can improve the regression function approximation by exploring regions of the input space in a more conservative manner than greedy approaches. In this sense, it can be seen in similar light to orthogonality regularization used in general matrix factorization approaches, such as iONMF, presented in the first part of the book.

Mklaren is a flexible approach for multiple kernel regression. It is kernel function independent, with data assumed to be accessed only through evaluation of the kernels. Nevertheless, its performance is comparable with methods that are optimized for specific kernels or types of input spaces. Natural extensions of the approach would include different target tasks, such as classification, ranking, or general linear models. If the used kernels are differentiable with respect to hyperparameters, the latter could also be optimized at the time of approximation, by appropriate modification of the inducing point selection heuristic. With the abundance of different data representations, we expect kernel methods to remain ubiquitous in machine learning applications.

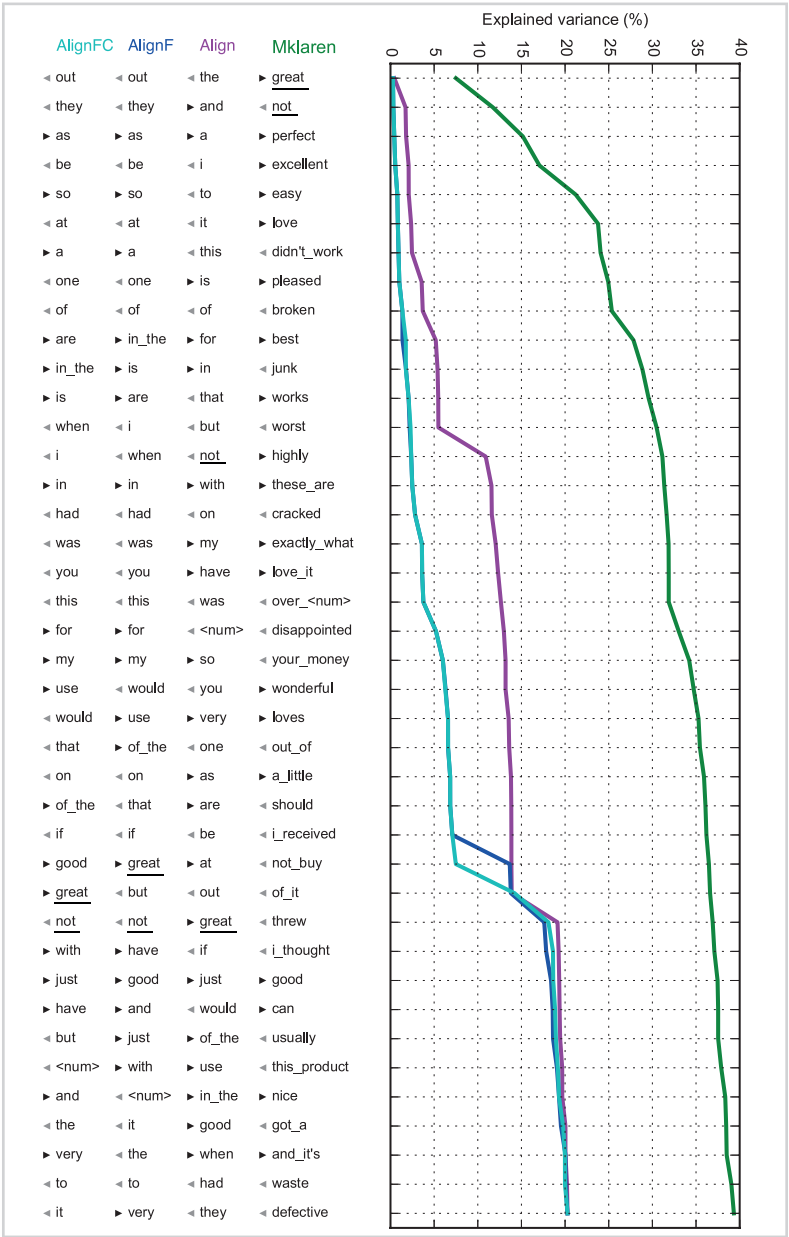


Figure 7.7

Increase in explained variance upon incrementally including features to an ordinary least-squares model. The order of features is determined by the magnitude of kernel weights for Align, AlignF and AlignFC or the order of selection by Mklaren. Arrows indicate positive (black) or negative (gray) sign of the feature in the model weight vector upon inclusion. Underscored are words "great" and "not", which significantly alter the explained variance when discovered by the Align, AlignF and AlignFC models.

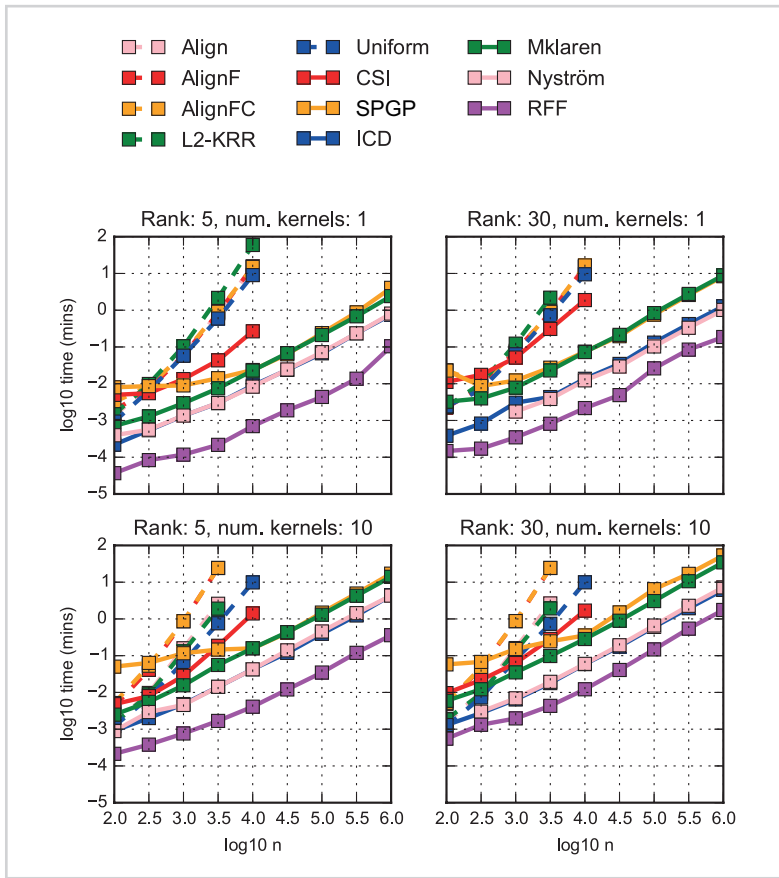


Figure 7.8

Timings (in minutes) for rank $r \in \{5, 30\}$ (in columns), number of kernels $p \in \{1, 10\}$. Both the time and the number of data points n are shown in $\log_{10}$ scale.

Conclusion

8

The inclusion of multiple, heterogeneous data sources often implies machine learning with large data sets. The modeling necessarily includes a notion of data approximation to enable tractable computation and storage. In this work, we look at two different modeling paradigms, non-negative matrix factorization and approximate multiple kernel learning, which share similar technical concepts associated to exploration of the space of models. We conclude by giving a high level view of these similarities from a general perspective.

Subject to no additional constraints provided, multiple kernel learning and multiple matrix factorization can be reduced to a straightforward concatenation of multiple features describing the data points. We show how including either different model capacity-limiting constraints - orthogonality regularization, or the order of operations to compute the kernel matrix approximation yield different spaces of models than the previously mentioned concatenation. This enables selective exploration of the input spaces, by discovering patterns present at varying magnitudes and selecting the basis functions of different parameters at different regions of the input space. Additionally, this notion favours interpretable models, either due to orthogonal patterns with little shared information, or the basis function induced from different kernels revealing some information about the problem of interest. By means of experimentation on real and synthetic datasets, we demonstrate that limiting the model capacity can improve the model performance on previously unseen data points or at least retain a statistically equivalent performance comparing to models based on complete (not approximate) data.

The presented work presents many further avenues for extending the algorithms to different modelling scenarios; for example, it would be interesting to tailor the (kernel) matrix approximation methods for data assuming non-Gaussian distributions (multinomial data / classification, count data, bounded data) - general linear models, or models with additional hierarchical structure. Also, the work can be extended to models based on different cost functions, such as support vector machines, or to couple the least-angle regression criterion with the efficiency of Random Fourier Features of multiple kernels.

The benefits of including of multiple, circumstantial data sources in machine learning solutions also come with a warning notice, in sense of potential to occlude the true signal of interest. By exploring some alternative ways of limiting the model capacity, provided by orthogonality and least-angle regression, we achieve feasible models for

linear and non-linear regression when multiple patterns are expected to contribute to the target output signal. Thus, by stating explicit assumptions about the output functions naturally provides tractable model inference both in non-negative matrix factorization and multiple kernel learning, which are two possible modeling paradigms of choice when interpretable decisions are required.

A brief introduction to RNA biology

A

In this section, we provide a short introduction on RNA and RNA-binding proteins, that are one of the main themes and motivation for development of specific machine learning methods in this work.

The Ribonucleic acid (RNA) molecules are one of the main carriers of regulatory information in the cell. The central dogma molecular biology (DNA → RNA → protein) places RNA as an intermediate, flexible entity between fixed and stable hereditary information (DNA) and proteins as the machinery enabling key processes: metabolism, immune response and chemical reaction catalysis in general. RNA is composed of a singled-stranded chain of nucleotides adenine (A), cytosine (C), guanine (G) and uridine (U), in a way similar to the double-stranded DNA. The process of conversion of DNA to RNA is termed *transcription*, while the process converting RNA to protein is known as *translation*.

At each level in the DNA → RNA → protein chain, there exist various types of interactions: protein-DNA interaction (regulation of gene expression), protein-protein interaction, RNA-RNA interactions and protein-RNA interaction, where all the interactions play an important mechanistic role on the path from DNA to the downstream processes. The traditionally well-understood role of DNA is the storage and multiplication of hereditary information passed from one subject of the species to the next. From an evolutionary perspective, DNA needs to be inherently robust to environmental perturbations. In contrast, the role of RNA is to provide an organism with plasticity - the ability to react to sudden changes in the environment on significantly shorter time scales.

The main focus of the preset work are protein-RNA interactions, which can be assayed with relatively recent next-generation sequencing (NGS) protocols and bring numerous challenges in computational data analysis, interpretation and prediction. The integration with a multitude of related available data sources provides a platform for biology, exploratory data analysis, computation and machine learning.

Protein-RNA interactions underpin important cell functions related to the regulation of gene expression, regulation of expression of gene isoforms (splicing; in eukaryotes [89]), nuclear export of proteins, and RNA processing [90]. The differing modes of protein-RNA interactions as well as the malfunctions are related to cell differentiation, and many diseases commonly related to aging and motoric, neuronal or sensory disabilities [91, 92].

The role of RNA and RBPs in gene expression is illustrated on A.1. A gene, the

basic unit of hereditary information in all organism is a defined, contiguous location a genome of an organism. It's DNA sequence contains a blueprint for a protein, which is to perform different functions. In eukaryotes, the path from a gene to a protein is quite complex. A gene can be further split into *exons*, the parts that end up in a protein, and *introns*, the parts that do not. This process is not always deterministic and the definition of exons and introns is carried out by RNA-binding proteins in a process called *splicing*. This is one on the main drivers of diversity in organism and an important mechanism that enables rapid changes in gene expression as a response to sudden changes in the environment.

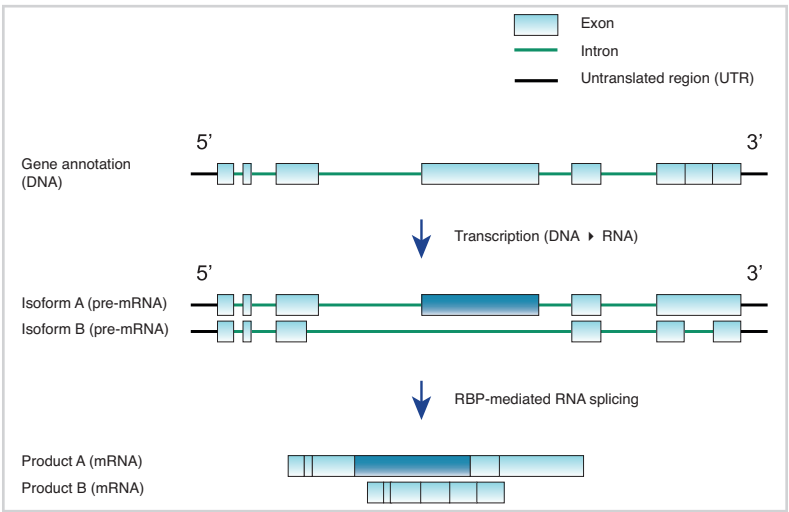


Figure A.1

A high-level schematic of gene expression. A gene is represented as a contiguous DNA sequence at a defined location in a genome of an organism. An eukaryotic gene is composed of untranslated regions, exons and introns. After RBP-mediated processing, the final gene products can vary in composition.

A distinctive feature of RNA-binding proteins are the various RNA-recognition domains, the subparts of the proteins that both recognize and directly interact with the RNA. Specificity is achieved by short, signature RNA sequences, called RNA motifs, recognized by the RNA-binding domains. The RNA motifs can be of varying length and non-deterministic sequence content but nevertheless present a confident marker of potential RBP binding sites, and can be identified both with *in vitro* and *in vivo* experimental protocols [93, 94]. For RBPs that play a key mechanistic part in gene expression, RNA splicing and processing, the positioning within the gene also plays an important role [95]. The interaction between RBPs and the RNA can also be indirect,

with many cases of RBPs being able to harbour other proteins in the close proximity of the target RNA, resulting in a cooperative or competitive mode of action [96].

Various protocols exist for capturing and mapping the interactions between RBPs and the RNA, varying in resolution, accuracy and cost [97]. One of the most common approaches is crosslinking and immunoprecipitation (CLIP), adopted by protocols HITS-CLIP, iCLIP and PAR-CLIP. A very brief description of the main processing steps follows.

The data acquisition in CLIP-based protocols starts by treating the cells with UV light in order to induce covalent bonds between RNA and RBPs (Fig. A.2, step 1). The RBP-RNA complexes are isolated via protein-specific antibodies and subsequently cleaved such that only the part of a proteins interacting with the RNA remains. Reverse transcription of the RNA fragments is then stopped by the remaining fragment of the RBP, resulting in RNA fragments that terminate on the interacting nucleotide (Fig. A.2, step 2). By mapping the obtained fragments back to the reference genome, a binding map is obtained for a protein of interest (Fig. A.2, step 3). The output of the experimental assay is a protein-specific binding signal. The computational data analyses include quality control, statistical analysis and modeling (Fig. A.2, step 4).

Expectedly, the data acquisition process is error prone and influenced by intrinsic noise factors, such as the variability in gene expression, mappability of the genomic sites, availability of correct reagents and more. To get a reliable set of potential binding sites, one must include additional information in the analysis, such as the sequence affinity, ideally obtained by independent measurement protocols (see e.g. Ray et al. [93]). To this end, the development of computational methods to integrate multiple data sources presents an important component of the field.

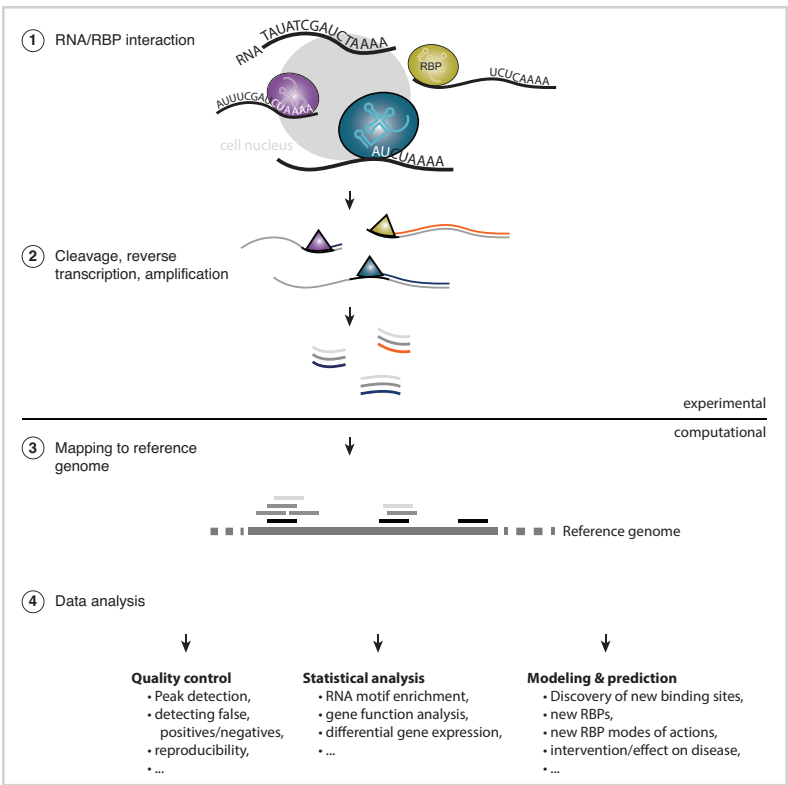


Figure A.2

A common RNA-binding protein (RBP) interaction analysis steps. Contemporary experimental protocols enable the capture of bound RNA-RBP complexes (step 1). The complexes are subsequently cleaved such that only a part of the protein in direct contact with the RNA remains. The RNA close to the interaction site is then reverse transcribed and amplify to produce a sequencing library of short RNA reads (step 2). The data is then processed computationally by first mapping the reads to the reference genome, producing an RBP binding map (step 3). Subsequent analyses include quality control, statistical analyses and modeling.

Details on derivations and algorithms

B

This section contains related work, auxiliary details, proofs and derivations that would otherwise clutter the main text. The mathematical notation used throughout the book is listed in Table B.1.

B.1 Low-rank matrix approximation

B.1.1 Notions of error

The matrix 2-norm. The matrix norm is defined as induced Euclidean vector norm for matrices. For a matrix $\mathbf{A}$ the 2-norm is defined as

$$||\mathbf{A}||_2 = \max_{||\mathbf{x}||=1} ||\mathbf{A}\mathbf{x}|| = \sigma_{\max}. \tag{B.1}$$

It represents the maximum amount for which the matrix $\mathbf{A}$, when treated as a linear operator, can stretch a unit length vector $\mathbf{x}$. The value $\sigma_{\max}$ is the maximum singular value of $\mathbf{A}$ and is equal to $\sqrt{\lambda_{\max}}$ — the maximum eigenvalue of $\mathbf{A}^T\mathbf{A}$.

Kullback-Leibler divergence / Maximum likelihood. The Euclidean distance / explained variance is appropriate when assuming that the data is distributed according to a normal distribution, both in terms of its domain and probability density function. A more general approach is obtained when considering general probability distributions, in practice allowing specification of distributional assumptions.

Let $p(\mathbf{x})$ be a probability density over a continuous variable $\mathbf{x} \in \mathbb{R}^d$. Assume p is approximated with another probability distribution q, possibly parametrized by a set of parameters $\boldsymbol{\theta}$ yielding $q(\mathbf{x}|\boldsymbol{\theta})$. The Kullback-Leibler divergence between p and q is defined as

$$\mathrm{KL}(p, q \mid \boldsymbol{\theta}) = -\int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x}|\boldsymbol{\theta})} d\mathbf{x}. \tag{B.2}$$

Observe that the KL divergence is non-symmetric and equal to 0 if and only if q matches p exactly. In practice, it is unfeasible to evaluate the exact KL-divergence as p is unknown. Given a finite sample of n data points enables approximate computation of the two quantities:

$$L = \sum_{i=1}^{n} -\log q(\mathbf{x}_i|\boldsymbol{\theta}) + \log p(\mathbf{x}_i),$$

Table B.1

A summary of notation used in the text.

Common scalars	
n	number of data points
d	dimensionality of various spaces/matrices
r	factorization / approximation rank
p	number of data matrices, kernels or kernel matrices
P	statistical significance level (p-value)
α, λ	optimization constraints
Functions	
$k(\mathbf{x}, \mathbf{x}')$	kernel function
$f(\mathbf{x})$	output regression function
J	cost function
$\mathcal{N}$	Multivariate normal (Gaussian) distribution
Vectors and matrices	
$\mathbf{x}$	data point as a vector
$\mathbf{X}$	data matrix
$\mathbf{y}$	target vector
$\mathbf{Y}$	target matrix
$\mathbf{K}$	kernel matrix
$\mathbf{x}^\star$	optimal solution to an optimization problem over vectors $\mathbf{x}$
Iteration	
$i = 1, ..., n$	iteration over data points
$j = 1, ..., d$	iteration over vector components
$q = 1, ..., p$	iteration over data sources, kernels or kernel matrices
Indexing	
$\mathbf{x}_i$	i-th row of matrix $\mathbf{X}$
$x_{ij}, \mathbf{X}(i, j)$	element at row i, column j in a matrix $\mathbf{X}$
$\mathbf{X}(:, j)$	j-th column of matrix $\mathbf{X}$
$\mathbf{X}(a : b, c : d)$	submatrix of $\mathbf{X}$
$\mathbf{K}(\mathscr{A}, \mathscr{B})$	submatrix of $\mathbf{K}$ indexed by index subsets $\mathscr{A}$ and $\mathscr{B}$
$k(\mathbf{X}_1, \mathbf{X}_2)$	kernel matrix from pairwise evaluations of the k on $\mathbf{X}_1$ and $\mathbf{X}_2$
$\mathbf{K}_1, \mathbf{K}_2, ..., \mathbf{K}_p$	different kernel matrices
$\mathbf{X}_*$	the data matrix $\mathbf{X}$ of an arbitrary number of test data points
$\mathbf{K}_*$	the kernel matrix $\mathbf{K}$ of an arbitrary number of test data points
Miscellaneous	
$C(\mathbf{K})$	centered kernel matrix $\mathbf{K}$

where the second term is independent of $\boldsymbol{\theta}$. The above expression is the data log-likelihood under the distribution q and can be maximized with respect to $\boldsymbol{\theta}$.

In context of matrix factorization, the data $\mathbf{X}$ can be considered as samples from a distribution over $\mathbb{R}^1$ and a matrix $\hat{\mathbf{X}} = \mathbf{A}\mathbf{B}^T$ as a sample from corresponding approximate distribution parametrized by $\mathbf{A}, \mathbf{B}$ [46]. In practice, to use $\mathbf{X}$ and $\hat{\mathbf{X}}$ as probability distributions, the values should be normalized to 1. This yields the following divergence:

$$D(\mathbf{X}||\hat{\mathbf{X}}) = \sum_{i=1}^{n}\sum_{j=1}^{d} x_{ij}\log\frac{x_{ij}}{\hat{x}_{ij}} - x_{ij} + \hat{x}_{ij},$$

which reduces to the Kullback-Leibler divergence when the matrices $\mathbf{X}$ and $\hat{\mathbf{X}}$ sum to 1. Stating the problem as maximum likelihood optimization enables the usage of general probability distributions, which is suitable for datasets of discrete data (Dirichlet distribution), count-based data (Poisson/Negative Binomial distribution), finite interval data (Beta distribution) and more.

We give an example optimization of the Kullback-Leibler divergence (Eq. B.2). The following strategy is based on Quasi-Newton optimization (QNO), an alternative way to circumvent the need to set a fixed step size in gradient-based optimization [48]. Briefly, as the gradient of the parameters determines the direction, the step can be set proportional to the inverse of the matrix of second gradients - the Hessian. Intuitively, the larger the change in gradient - the cost function changes more rapidly - the smaller steps are taken. Conversely, the smaller change in gradient implies more flat surfaces where the optimization can proceed more rapidly by using larger step sizes.

Again, let $\mathbf{A} \in \mathbb{R}^{n\times r}$ and $\mathbf{B} \in \mathbb{R}^{d\times r}$ and $\hat{\mathbf{X}} = \mathbf{A}\mathbf{B}^T$. This second order strategy is encoded as follows:

$$\mathbf{A}^{\text{new}} = \mathbf{A} - u(\mathbf{H}_{\mathbf{A}}^{-1}\frac{\delta D}{\delta \mathbf{A}})$$
$$\mathbf{B}^{\text{new}} = \mathbf{B} - u(\mathbf{H}_{\mathbf{B}}^{-1}\frac{\delta D}{\delta \mathbf{B}}),$$

where $\frac{\delta D}{\delta \mathbf{A}} \in \mathbb{R}^{nr}$ is a vector of model parameters and $\mathbf{H}_{\mathbf{A}} \in \mathbb{R}^{nr\times nr}$ is the matrix of second derivatives. Here, we have used $u(\cdot)$ to denote a operation of *unrolling* the vector of size nr into a matrix of size $n \times r$. The definitions of $\mathbf{B}$, $\frac{\delta D}{\delta \mathbf{A}}$ and $\mathbf{H}_{\mathbf{B}}^{-1}$

are analogous. The issues with negative parameter values are solved by the projected gradient descent (Section 2.2).

In case D is set as the KL divergence (Eq. B.2), Zdunek and Cichocki [48] propose a method that solves for a generalized alpha Amari divergence, written as

$$D_A(\mathbf{X} \| \hat{\mathbf{X}}) = \sum_{i=1}^{n} \sum_{j=1}^{d} x_{ij} \frac{(x_{ij}/\hat{x}_{ij})^{\alpha-1} - 1}{\alpha(\alpha-1)} + \frac{\hat{x}_{ij} - x_{ij}}{\alpha}$$

for a scalar parameter α, which retrieves the KL-divergence in the limit $\alpha \to 1$. Other divergences can be retrieved by setting α appropriately, see ref. Zdunek and Cichocki [48] and the references therein. The first order gradients with respect to $\mathbf{A}$ are

$$\frac{\delta D_A}{\delta \mathbf{A}} = \frac{1}{\alpha} \mathbf{B}^T \Big(1 - (\frac{\mathbf{X}}{\hat{\mathbf{X}}})^{\alpha}\Big),$$

where matrix divisions are element-wise. Furthermore, the second order gradients with respect to parameters $\mathbf{A}$ are in turn defined as matrices:

$$\frac{\delta^2 D_A}{\delta a_{ip} \delta a_{kq}} = \sum_{j=1}^{d} \frac{b_{jp} x_{ij}^{\alpha} b_{jq}}{(\sum_{q=1}^{r} a_{iq} b_{jq})^{\alpha+1}}.$$

The definitions for the derivatives with respect to $\mathbf{B}$ are defined analogously. In practice, a small magnitude diagonal matrix is added to the Hessian to guarantee invertibility (known as Levenberg-Marquardt regularization, Press [98]).

B.1.2 Principal component analysis

If the matrices $\mathbf{X} \in \mathbb{R}^{n \times d}$, $\mathbf{A} \in \mathbb{R}^{n \times r}$ and $\mathbf{B} \in \mathbb{R}^{d \times r}$ are unconstrained, there exist a unique projection to an r-dimensional subspace that satisfies two equivalent projection measures. Suppose the $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^d$ are rows of $\mathbf{X}$. Then, there exists a *principal subspace* spanned by the orthogonal set of unit vectors $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_r\} \in \mathbb{R}^d$ such that

- The variance of the projected data $\hat{\mathbf{X}}$, $\hat{\mathbf{x}}_i = \mathbf{U}\mathbf{x}_i$ is maximal.
- The distance between the original data and the projected subspace is minimal.

Both criteria lead to the same unique solution, obtained with Principal component analysis (PCA), which depends on the symmetric eigenvalue decomposition. Here, we briefly present the solution derived by the explained variance maximization [29]. Consequently, PCA ensures maximal variance by the approximated matrix and maximizes the 2-norm $||\hat{\mathbf{X}}||_2$, or, equivalently, minimizes the 2-norm of the residual $||\mathbf{X} - \hat{\mathbf{X}}||_2$ (see Eq. B.1).

The coordinate system of any vector space can be represented by an orthonormal basis $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_d\} \in \mathbb{R}^d$ by definition. The goal is to find an orthonormal subset $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_r\}$, $r < d$ such that the projected data has maximal variance.

Suppose, without loss of generality, the projection to one dimensional subspace, such that $\mathbf{U} = \mathbf{u}_1$. The variance of the projected data is defined as

$$\mathbb{E}[(\hat{\mathbf{x}} - \mathbb{E}[\hat{\mathbf{x}}])^2] =$$

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{u}_1^T\mathbf{x}_i - \mathbf{u}_1^T\bar{\mathbf{x}})^2 =$$

$$\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1,$$

where $\bar{\mathbf{x}}$ is the average row in $\mathbf{X}$ and $\mathbf{S}$ is a $d \times d$ covariance matrix:

$$\mathbf{S} = \sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T.$$

To maximize the projected variance $\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1$, $\mathbf{u}_1$ must be constrained to the set of unit vectors, i.e. $||\mathbf{u}_1||_2 = 1 \implies \mathbf{u}_1^T\mathbf{u}_1 = 1$. This is achieved by maximizing the Lagrangian:

$$\max_{\mathbf{u}_1} \quad \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T\mathbf{u}),$$

which is a quadratic optimization problem with respect to $\mathbf{u}_1$. Setting the derivative with respect to $\mathbf{u}_1$ to zero, we obtain the conditions for a stationary point:

$$\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1 \tag{B.3}$$

and so the projected variance $\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}^T\mathbf{u}_1 = \lambda_1$. From Eq. B.3 we recognize the eigenvalue decomposition, so $\mathbf{u}_1$ is an eigenvector of $\mathbf{S}$. The projected variance is given by the eigenvalue λ_1, so the optimal projection to the one-dimensional subspace is

obtained by selecting the eigenvector with the largest eigenvalue. The general solution for the optimal, r-dimensional subspace is obtained by selected the r largest eigenvalues and corresponding eigenvectors, resulting in the corresponding orthonormal matrix $\mathbf{U} \in \mathbb{R}^{d \times r}$. To express the PCA solution as matrix factorization defined in Chapter 2, we set $\hat{\mathbf{X}} = \mathbf{X}\mathbf{U}\mathbf{U}^T$ with $\mathbf{A} = \mathbf{X}\mathbf{U} \in \mathbb{R}^{n \times r}$ and $\mathbf{B} = \mathbf{U} \in \mathbb{R}^{d \times r}$.

B.2 Linear regression

B.2.1 The relation between dual and primal regression weights

A regression problem is defined given the data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and target values (outputs) $\mathbf{y} \in \mathbb{R}^n$.

Lemma. Let the parameters $\mathbf{w} \in \mathbb{R}^d$ be the solution of a regression problem, defined by the expression

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{X}^T\mathbf{y}. \tag{B.4}$$

Then, $\mathbf{w}$ can be written as a linear combination of rows in $\mathbf{X}$, i.e. $\mathbf{w} = \mathbf{X}^T\boldsymbol{\alpha}$ for some real vector $\boldsymbol{\alpha} \in \mathbb{R}^n$.

Proof. The proof follows from a direct application of the Sherman-Morrison-Woodbury lemma on the matrix inverse [83]. Specifically, the linear transform $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_d)^{-1}$ in Eq. B.4 can be expanded as

$$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_d)^{-1} = (\lambda^{-1}\mathbf{I} - \lambda^{-2}\mathbf{X}^T(\mathbf{I}_n + \lambda^{-1}\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}).$$

The right-hand side of Eq. B.4 can then be re-arranged as:

$$\mathbf{w} = \lambda^{-1}\mathbf{X}^T\Big[\mathbf{y} - \lambda^{-1}(\mathbf{I}_n + \lambda^{-1}\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X})\mathbf{X}^T\mathbf{y}\Big].$$

Indeed, $\mathbf{w}$ can be expressed as a linear combination of rows of $\mathbf{X}$, with $\boldsymbol{\alpha}$ defined as the expression in the square brackets above.

■

B.2.2 Least-angle regression

Least-angle regression (LAR) is an *active set method*, originally designed for feature subset selection in linear regression [36, 42, 99]. A column is chosen from the set of candidates such that the correlations with the residual are equal for all active variables. This is possible because all variables (columns) are known *a priori*.

Let the predictor variables $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p\}$ be vectors in $\mathbb{R}^n$, arranged in a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$. The associated response vector is $\mathbf{y} \in \mathbb{R}^n$. The LAR method iteratively selects the predictor variables $\mathbf{x}_j$, and the corresponding coefficients w_j are updated at

the same time as they are moved towards their least-squares coefficients. In the last step, the method reaches the least-squares solution.

The high-level pseudo code is as follows:

1. Start with the residual $\mathbf{e} = \mathbf{y} - \bar{\mathbf{y}}$, and regression coefficients $w_1, w_2, ...w_p = 0$.
2. Find the variable $\mathbf{x}_j$ most correlated with $\mathbf{e}$.
3. Move w_j towards its least-squares coefficient until another $\mathbf{x}_k$ has as much correlation with $\mathbf{e}$.
4. Move w_j and w_k in the direction towards their joint least-sq. coeff., until some new $\mathbf{x}_l$ has as much correlation with $\mathbf{e}$.
5. Repeat until all variables have been entered, reaching the least-sq. solution.

Note that the method is easily modified to include early stopping, after a maximum number of selected predictor variables are included. Importantly, the method can be viewed as a version of supervised Incomplete Cholesky decomposition of the *linear kernel* $\mathbf{K} = \mathbf{X}\mathbf{X}^T$ which corresponds to the usual inner product in $\mathbb{R}^p$. A more detailed description of the algorithm follows.

Assume the predictor variables are standardized and response is centered:

$$\mathbf{1}^T\mathbf{x}_j = 0 \text{ and } \|\mathbf{x}_j\|_2 = 1 \text{ for } j = 1, 2, ..., p,$$
$$\mathbf{1}^T\mathbf{y} = 0.$$

Initialize the *regression estimate* $\mathbf{f}$, the *residual* $\mathbf{e}$ and the *active set* $\mathscr{A}$:

$$\mathbf{f} = \mathbf{0}, \mathbf{e} = \mathbf{y} \text{ and } \mathscr{A} = \varnothing.$$

The LAR algorithm estimates $\mathbf{f} = \mathbf{X}\mathbf{w}$ in successive steps. Say the predictor $\mathbf{x}_i$ has the largest correlation with $\mathbf{e}$. Then, the index i is added to the active set $\mathscr{A}$ and the regression estimate and residual are updated:

$$\mathbf{f}^{\text{new}} = \mathbf{f} + \gamma\mathbf{x}_i,$$
$$\mathbf{e}^{\text{new}} = \mathbf{e} - \gamma\mathbf{x}_i.$$

The step size γ is set such that a new predictor $\mathbf{x}_j$ will enter the model after $\mathbf{f}$ is updated and all predictors in the active set as well as $\mathbf{x}_j$ are equally correlated to $\mathbf{e}$. The key parts are the selection of predictors added to the model and the calculation of the step size.

The active matrix for a subset of indices j with sign s_j is defined as

$$\begin{aligned}\mathbf{X}_A &= \left(\cdots s_j\mathbf{x}_j \cdots\right) \text{ for } j \in \mathscr{A}\\ s_j &= \text{sign}\{\mathbf{x}_j^T\mathbf{e}\}.\end{aligned}$$

By elementary linear algebra, there exists a *bisector* $\mathbf{u}_A$ — an equiangular vector, having $\|\mathbf{u}_A\|_2 = 1$ and making equal angles, less than 90 degrees, with vectors in $\mathbf{X}_A$. Define the following quantities respectively: $\mathbf{X}_A$ the active matrix, A the normalization scalar, $\mathbf{u}_A$ the bisector, and $\boldsymbol{\omega}$ the vector making equal angles with the columns of $\mathbf{X}_A$. The bisector is obtained as follows:

$$\begin{aligned}\mathbf{T}_A &= \mathbf{X}_A^T\mathbf{X}_A,\\ A &= (\mathbf{1}_A^T\mathbf{T}_A\mathbf{1}_A)^{-1/2},\\ \boldsymbol{\omega} &= A\mathbf{T}_A^{-1}\mathbf{1}_A,\\ \mathbf{u}_A &= \mathbf{X}_A\boldsymbol{\omega}_A.\end{aligned} \tag{B.5}$$

The calculation of step size γ proceeds as follows. Get the maximum vector of correlations. The active set contains variables with highest absolute correlations.

$$\begin{aligned}c_j &= \mathbf{x}_j^T\mathbf{e},\\ C &= \max_j \quad \{c_j\},\\ \mathbf{a} &= \mathbf{X}_A^T\mathbf{u}_A,\end{aligned} \tag{B.6}$$

$$\gamma = \min^+_{j\in\mathscr{A}^c} \quad \{\frac{C-c_j}{A_A-a_j}, \frac{C+c_j}{A_A+a_j}\},$$

where $\min^+$ is the minimum over positive components. By Eq. B.2.2, the change in correlations within the active set can be expressed.

$$c_j^{\text{new}} = \mathbf{x}_j^T(\mathbf{y} - \mathbf{e}^{\text{new}}) = c_j - \gamma a_j. \tag{B.7}$$

For the predictors in the active set, we have

$$|c_j^{\text{new}}| = C - \gamma A, \text{ for } j \in \mathscr{A}. \tag{B.8}$$

A variable is selected from the remaining variables in $\mathscr{A}^c$, such that c_j^{new} is maximal. Equaling Eq. B.7 and Eq. B.8, and maximizing yields $\gamma = \frac{C-c_j}{A-a_j}$. Similarly, $-c_j^{\text{new}}$ for the reverse covariate is maximal at $\gamma = \frac{C+c_j}{A+a_j}$. Hence, γ is chosen in Eq. B.6 as a minimal value for which a variable joins the active set.

B.3 Kernel methods

B.3.1 Inner product spaces

An inner product space is a vector space $\mathscr{X}$, endowed with an *inner product* function denoted $\langle \cdot, \cdot \rangle$, mapping $(\mathscr{X} \times \mathscr{X}) \mapsto \mathbb{R}$. Let $\mathbf{a}, \mathbf{b}, \mathbf{c}$ be arbitrary elements from the space $\mathscr{X}$. An inner product is any function satisfying the following properties:

1. Non-negativity.

   $\langle \mathbf{a}, \mathbf{a} \rangle \geq 0$, and $\langle \mathbf{a}, \mathbf{a} \rangle = 0 \equiv \mathbf{a} = \mathbf{0}$

2. Multiplication with a scalar. For a scalar $c \in \mathbb{R}$:

   $\langle c\mathbf{a}, \mathbf{b} \rangle = c\langle \mathbf{a}, \mathbf{b} \rangle$

3. Distributivity over sums of elements.

   $\langle \mathbf{a} + \mathbf{b}, \mathbf{c} \rangle = \langle \mathbf{a}, \mathbf{c} \rangle + \langle \mathbf{b}, \mathbf{c} \rangle$

4. Symmetry.

   $\langle \mathbf{a}, \mathbf{b} \rangle = \langle \mathbf{b}, \mathbf{a} \rangle$

A canonical example of a inner product is the mapping $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_i a_i b_i$. An inner product of an element with itself defines a *norm*, $\sqrt{\langle \mathbf{a}, \mathbf{a} \rangle} = \|\mathbf{a}\|$.

B.3.2 A simple example of (kernel) linear regression

Let $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a set of data points in a inner product space $\mathscr{X}$ of dimension d, associated with target values (outputs) $\mathbf{y} \in \mathbb{R}^n$. A mapping f from $\mathscr{X}$ to $\mathbb{R}$ can be used to predict the value y_i given an $\mathbf{x}_i$. Given a set of n example pairs $(\mathbf{x}_i, y_i)$, the problem is stated as minimization of loss over a space of functions:

$$\min_f \quad \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2. \tag{B.9}$$

In *linear regression*, $\mathscr{X}$ is a real vector space of dimension d, so $\mathscr{X}$ is $\mathbb{R}^d$. The functions f are limited to linear functions, defined by parameters $\mathbf{w} \in \mathbb{R}^d$:

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \sum_{j=1}^{d} x_j w_j. \tag{B.10}$$

Note that in this case, $\mathbf{w}$ is simply another point in the input space $\mathscr{X}$. The optimal solution for f is found as the solution to a linear system of equations

$$\mathbf{X}\mathbf{w} = \mathbf{y}, \tag{B.11}$$

where $\mathbf{X}$ is an $n \times d$ matrix (of rank $\min(n, d)$) with examples $\mathbf{x}_i$ in rows. In general, n does not equal d and the above system may be under-determined (if $d > n$) or over-determined (if $d < n$). In the former case, the system has an infinite number of solutions, while in the latter case there is no exact solution in general. In practice, the system in Eq. B.11 is replaced by a full-rank, $d \times d$ system with the *Ridge regression* solution

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{X}^T\mathbf{y}, \tag{B.12}$$

where $\mathbf{I}_d$ is the identity matrix of dimension d and $\lambda > 0$ is a scalar *regularization* parameter, which bounds the norm $\|\mathbf{w}\|$. The interpretation is as follows. The function f, determined by a point in $\mathscr{X}$, lies in the linear span of data points $\mathbf{X}$. So, $\mathbf{w}$ can be defined as a linear combination of rows of $\mathbf{X}$:

$$\mathbf{w} = \mathbf{X}^T\boldsymbol{\alpha}, \tag{B.13}$$

for an $\boldsymbol{\alpha} \in \mathbb{R}^n$. For derivation of Eq. B.13 from Eq. B.12, see Appendix B.2.1. The optimization problem can then be stated as:

$$(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}_n)\boldsymbol{\alpha} = \mathbf{y}, \tag{B.14}$$

with solution $\boldsymbol{\alpha} = (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}_n)^{-1}\mathbf{y}$. The matrix $\mathbf{X}\mathbf{X}^T$ is called the *Gram matrix* and is obtained by evaluation of *inner products* $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Thus, linear regression is defined for any inner product space.

B.3.3 Kernel-specific approximations

In this part, we present the kernel approximation that are limited either to specific, individual kernels or assume properties other than the basic properties of kernels listed in Section 5.1. Unsurprisingly, this can move the lower bound of the computational complexity below $O(nr^2)$ achieved by kernel matrix approximation methods presented in Section 5.5. Furthermore, hyperparameter optimization can be done simultaneously with kernel approximation, allowing for models of larger capacity.

B.3.4 Translation-invariant kernels and explicit feature maps

A translation-invariant or *stationary* kernel is a kernel that can be written as a function of a single-argument — e.g. the difference between the two elements, $k(\mathbf{x}, \mathbf{x'}) = k(\mathbf{x} - \mathbf{x'})$. Kernels of this type are often continuous functions. A natural idea to approximate a kernel is to derive its underlying (approximate) feature map, as defined Section 5.1.

Bochner's theorem. A continuous translation-invariant kernel $k(\mathbf{x} - \mathbf{x'})$ on a real domain $\mathbb{R}^d$ is positive definite if and only if k is the Fourier transform of a non-negative measure[1]. This means that the corresponding Fourier transform gives rise to a valid probability distribution. Letting $\zeta_{\boldsymbol{\omega}}(\mathbf{x}) = e^{i\boldsymbol{\omega}^T\mathbf{x}}$, the statement of the theorem is:

$$k(\mathbf{x} - \mathbf{x'}) = \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^T(\mathbf{x}-\mathbf{x'})} d\boldsymbol{\omega} = \mathbb{E}_{\boldsymbol{\omega}}[\zeta_{\boldsymbol{\omega}}(\mathbf{x})\zeta_{\boldsymbol{\omega}}(\mathbf{x'})]. \tag{B.15}$$

Random Fourier features. The Bochner's theorem provides a straightforward tool to generate explicit feature maps if the Fourier transform of k can be derived analytically [100]. As both k and p are real-valued functions, the complex exponentials are replaced with cosines. An explicit feature map can then be defined as $z_{\boldsymbol{\omega}}(\mathbf{x}) = cos(\boldsymbol{\omega}^T\mathbf{x} + b)$ with $\boldsymbol{\omega}$ drawn from $p(\boldsymbol{\omega})$ and b drawn uniformly from $[0, 2\pi]$, which is derived from applying basic trigonometry rules on $e^{i\boldsymbol{\omega}^T(\mathbf{x}-\mathbf{x'})}$. Equivalently, following Ton et al. [86], random features can be defined as sine-cosine pairs:

$$\Phi_{\boldsymbol{\omega}}(\mathbf{x}) = \Big(\cos(\boldsymbol{\omega}^T\mathbf{x}), \quad \sin(\boldsymbol{\omega}^T\mathbf{x})\Big).$$

To lower the variance of the estimator, r random vectors $\boldsymbol{\omega}$ are sampled from $p(\boldsymbol{\omega})$, determining the effective dimensionality of the feature space. The approximation to the kernel matrix given random vectors $\boldsymbol{\omega}_j, j = 1...r$ and a data matrix $\mathbf{X}$ is then equal to

$$\mathbf{L} = \frac{1}{2r}\sum_{j=1}^{r} \boldsymbol{\Phi}_{\boldsymbol{\omega}_j}(\mathbf{X})\boldsymbol{\Phi}_{\boldsymbol{\omega}_j}(\mathbf{X})^T.$$

Here, one needs to be able to solve an indefinite integral for the kernel defined as a function of $\mathbf{x}–\mathbf{x'}$. For example, evaluating the Fourier transform for the exponentiated-quadratic kernel results in probability distribution

[1] Measure: a continuous function defined over a set with a finite value of any definite integral. The function value is monotonically increasing with number of elements in the set, and therefore it can be interpreted as set size. Example: all probability distributions are measures over the respective domains.

$$p_{\exp}(\boldsymbol{\omega}) = (2\pi\sigma^2)^{-d/2} e^{-\frac{1}{2\sigma^2}\|\boldsymbol{\omega}\|_2^2}, \tag{B.16}$$

which is the multivariate normal distribution over $\mathbb{R}^d$, centered at the origin. The variance σ^2 is directly related to the bandwidth of the exponentiated-quadratic kernel (Eq. 5.1), with the equality $\sigma^2 = 2d\gamma$.

Feature maps of non-stationary kernels. The above approximation applies to *stationary kernels* only. The results of Samo and Roberts [101] provide a generalization based on combinations of kernels yielding approximations of non-stationary kernels. The modification to the Bochner's theorem in Eq. B.15 is as follows:

$$k(\mathbf{x}, \mathbf{x'}) = \int_{\mathbb{R}^d} p_2(\boldsymbol{\omega}_a, \boldsymbol{\omega}_b) e^{i(\boldsymbol{\omega}_a^T \mathbf{x} - \boldsymbol{\omega}_b^T \mathbf{x'})} d\boldsymbol{\omega}_a d\boldsymbol{\omega}_b,$$

where $\boldsymbol{\omega}_a$, $\boldsymbol{\omega}_b \in \mathbb{R}^d$ are two distinct random vectors, sampled from a positive-definite measure p_2 defined over $\mathbb{R}^d \times \mathbb{R}^d$. If $\boldsymbol{\omega}_a = \boldsymbol{\omega}_b$, the stationary kernel approximation in Eq. B.15 is recovered. The explicit feature map is then equal to

$$\Phi_{\boldsymbol{\omega}_{aj}, \boldsymbol{\omega}_{bj}}(\mathbf{x}) = \left(\cos(\boldsymbol{\omega}_{aj}^T \mathbf{x}) + \cos(\boldsymbol{\omega}_{bj}^T \mathbf{x}), \quad \sin(\boldsymbol{\omega}_{aj}^T \mathbf{x}) + \sin(\boldsymbol{\omega}_{bj}^T \mathbf{x})\right),$$

and the kernel matrix approximation equals

$$\mathbf{L} = \frac{1}{4r} \sum_{j=1}^{r} \boldsymbol{\Phi}_{\boldsymbol{\omega}_{aj}, \boldsymbol{\omega}_{bj}}(\mathbf{X}) \boldsymbol{\Phi}_{\boldsymbol{\omega}_{aj}, \boldsymbol{\omega}_{bj}}(\mathbf{X})^T.$$

There is a large flexibility in choosing the positive-definite measure p_2, and the simplest case is to set $p_2(\boldsymbol{\omega}_a, \boldsymbol{\omega}_b) = p(\boldsymbol{\omega}_a) q(\boldsymbol{\omega}_b)$, where p, q are measures over $\mathbb{R}^d$ and typically correspond to densities of known stationary kernels [86].

γ-homogeneous kernels. An alternative approximation to non-stationary, single-argument kernel functions, is as follows. The work of Rahimi and Recht [100] was extended by Vedaldi and Zisserman [102] from stationary to γ-homogeneous kernels.

A γ-homogeneous kernel $k_h(x, x')$ for $x, x' \in \mathbb{R}$ has the following property: for a scalar $c > 0$, $k_h(cx, cx') = c^\gamma$ for some real number γ. Setting γ to $1/\sqrt{xx'}$, enables k_h to be written as

$$(xx')^{\frac{1}{2}} k_h(\sqrt{\frac{x'}{x}}, \sqrt{\frac{x}{x'}}) =$$
$$(xx')^{\frac{1}{2}} k_h(e^{\frac{1}{2}\log\frac{x'}{x}}, e^{-\frac{1}{2}\log\frac{x'}{x}}) =$$
$$(xx')^{\frac{1}{2}} \mathscr{K}(\log x' - \log x) = (xx')^{\frac{1}{2}} \mathscr{K}(\lambda),$$

where $\mathscr{K}$ is the *signature function* and λ is a scalar parameter. Similarly as with Random Fourier Features and stationary kernels, the γ-homogeneous kernels can be transformed via discrete Fourier transform,

$$\hat{\mathscr{K}}(\lambda) = \sum_{j=-\infty}^{+\infty} \hat{\kappa}_j e^{-ijL\lambda}$$

and $\hat{\kappa}_j$ presents a discrete sampling at periods $j = 0, 1, ...$ of continuous Fourier transform of the signature function $\mathscr{K}(\lambda)$. Given a continuous spectrum $\kappa(\omega)$ of a signature function, the discrete spectrum can be obtained by sampling and rescaling, $\hat{\kappa}_j = L\kappa(jL)$ for any frequency L. A discrete feature map for a set number of components $j = 0, 1, ..., K$ and a fixed L is then obtained as:

$$\hat{\Psi}_j(x) = \begin{cases} \sqrt{\hat{\kappa}_0}, & j = 0, \\ \sqrt{2x^\gamma \hat{\kappa}_{\frac{j+1}{2}}} \cos(\frac{j+1}{2} L \log x) & j > 0 \text{ odd}, \\ \sqrt{2x^\gamma \hat{\kappa}_{\frac{j}{2}}} \sin(\frac{j}{2} L \log x) & j > 0 \text{ even}. \end{cases}$$

Example γ-homogeneous kernels are the Hellinger kernel $k(x, x') = \sqrt{xx'}$ or the intersection kernel $k(x, x') = \min(x, x')$. Their respective continuous spectra are $\kappa(\omega) = \delta(\omega)$ (Dirac's delta function) and $\kappa(\omega) = \frac{2}{\pi}\frac{1}{1+4\omega^2}$, with respective continuous feature maps $\Psi_w = \sqrt{x}$ and $\Psi_w = e^{i\omega \log x}\sqrt{x\kappa(\omega)}$.

B.3.5 Optimization of differentiable kernels

Kernels can be optimized with respect to the target task (Bishop [29], ch. 6). A common assumption is that the kernels $k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})$ are differentiable functions with respect to the input points and possible hyperparameters $\boldsymbol{\theta}$. Examples of differentiable kernels with hyperparameters are the exponentiated-quadratic, sigmoid or the polynomial

kernel presented in Table 5.1, Section 5.1. The following ideas are presented in context of Gaussian process regression and are transferable to any task based on gradient optimization.

Optimization of kernel hyperparameters. The distribution of targets $\mathbf{y}$ assuming a Gaussian process prior is presented in Eq. 5.16, Section 5.4. The covariance matrix is assumed to be $\mathbf{C} = \mathbf{K} + \sigma^2\mathbf{I}$ and depends both on the dataset $\mathscr{D}$ and kernel hyperparameters $\boldsymbol{\theta}$. The log-likelihood is then evaluated as

$$\log p(\mathbf{y}|\mathscr{D}, \boldsymbol{\theta}) = -\frac{1}{2}\log|\mathbf{C}| - \frac{1}{2}\mathbf{y}^T\mathbf{C}^{-1}\mathbf{y} - \frac{n}{2}\log(2\pi),$$

where $|\cdot|$ is the matrix determinant. The gradient with respect to the i-th scalar hyperparameter θ_i is:

$$\frac{\delta}{\delta\theta_i}\log p(\mathbf{y}|\mathscr{D}, \boldsymbol{\theta}) = -\frac{1}{2}\mathrm{tr}(\mathbf{C}^{-1}\frac{\delta\mathbf{C}}{\delta\theta_i}) + \frac{1}{2}\mathbf{y}^T\mathbf{C}^{-1}\frac{\delta\mathbf{C}}{\delta\theta_i}\mathbf{C}^{-1}\mathbf{y}. \tag{B.17}$$

The log-likelihood is generally a non-convex function and is solved by gradient ascent to find the local maxima.

Joint optimization of the inducing locations and hyperparameters. In contrast to active set sampling-based methods presented in Section 5.5, there exist alternative ways to optimize the low-rank kernel matrix. One approach, suitable for continuous input spaces are the Sparse Pseudo Input Gaussian Processes (SPGPs, [103]).

Let $\mathbf{X} \in \mathbb{R}^{n\times d}$ be the matrix of data points in the input space $\mathbb{R}^d$, and $\mathbf{U} \in \mathbb{R}^{m\times d}$ be a set of m inducing input locations in the same input space. This is different from the *active sets* $\mathscr{A}$ described for the kernel matrix approximations in Section 5.5, as their locations are not limited to the training set. The following kernel matrices are defined: $\mathbf{K}_{uu} \in \mathbb{R}^{r\times r}$ as $k(\mathbf{U},\mathbf{U})$, $\mathbf{K}_{xu}^T = \mathbf{K}_{ux} \in \mathbb{R}^{r\times n}$ as $k(\mathbf{U},\mathbf{X})$, $\mathbf{K}_{xx} \in \mathbb{R}^{n\times n}$ as $k(\mathbf{X},\mathbf{X})$. The latter matrix is to be avoided in an attempt to avoid $O(n^3)$ complexity of matrix inversion. The vector of noisy target values is denoted $\mathbf{y} \in \mathbb{R}^n$ and the true underlying function values by $\mathbf{f}$.

The authors assume existence of $\mathbf{f}_u$ — the noiseless pseudo targets at locations $\mathbf{U}$. The crucial assumption is the prior distribution on the pseudo targets:

$$p(\mathbf{f}_u|\,\mathbf{U}) = \mathscr{N}(\mathbf{0}, \mathbf{K}_{uu}), \tag{B.18}$$

implying that the mean of $\mathbf{f}_u$ is the same as the mean of the true targets $\mathbf{f}$. Moreover, this necessarily implies that the pseudo inputs $\mathbf{U}$ must be *well-spread* throughout the input region. To see why this is the case, imagine $\mathbf{U}$ being concentrated in an increasingly small region. This would cause $\mathbf{f}_u$ to be increasingly concentrated around values of f in that region. Now, for smooth functions, the probability of hitting a small region where the mean of $\mathbf{f}_u$ is exactly zero is infinitesimal. It turns out that the assumption in Eq. B.18 is crucial for gradient-based optimization of pseudo input placement. This is realized with optimizing the marginal likelihood with respect to $\mathbf{U}$ and other possible hyperparameters of the kernel(s).

The second key point is that the targets $\mathbf{y}$ are modelled with the *predictive posterior* distribution with $\mathbf{K}_{uu}$ used in place of $\mathbf{K}_{xx}$. If $\mathbf{f}_u$ would be known, the posterior would equal

$$p(\mathbf{y}|\mathbf{X}, \mathbf{U}, \mathbf{f}_u) = \mathcal{N}(\mathbf{K}_{xu}\mathbf{K}_{uu}^{-1}\mathbf{f}_u, \boldsymbol{\Lambda} + \sigma^2\mathbf{I}), \tag{B.19}$$

where $\boldsymbol{\Lambda}$ is diagonal matrix with entries equal to $\boldsymbol{\lambda} = \text{diag}(\mathbf{K}_{xx} - \mathbf{K}_{xu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{ux})$. The equation is similar to the standard Nyström approximation in Eq. 5.17, except that the pseudo targets are used. This method is referred in sparse Gaussian Processes literature as Fully-independent training conditional (FITC, Quiñonero-Candela and Rasmussen [104]).

The marginal likelihood for the targets $\mathbf{y}$ is obtained via Bayes rule, by combining the prior in Eq. B.18 with the standard likelihood in Eq. B.19. It is obtained by marginalizing out the variables $\mathbf{f}_u$, which has an closed-form expression in the case of the product of two normal distributions:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{U}, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{X}, \mathbf{U}, \mathbf{f}_u)p(\mathbf{f}_u|\,\mathbf{U})d\mathbf{f}_u = \mathcal{N}(\mathbf{0}, \mathbf{K}_{xu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{ux} + \boldsymbol{\Lambda} + \sigma^2 I). \tag{B.20}$$

The matrix $\boldsymbol{\Lambda}$ is the crucial increase in variance that forces gradient-based optimization to *explore* the input region, and makes the likelihood different than the case where the classic Nyström approximation is used in place of the full kernel matrix (Eq. 5.17). The log-likelihood from Eq. B.20 can be optimized very similarly as in B.17 with respect to $\mathbf{U}$ and $\boldsymbol{\theta}$. However, a principal limitation of SPGPs is that input space must be continuous and not too large (in the sense of dimension d).

Supplementary Information on iONMF

C

C.1 Detailed information on analyzed RBP experiments

Reference and details about all experiments used in the study are listed. Experiments for the same proteins are sorted into groups A-Q. Factor models for each experiment do not consider biological or technical replicates in the same group to eliminate bias. Depending on the experimental protocol used (PARCLIP, CLIPSEQ, iCLIP, HITSCLIP) we report number of crosslinking clusters and number of individual sites (measured as the sum of cluster lengths) for each experiment used. We used information on clusters (column Clusters) if provided by the original study. In case of experiments 18-20 and 22, we used information on individual crosslink sites (column CL sites). From individual positions (clusters or individual crosslinks), we select up to 100.000 samples with highest cDNA counts.

C.2 Details on models inferred from subsets of data sources

In this section, we present details on parameter settings (factorization rank) for factor models learned on different subsets of data sources. The model learned on the complete set of data sources $\{\mathbf{X}_{\text{CLIP}},\mathbf{X}_{\text{GO}},\mathbf{X}_{\text{KMER}},\mathbf{X}_{\text{RNA}},\mathbf{X}_{\text{RG}}\}$, with $n = 50{,}000$ training samples and rank $r = 10$ results in

$$\begin{aligned} N &= (n + \sum_q d_q) \cdot r \\ &= (50{,}000 + 1 + 3{,}030 + 101 + 505 + 25{,}865 + 39{,}560) \cdot 10 = 1{,}190{,}570 \end{aligned}$$

free parameters. This is the total number of entries in matrices $\mathbf{W}$, $\mathbf{H}_Y$ and all $\mathbf{H}_q$ of the full model. Note that the $\mathbf{X}_{\text{CLIP}}$ matrix has up to 3,030 columns, the exact number is depending on the replicate group corresponding to the selected protein. To ensure fair comparison among different combinations of data sources, we set the rank to

$$r_s = \lceil N/(n + 1 + \sum_s d_s) \rceil$$

for a subset of data sources $s \subset \{\mathbf{X}_{\text{CLIP}},\mathbf{X}_{\text{GO}},\mathbf{X}_{\text{KMER}},\mathbf{X}_{\text{RNA}},\mathbf{X}_{\text{RG}}\}$. This ensures an approximately equal number of free parameters of the factor model (column N_s in Suppl. Table C.2).

C.3 Importance of different data sources

C.3.1 Prediction accuracy of data source subsets

Each combination of RBP experiments and subset of data sources yields a factor model. In Suppl. Tables C.3,C.4 and C.5 we compare all such models on area under ROC curve (AUC), obtained via prediction of the independent hold-out test set (see main text). For each protein, we highlight the best-scoring subset of data sources.

C.3.2 Mutual information within individual data sources

Assessing the importance of individual data sources in Suppl. Tables C.3-C.5 is non-trivial since the scores obtained by a particular subsets are not independent. Nevertheless, we transformed each column of Suppl. Tables C.3-C.5 in a 31×5 binary matrix $\mathbf{B}$, corresponding to 31 subsets of 5 data sources. For each data source, we calculate the Spearman correlation coefficient between: its corresponding column in $\mathbf{B}$ and column with AUC scores. Thus, we obtain a 5×1 vector containing correlation coefficients for each protein, which we use to perform hierarchical clustering in Suppl. Figure C.1.

We observe an influence of $\mathbf{X}_{\text{KMER}}$ on sequence-dependent proteins (TIA1/TIAL1, PUM2, hnRNPs, U2AF2, FUS, etc.), while the influence is less pronounced for more region-type dependent proteins, such as those displaying bias towards introns and 3'UTRs (ELAVL, [105]), or in both exons and introns (SRSF1, [106]). On the other hand, RNA structure is most informative data source for eIF4AIII, which binds unstructured RNA ([107]).

Interestingly, all experiments performed using iCLIP show strong RNA k-mer preference, which is attributed to the individual nucleotide resolution. Protocols with lower resolution, such as CLIPSEQ, conversely are not correlated with RNA k-mers, but are rather modelled by more coarse data sources such as region type and RNA structure.

C.3.3 Clustering of RBPs based on individual data sources

We examine values of features in the coefficient matrices $\mathbf{H}_q$ for each RBP experiment. By comparing the magnitude of individual features in the modules related to positive samples (crosslink sites), we identify features determining RBP binding.

For each RBP experiment, we select the most relevant module (see main text) and normalize the corresponding row vector in $\mathbf{H}_q$. We run hierarchical clustering (Ward's

linkage) on the row vectors and display the results as heatmaps for each data source. The results are shown in Suppl. Fig. C.2-C.7.

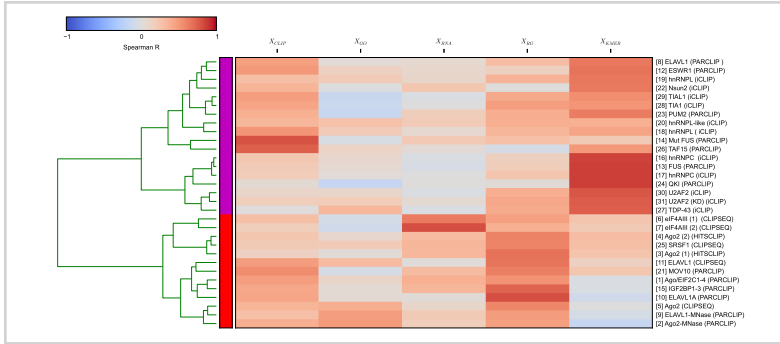


Figure C.1

Hierarchical clustering of proteins based on the importance of each individual data source.

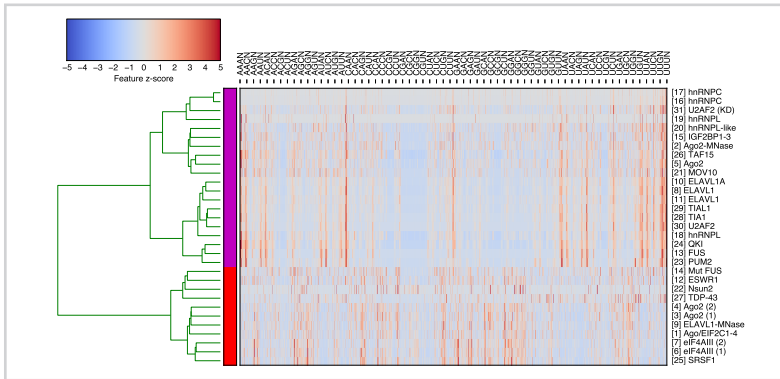


Figure C.2

Protein similarity based on RNA k-mer row vectors in $\mathbf{H}_{\text{KMER}}$. Features represent all possible kmers within the interval [-50..50] relative to the crosslink sites, resulting in $101 \times 256 = 25{,}856$ features. To avoid clutter, only the centers and first three nucleotides of the 4-mer are displayed.

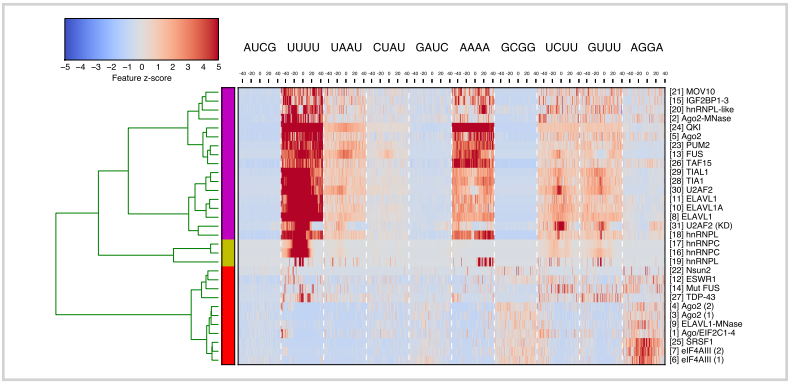


Figure C.3

Protein similarity based on RNA k-mer row vectors in $\mathbf{H}_{\text{KMER}}$. K-means clustering is performed on row vectors from top modules for each of the RBP experiments. Ten k-mers closest to the centroid vectors were selected. The z-scores within the intervals $[-50..50]$ nucleotides are displayed.

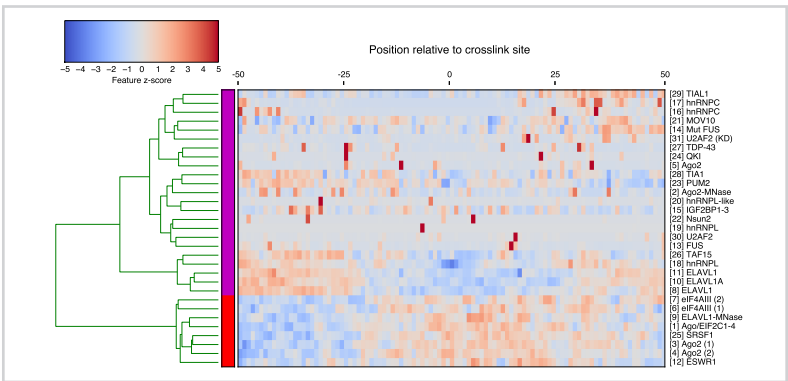


Figure C.4

Protein similarity based on RNA secondary structure row vectors in $\mathbf{H}_{\text{RNA}}$. Z-scores of features obtained via RNAfold output are proportional to the predicted probability of double-stranded RNA at the particular nucleotide within the $[-50..50]$ interval relative to the crosslink sites.

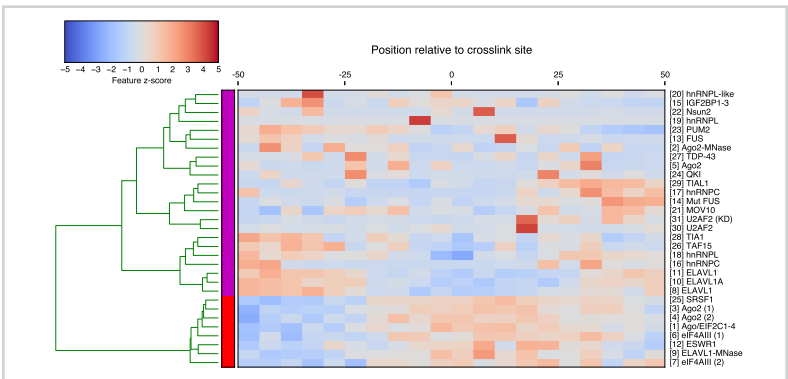


Figure C.5

Protein similarity based on RNA secondary structure row vectors in $\mathbf{H}_{\text{RNA}}$. Z-scores of features obtained via RNAfold output are proportional to the predicted probability of double-stranded RNA at the particular nucleotide within the $[-50..50]$ interval relative to the crosslink sites. Scores within 5 nucleotide bins were summed.

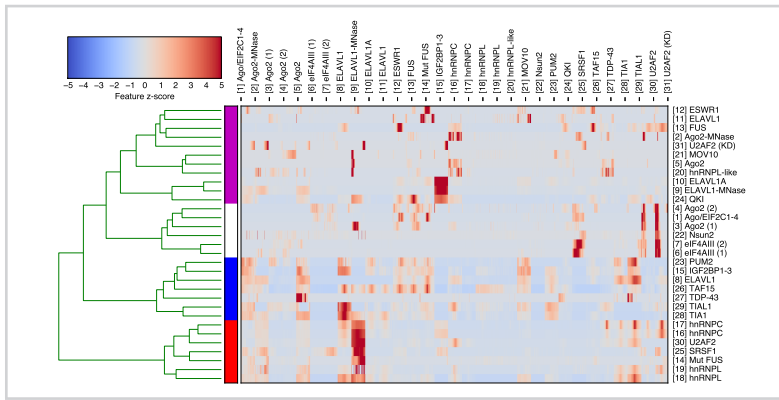


Figure C.6

Protein similarity based on cDNA counts row vectors in $\mathbf{H}_{\text{CLIP}}$. Note that the values for RBPs in the same groups are zero in order not to bias the clustering.

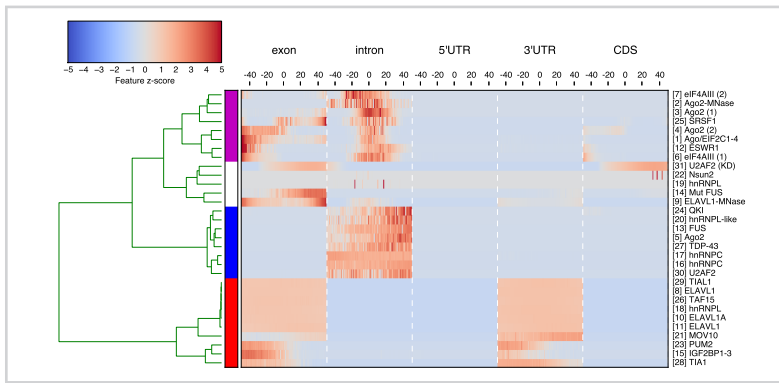


Figure C.7

Protein similarity based on genomic region types row vectors in $\mathbf{H}_{\text{RG}}$. For each region type, the interval [-50..50] relative to the crosslink sites is shown.

Table C.1

Detailed information on individual protein-RNA experimental interaction data used. Total number of sites for each experiment. Experiments describing same protein in different technical or biological replicates are grouped.

ID	Protein	Tissue	Protocol	Ref. group	CL sites	Clusters	Reference
[1]	Ago/EIF2C1-4	HEK293	PARCLIP	A	1364096	41450	[108]
[2]	Ago2-MNase	HEK293	PARCLIP	A	3013785	33396	[109]
[3]	Ago2 (1)	HEK293	HITSCLIP	A	432139	7153	[110]
[4]	Ago2 (2)	HEK293	HITSCLIP	A	432089	7152	[110]
[5]	Ago2	HEK293	CLIPSEQ	A	2766476	27812	[109]
[6]	eIF4AIII (1)	HeLa	CLIPSEQ	B	63353334	5397466	[107]
[7]	eIF4AIII (2)	HeLa	CLIPSEQ	B	20925715	1693124	[107]
[8]	ELAVL1	HEK293	PARCLIP	C	1202570	32129	[109]
[9]	ELAVL1-MNase	HEK293	PARCLIP	C	7940664	84469	[109]
[10]	ELAVL1A	HEK293	PARCLIP	C	256387	5110	[109]
[11]	ELAVL1	HEK293	CLIPSEQ	C	223121	4806	[109]
[12]	ESWR1	HEK293	PARCLIP	D	543116	19019	[111]
[13]	FUS	HEK293	PARCLIP	E	1012411	39983	[111]
[14]	Mut FUS	HEK293	PARCLIP	E	380345	14953	[111]
[15]	IGF2BP1-3	HEK293	PARCLIP	F	6097934	43530	[108]
[16]	hnRNPC	HeLa	iCLIP	G	4602041	438360	[67]
[17]	hnRNPC	HeLa	iCLIP	G	228961	24448	[54]
[18]	hnRNPL	HeLa	iCLIP	H	1125304	-	[112]
[19]	hnRNPL	U266	iCLIP	H	123685	-	[112]
[20]	hnRNPL-like	U266	iCLIP	H	128958	-	[112]
[21]	MOV10	HEK293	PARCLIP	I	592451	17053	[113]
[22]	Nsun2	HEK293	iCLIP	J	75343	-	[114]
[23]	PUM2	HEK293	PARCLIP	K	368700	10962	[108]
[24]	QKI	HEK293	PARCLIP	L	381140	12035	[108]
[25]	SRSF1	HEK293	CLIPSEQ	M	966912	23629	[96]
[26]	TAF15	HEK293	PARCLIP	N	222421	8677	[111]
[27]	TDP-43	HeLa	iCLIP	O	3053718	118703	[115]
[28]	TIA1	HeLa	iCLIP	P	393111	21884	[116]
[29]	TIAL1	HeLa	iCLIP	P	1146658	51751	[116]
[30]	U2AF2	HeLa	iCLIP	Q	3668916	518794	[67]
[31]	U2AF2 (KD)	HeLa	iCLIP	Q	9147292	1122142	[67]

Table C.2

Details on parameter settings for factor models on different subsets of data sources.

Subset name	Data subset s	$\sum_s d_s$	N_s	r_s	Avg. AUC
C	$\mathbf{X}_{\text{CLIP}}$	3030	1219713	23	0.733 ± 0.018
G	$\mathbf{X}_{\text{GO}}$	101	1202448	24	0.493 ± 0.009
K	$\mathbf{X}_{\text{KMER}}$	505	1212144	24	0.690 ± 0.017
R	$\mathbf{X}_{\text{RNA}}$	25856	1213712	16	0.744 ± 0.024
T	$\mathbf{X}_{\text{RG}}$	39560	1253854	14	0.704 ± 0.018
CG	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{GO}}$	3131	1222059	23	0.701 ± 0.021
CK	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{KMER}}$	3535	1231351	23	0.820 ± 0.020
CR	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{RNA}}$	28886	1262208	16	0.788 ± 0.023
CT	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{RG}}$	42590	1203696	13	0.796 ± 0.021
GK	$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{KMER}}$	606	1214592	24	0.776 ± 0.020
GR	$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{RNA}}$	25957	1215344	16	0.699 ± 0.026
GT	$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{RG}}$	39661	1255282	14	0.816 ± 0.015
KR	$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RNA}}$	26361	1221808	16	0.763 ± 0.019
KT	$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RG}}$	40065	1260938	14	0.860 ± 0.018
RT	$\mathbf{X}_{\text{RNA}}$,$\mathbf{X}_{\text{RG}}$	65416	1269598	11	0.735 ± 0.022
CGK	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{KMER}}$	3636	1233697	23	0.842 ± 0.016
CGR	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{RNA}}$	28987	1263840	16	0.774 ± 0.026
CGT	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{RG}}$	42691	1205022	13	0.858 ± 0.018
CKR	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RNA}}$	29391	1190910	15	0.911 ± 0.000
CKT	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RG}}$	43095	1210274	13	0.910 ± 0.000
CRT	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{RNA}}$,$\mathbf{X}_{\text{RG}}$	68446	1302939	11	0.807 ± 0.021
GKR	$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RNA}}$	26462	1223440	16	0.830 ± 0.017
GKT	$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RG}}$	40166	1262366	14	0.884 ± 0.008
GRT	$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{RNA}}$,$\mathbf{X}_{\text{RG}}$	65517	1270720	11	0.834 ± 0.015
KRT	$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RNA}}$,$\mathbf{X}_{\text{RG}}$	65921	1275164	11	0.873 ± 0.016
CGKR	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RNA}}$	29492	1192440	15	0.903 ± 0.008
CGKT	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RG}}$	43196	1211600	13	0.880 ± 0.011
CGRT	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{RNA}}$,$\mathbf{X}_{\text{RG}}$	68547	1304061	11	0.863 ± 0.016
CKRT	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RNA}}$,$\mathbf{X}_{\text{RG}}$	68951	1308505	11	0.921 ± 0.007
GKRT	$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RNA}}$,$\mathbf{X}_{\text{RG}}$	66022	1276286	11	0.878 ± 0.012
CGKRT	$\mathbf{X}_{\text{CLIP}}$,$\mathbf{X}_{\text{GO}}$,$\mathbf{X}_{\text{KMER}}$,$\mathbf{X}_{\text{RNA}}$,$\mathbf{X}_{\text{RG}}$	69052	1190570	10	0.887 ± 0.011

Table C.3

Area under ROC curve for all combinations of RBP experiments and data sources subsets.

Subset	[1] Ago/EIF2C1-4	[2] Ago2-MNase	[3] Ago2 (1)	[4] Ago2 (2)	[5] Ago2	[6] eIF4AIII (1)	[7] eIF4AIII (2)	[8] ELAVL1	[9] ELAVL1-MNase	[10] ELAVL1A
C	0.805	0.716	0.689	0.698	0.711	0.729	0.716	0.783	0.688	0.927
G	0.497	0.505	0.515	0.527	0.505	0.439	0.513	0.519	0.512	0.540
K	0.601	0.518	0.641	0.660	0.523	0.776	0.762	0.682	0.530	0.660
R	0.885	0.705	0.874	0.870	0.691	0.926	0.939	0.749	0.736	0.907
T	0.732	0.632	0.771	0.584	0.634	0.797	0.752	0.651	0.638	0.964
CG	0.751	0.713	0.662	0.697	0.674	0.667	0.675	0.774	0.672	0.932
CK	0.698	0.601	0.761	0.742	0.593	0.863	0.863	0.924	0.594	0.960
CR	0.909	0.745	0.889	0.887	0.720	0.926	0.938	0.810	0.749	0.943
CT	0.907	0.691	0.881	0.879	0.661	0.897	0.920	0.847	0.704	0.980
GK	0.625	0.538	0.718	0.725	0.582	0.853	0.889	0.789	0.585	0.848
KR	0.685	0.543	0.805	0.750	0.551	0.909	0.947	0.725	0.558	0.877
KT	0.886	0.617	0.920	0.925	0.696	0.931	0.935	0.943	0.621	0.967
GR	0.808	0.695	0.829	0.854	0.659	0.877	0.922	0.713	0.683	0.908
GT	0.884	0.778	0.883	0.895	0.785	0.901	0.898	0.807	0.811	0.973
RT	0.856	0.645	0.815	0.717	0.656	0.883	0.949	0.600	0.620	0.968
CGK	0.770	0.710	0.744	0.759	0.718	0.791	0.777	0.934	0.678	0.968
CKR	0.831	0.592	0.816	0.802	0.600	0.946	0.955	0.943	0.590	0.967
CKT	0.913	0.714	*0.928*	0.925	0.752	0.939	0.942	*0.954*	0.723	0.974
CGR	0.915	0.748	0.843	0.880	0.700	0.938	0.944	0.805	0.740	0.963
CGT	0.909	0.824	0.897	0.906	0.816	0.932	0.926	0.861	0.816	*0.984*
CRT	0.860	0.690	0.859	0.908	0.667	0.952	0.962	0.811	0.670	0.980
GKR	0.719	0.628	0.829	0.853	0.596	0.924	0.939	0.888	0.691	0.905
GKT	0.885	0.808	0.903	0.912	0.775	0.900	0.892	0.921	0.819	0.969
KRT	0.890	0.659	0.918	0.929	0.713	0.951	0.959	0.932	0.649	0.970
GRT	0.901	0.810	0.906	0.911	0.779	0.936	0.953	0.840	0.817	0.972
CGKR	0.919	0.771	0.910	0.919	0.752	0.942	0.957	0.951	0.753	0.967
CGKT	0.900	0.846	0.906	0.914	0.821	0.926	0.938	0.933	0.820	0.970
CKRT	*0.927*	0.724	0.928	*0.932*	0.731	*0.958*	*0.964*	0.948	0.777	0.973
CGRT	0.923	*0.851*	0.913	0.915	0.822	0.942	0.956	0.865	0.816	0.983
GKRT	0.906	0.822	0.913	0.905	0.783	0.940	0.951	0.924	0.824	0.963
CGKRT	0.924	0.849	0.917	0.924	*0.825*	0.944	0.956	0.929	*0.828*	0.973

Table C.4

Area under ROC curve for all combinations of RBP experiments and data sources subsets (continued).

Subset	[11] ELAVL1	[12] ESWR1	[13] FUS	[14] Mut FUS	[15] IGF2BP1-3	[16] hnRNPC	[17] hnRNPC	[18] hnRNPL	[19] hnRNPL	[20] hnRNPL-like
C	0.918	0.783	0.633	0.893	0.808	0.603	0.597	0.733	0.656	0.685
G	0.555	0.498	0.499	0.510	0.508	0.399	0.402	0.487	0.519	0.514
K	0.690	0.749	0.761	0.724	0.570	0.768	0.970	0.627	0.614	0.599
R	0.888	0.737	0.573	0.862	0.885	0.509	0.511	0.708	0.666	0.712
T	0.930	0.604	0.627	0.769	0.758	0.734	0.724	0.613	0.600	0.586
CG	0.928	0.760	0.607	0.886	0.782	0.491	0.482	0.706	0.646	0.660
CK	0.956	0.863	0.798	0.930	0.699	0.952	0.973	0.788	0.713	0.728
CR	0.922	0.808	0.624	0.931	0.910	0.580	0.569	0.760	0.693	0.736
CT	0.968	0.649	0.664	0.941	0.928	0.755	0.721	0.669	0.622	0.612
GK	0.883	0.795	0.807	0.804	0.613	0.952	0.974	0.697	0.697	0.669
KR	0.762	0.763	0.773	0.766	0.698	0.949	0.971	0.658	0.670	0.678
KT	0.965	0.831	0.782	0.910	0.875	0.945	0.970	0.761	0.767	0.679
GR	0.864	0.662	0.554	0.823	0.852	0.424	0.430	0.668	0.627	0.665
GT	0.975	0.770	0.638	0.868	0.904	0.735	0.724	0.768	0.704	0.767
RT	0.952	0.574	0.628	0.796	0.911	0.737	0.726	0.612	0.604	0.587
CGK	0.974	0.872	0.812	0.938	0.791	0.956	0.975	0.770	0.748	0.702
CKR	0.963	0.866	0.811	*0.956*	0.821	0.955	0.973	0.801	0.784	0.747
CKT	0.971	0.893	0.811	0.955	0.936	0.942	0.976	*0.826*	*0.801*	0.796
CGR	0.960	0.816	0.609	0.937	0.920	0.484	0.477	0.753	0.698	0.722
CGT	*0.983*	0.834	0.702	0.933	0.926	0.780	0.770	0.794	0.754	0.786
CRT	0.968	0.765	0.675	0.950	0.939	0.754	0.729	0.689	0.621	0.590
GKR	0.947	0.809	0.813	0.825	0.705	0.952	0.974	0.732	0.734	0.724
GKT	0.967	0.833	0.821	0.890	0.918	0.955	0.973	0.795	0.785	0.776
KRT	0.965	0.834	0.782	0.929	0.922	0.932	0.975	0.770	0.783	0.769
GRT	0.969	0.785	0.637	0.900	0.921	0.738	0.734	0.783	0.744	0.779
CGKR	0.971	0.886	*0.832*	0.955	0.916	0.955	0.975	0.794	0.764	0.769
CGKT	0.978	0.846	0.817	0.933	0.931	*0.958*	*0.978*	0.810	0.786	0.791
CKRT	0.969	*0.894*	0.829	0.954	*0.944*	0.952	0.975	0.822	0.790	*0.801*
CGRT	0.979	0.830	0.715	0.945	0.934	0.779	0.755	0.799	0.742	0.778
GKRT	0.968	0.836	0.796	0.906	0.930	0.926	0.970	0.769	0.761	0.776
CGKRT	0.980	0.865	0.776	0.950	0.940	0.953	0.976	0.795	0.785	0.788

Table C.5

Area under ROC curve for all combinations of RBP experiments and data sources subsets (continued).

Subset	[21] MOV10	[22] Nsun2	[23] PUM2	[24] QKI	[25] SRSF1	[26] TAF15	[27] TDP-43	[28] TIA1	[29] TIAL1	[30] U2AF2	[31] U2AF2 (KD)
C	0.879	0.676	0.812	0.585	0.700	0.843	0.512	0.839	0.811	0.632	0.650
G	0.496	0.531	0.511	0.521	0.515	0.516	0.529	0.478	0.334	0.372	0.505
K	0.652	0.715	0.746	0.845	0.718	0.768	0.696	0.682	0.706	0.754	0.695
R	0.882	0.761	0.836	0.536	0.874	0.728	0.480	0.780	0.713	0.554	0.593
T	0.792	0.554	0.758	0.658	0.679	0.584	0.729	0.870	0.734	0.695	0.673
CG	0.861	0.668	0.795	0.582	0.661	0.842	0.540	0.798	0.699	0.504	0.632
CK	0.907	0.823	0.936	0.901	0.787	0.897	0.733	0.928	0.907	0.818	0.795
CR	0.924	0.798	0.872	0.572	0.878	0.838	0.492	0.858	0.821	0.647	0.679
CT	0.933	0.709	0.902	0.677	0.852	0.767	0.739	0.905	0.892	0.720	0.692
GK	0.684	0.779	0.834	0.912	0.761	0.835	0.868	0.821	0.820	0.878	0.837
KR	0.731	0.720	0.879	0.902	0.745	0.790	0.758	0.769	0.725	0.819	0.786
KT	0.926	0.834	0.943	0.866	0.888	0.816	0.838	0.935	0.912	0.909	0.876
GR	0.841	0.716	0.752	0.546	0.812	0.690	0.521	0.709	0.557	0.423	0.575
GT	0.896	0.702	0.869	0.662	0.873	0.754	0.759	0.875	0.849	0.796	0.796
RT	0.908	0.556	0.842	0.657	0.708	0.646	0.729	0.787	0.693	0.713	0.696
CGK	0.896	0.846	0.931	0.847	0.800	0.905	0.859	0.918	0.911	0.909	0.892
CKR	0.944	0.840	0.945	*0.912*	0.806	0.909	0.808	0.937	0.920	0.851	0.821
CKT	0.956	0.868	*0.961*	0.877	0.898	*0.910*	0.860	0.943	*0.942*	0.915	*0.925*
CGR	0.932	0.748	0.875	0.574	0.878	0.849	0.523	0.841	0.756	0.528	0.651
CGT	0.933	0.779	0.897	0.671	0.881	0.879	0.753	0.907	0.897	0.832	0.821
CRT	0.950	0.697	0.906	0.683	0.849	0.824	0.738	0.914	0.896	0.780	0.737
GKR	0.854	0.811	0.893	0.889	0.824	0.845	0.869	0.874	0.880	0.908	0.884
GKT	0.932	0.744	0.928	0.831	0.884	0.826	*0.895*	0.919	0.889	0.926	0.901
KRT	0.947	0.840	0.952	0.871	0.884	0.827	0.845	0.941	0.927	0.906	0.879
GRT	0.926	0.782	0.881	0.665	0.896	0.795	0.734	0.886	0.866	0.820	0.804
CGKR	0.941	*0.872*	0.933	0.846	0.880	0.895	0.872	0.933	0.903	0.908	0.902
CGKT	0.950	0.794	0.924	0.797	0.891	0.880	0.891	0.918	0.917	0.908	0.899
CKRT	*0.959*	0.869	0.956	0.868	*0.913*	0.900	0.858	*0.949*	0.931	*0.933*	0.904
CGRT	0.949	0.794	0.902	0.686	0.898	0.875	0.742	0.905	0.893	0.830	0.820
GKRT	0.941	0.760	0.932	0.834	0.898	0.836	0.893	0.903	0.879	0.872	0.895
CGKRT	0.955	0.793	0.930	0.842	0.909	0.896	0.893	0.911	0.885	0.903	0.901

Razširjeni povzetek

D

V času pospešenega zbiranja, organiziranja in dostopnosti podatkov se pojavlja potreba po razvoju napovednih modelov na osnovi hkratnega učenja iz več podatkovnih virov. Konkretni primeri uporabe obsegajo področja strojnega učenja, priporočilnih sistemov, socialnih omrežij, financ in računske biologije. Heterogenost in velikost tipičnih podatkovnih zbirk vodi razvoj postopkov za hkratno zmanjšanje velikosti (zgoščevanje) in sklepanje iz več virov podatkov v skupnem model. Matrična faktorizacija in jedrne metode (ang. *kernel methods*) sta dve splošni orodji, ki omogočata dosego navedenega cilja. Pričujoče delo se osredotoča na naslednja specifična cilja: i) iskanje interpretabilnih, neprekrivajočih predstavitev vzorcev v podatkih s pomočjo ortogonalne matrične faktorizacije in ii) nadzorovano hkratno faktorizacijo več jedrnih matrik, ki omogoča modeliranje nelinearnih odzivov in interpretacijo pomembnosti različnih podatkovnih virov.

Motivacija za razvoj modelov in algoritmov v pričujočem delu izhaja iz RNA biologije in bogate kompleksnosti interakcij med proteini in RNA molekulami v celici. Čeprav se regulacija RNA dogaja na več različnih nivojih - kar vodi v več podatkovnih virov/pogledov - lahko odgovorimo na vprašanja s pomočjo omejitev v fazi modeliranja. V delu predstavimo postopek hkratne matrične faktorizacije z omejitvijo, da se posamezni vzorci v podatkih ne prekrivajo med seboj - so neodvisni oz. ortogonalni. V praksi to pomeni, da lahko odkrijemo različne, neprekrivajoče načine regulacije RNA s strani različnih proteinov. Z vzključitvijo več podatkovnih virov izboljšamo napovedno točnost pri napovedovanju potencialnih vezavnih mest posameznega RNA-vezavnega proteina. Vzorci, odkriti iz podatkov so primerljivi z eksperimentalno določenimi lastnostmi posameznega RNA vezavnega proteina, ki obsegajo kratka zaporedja nukleotidov na RNA, kooperativno vezavo z drugimi proteini, RNA 3D strukturnimi lastnostmi ter funkcijsko anotacijo.

Klasične metode matrične faktorizacije tipično temeljijo na linearnih modelih podatkov oz. tarčnih izhodnih funkcij. Jedrne metode so eden od načinov za razširitev modelov matrične faktorizacije za modeliranje nelinearnih odzivov. Učenje z več jedri (ang. *Multiple kernel learning*) omogoča učenje iz več podatkovnih virov, a je omejeno s kvadratno računsko zahtevnostjo v odvisnosti od števila primerov (instanc) v podatkih. To omejitev lahko omilimo z ustreznimi približki pri izračunu jedrnih matrik (ang. *kernel matrix*). V ta namen izboljšamo obstoječe metode na način, da hkrati izračunamo aproksimacijo jedrnih matrik ter njihovo linearno kombinacijo, ki modelira podan tarčni signal. To dosežemo z metodo Mklaren (ang. *Multiple kernel learn-*

ing based on Least-angle regression), ki je sestavljena iz Nepopolnega razcepa Choleskega in Regresije najmanjših kotov (ang. *Least-angle regression*). Načrt algoritma vodi v linearno časovno in prostorsko odvisnost tako glede na število primerov v podatkih kot tudi glede na število jeder. Osnovne prednosti postopka so poleg računske odvisnosti tudi splošnost oz. neodvisnost od uporabljenih jedrnih funkcij. Tako lahko uporabimo različne, splošne jedrne funkcije za modeliranje različnih delov prostora vhodnih podatkov, ki so lahko zvezni ali diskretni, npr. vektorski prostori, prostori nizov in drugih podatkovnih struktur, kar je prikladno za uporabo v bioinformatiki.

V delu tako razvijemo algoritme na osnovi hkratne matrične faktorizacije in jedrnih metod, obravnavnamo modele linearne in nelinearne regresije ter interpretacije podatkovne domene - odkrijemo pomembne jedrne funkcije in primere podatkov, pri čemer je metode mogoče poganjati na milijonih podatkovnih primerov in podatkovnih virov.

Ortogonalna matrična faktorizacija z več podatkovnimi viri

Matrična faktorizacija je matematična metoda, ki rešuje problem razcepa matrike in njenega zapisa v obliki produkta vsaj dveh matrik. Je orodje za reševanje sistemov linearnih enačb, v strojnem učenju pa pogosto služi kot približen zapis matrike podatkov in omogoča stiskanje podatkov, iskanje vzorcev, odstranjevanje šuma, napovedovanje neznanih ali manjkajočih vrednosti in več.

Naj bo $\mathbf{X} \in \mathbb{R}^{n \times d}$ matrika $n \times d$ realnih števil oz. podatkov. Cilj je poiskati matriki $\mathbf{A} \in \mathbb{R}^{n \times r}$ ter $\mathbf{B} \in \mathbb{R}^{d \times r}$, kjer je $r \leq \min(n, d)$, tako da:

$$\mathbf{X} \approx \mathbf{A}\mathbf{B}^T. \tag{D.1}$$

Pogosto je število r rang matrik $\mathbf{A}$ in $\mathbf{B}$, ter bistveno manjši od originalnih dimenzij n in d, kar pomeni, da je rešitev približek natančne rešitve. Problem je rešljiv s pomočjo matematične optimizacije, algoritem pa je odvisen od omejitev, ki jih predpostavimo za *model*, podan z $\mathbf{A}$ in $\mathbf{B}$.

V primeru strojnega učenja tako iščemo kompromis med natančnostjo približka ter kompaktnostjo takega zapisa. Izkaže se, da so morebitni vzorci prisotni v $\mathbf{X}$ - podobni stolpci oz. vrstice matrike - povzeti v $\mathbf{A}$ in $\mathbf{B}$. Zaradi velikega prostora možnih modelov (veliko možnih lokalnih minimumov Neenačbe D.1) pa je iskanje vzorcev lahko oteženo, posebej ko so vrednosti $\mathbf{X}$ različnih velikostnih redov, vzorci pa prisotni v

neenakomernih razmerjih. Tako pri optimizaciji parametrov $\mathbf{A}$ in $\mathbf{B}$ uporabimo dodatne predpostavke, kot so npr. nenegativnost (nenegativna matrična faktorizacija, NMF), redkost, omejenost ali ortogonalnost.

V pričujočem delu razvijemo algoritem za reševanje problem nenegativne, ortogonalne matrične faktorizacije (ang. *Integrative, orthogonal non-negative matrix factorization*, iONMF), kjer so vhodni podatki predstavljeni z več podatkovnimi matrikami $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_p$; te lahko predstavljajo več podatkovnih virov oz. pogledov na podatke. V modelu poiščemo skupno matriko $\mathbf{A}$ ter matrike $\mathbf{B}_1, \mathbf{B}_2, ..., \mathbf{B}_p$, ki so lastne vsakemu od podatkovnih virov:

$$\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_p \approx \mathbf{A}\mathbf{B}_1^T, \mathbf{A}\mathbf{B}_2^T, ..., \mathbf{A}\mathbf{B}_p^T,$$

z omejitvami

- nenegativnosti; $\mathbf{X} \geq \mathbf{0}$, $\mathbf{A} \geq \mathbf{0}$, $\mathbf{B} \geq \mathbf{0}$, ter
- približne ortogonalnosti $\|\mathbf{B}_q^T\mathbf{B}_q - \mathbf{I}\| < t$ za skalar $0 < t < \infty$ in $q = 1, 2, ...p$.

Omejitvi sta pomembni zaradi izboljšane interpretacije vzorcev (nenegativnost, ortogonalnost) ter redkosti in iskanja neprekrivajočih vzorcev, prisotnih v različnih številih primerov (ortogonalnost). Poseben poudarek je na ortogonalnosti matrik, specifičnih za posamezne podatkovne vire $\mathbf{B}$, saj slednja omejitev naredi problem različen od preprostega združevanja (konkatenacije) podatkov v eno samo matriko.

Omenjene izboljšave postopka so navidez zelo tehnične narave, vendar pomembno izboljšajo modeliranje interakcij med proteini in RNA. V konkretnem primeru vrstice matrik predstavljajo različne genomske lokacije, stolpci pa predstavitev različnih podatkovnih virov oz. dejavnikov: afinitete vezave (protokol CLIP), RNA zaporedja (sekvence), RNA 3D strukture, anotacije genskih regij ter funkcijske anotacije. Različni proteini prepoznavanjo oz. nastopajo v reakcijah z različnimi RNA molekulami. Proces interakcije je odvisen od različnih dejavnikov; proteini prepoznavajo specifična, kratka zaporedja nukleotidov na RNA (A, C, G in U) - RNA motive. Da pa bo do dejanske vezave v resnici prišlo, je odvisno tudi od drugih dejavnikov, kot so npr. dostopnost RNA motiva v prostoru (3D struktura RNA), lokacija RNA v celici, zadostne koncentracije obravnavanega proteina, kompeticije za vezavna mesta z drugimi proteini. Našteta dejstva govorijo v prid modeliranja/napovedovanja interakcij s hkratno uporabo več podatkovnih virov oz. pogledov na podatke. Vzorci, ki jih odkrijemo s

pomočjo modelov NMF, so prisotni v več podatkovnih virih in so v podatkih različno pristotni - odkrivanje vzorcev izboljšamo z uporabo ortogonalnih modelov NMF.

Metodo primerjamo s sorodnimi pristopi matrične faktorizacije na podlagi več kriterijev: napovedne točnosti, prekrivanja med odkritimi vzorci, redkosti modelov in ujemanjem z obstoječim znanjem o posameznih RNA vezavnih proteinih. Pri tem izboljšamo napovedno točnost pri 24 od 31 obravnavanih RNA vezavnih proteinov (Slika 4.1, str. 35). Razlika med iONMF in običajnim pristopom NMF se povečuje s številom učnih primerov genomskih lokacij (Slika 4.2, str. 34). Z nadzorovanjem ortogonalnosti faktorjev modela najdemo vzorce z bistveno manjšo stopnjo prekrivanja, večjo stopnjo redkosti, za bistveno manjšo ceno v napovedni točnosti (Slika 4.3, str. 36). V primerjavi s klasično metodo NMF, predlagana metoda iONMF odkrije neprekrivajoče vzorce in različne načine regulacije RNA s strani istega proteina (Slika 4.9, str. 43). V pripadajoči objavi odkrijemo tipične lastnosti molekul RNA, ki nastopajo v reakcijah z vsakim od 31 obravnavanih RNA-vezavnih proteinov [66].

Na ta način pokažemo praktično uporabo metode za napovedovanje interakcij med proteini in RNA ter iskanje vzorcev v podatkih. V primeru velikih prostorov modelov - v tem primeru podanim z nekaj deset tisoč dimenzionalnimi vektorskimi prostori - je pomembna uporaba dodatnih omejitev pri optimizaciji parametrov. Metoda iONMF tako izboljša obstoječe metode matrične faktorizacije in je uporabna v primeru iskanja neprekrivajočih vzorcev v podatkih.

Nadzorovana aproksimacija jedrnih matrik

Zgoraj opisane metode matrične faktorizacije lahko obravnavamo kot modele linearne regresije z enim ali več tarčnimi funkcijami. Za modeliranje nelinearnih izhodnih funkcij lahko uporabimo jedrne metode (ang. *kernel methods*). V tem primeru kot podatkovni viri nastopajo različne jedrne funkcije, ki jih lahko interpretiramo kot funkcije podobnosti med podatki - določajo kovariančno strukturo skupne porazdelitve. Jedrna funkcija med elementoma vektorskega prostora $\mathbf{x}_i$ in $\mathbf{x}_j$, $k(\mathbf{x}_i, \mathbf{x}_j)$ predstavlja notranji produkt in je lahko definirana na prostorih realnih števil, nizov znakov, podatkovnih struktur in drugih objektov. Tako lahko modeliramo podatke z bogatimi predstavitvami, kar je v bioinformatiki uporabno npr. pri modeliranju bioloških zaporedij.

Cena pri uporabi jedrnih metod je kvadratna računska zahtevnost pri izračunu in shranjevanju jedrne matrike. Tudi v tem primeru se pokaže potreba po hkratnem

zmanjševanju dimenzije podatkov in iskanju parametrov modela. Predlagamo novo metodo *Mklaren*, ki hkrati rešuje oba omenjena problema, tako da pri aproksimaciji več jedrnih matrik poišče relevantne dele, ki se prilegajo tarčni funkciji. Na ta način lahko odkrijemo relevantne dele prostora vhodnih podatkov in izberemo najprimernejše jedrne funkcije za obravnavani problem (Slika 6.1, str. 68).

Za zmanjšanje dimenzionalnosti prostora prilagodimo metodo Nepopolnega razcepa Choleskega (ang. *Incomplete Cholesky Decomposition*). Za dano simetrično, pozitivno-semidefinitno jedrno matriko $\mathbf{K}$, kjer je $\mathbf{K}(i,j) = k(\mathbf{x}_i, \mathbf{x}_j)$, je Razcep Choleskega definiran kot

$$\mathbf{K} = \mathbf{G}\mathbf{G}^T, \tag{D.2}$$

kjer je $\mathbf{G}$ zgornje-trikotna matrika. Enačba D.2 drži v primeru, ko je rang matrike $\mathbf{K}$ enak rangu matrike $\mathbf{G}$. Ker je tak rang v primerih velikih podatkov prevelik za praktično uporabo, metodo razcepa ustavimo pri manjšem rangu od polnega in tako dobimo Nepopoln razcep Choleskega. V pričujočem delu predlagamo tehnični rešitvi za naslednja problema:

- obravnava razcepa v kontekstu nadzorovanega učenja, ter
- hkratni razcep več jedrnih matrik.

Problema se prevedeta na izbiro pivotov (posameznih delov jedrne matrike) oz. *baznih funkcij*. Če delujemo v kontekstu nadzorovanega učenja, npr. regresije, lahko pivot izberemo na način, ki najbolj ustreza modeliranju tarčnega signala. V ta namen uporabimo tehniko regresije najmanjših kotov (LAR, ang. *Least-angle regression*, Slika 6.2, str. 71) in poudarimo povezavo med metodama LAR in Nepopolnim razcepom Choleskega.

Rezultat je splošna metoda za nadzorovano učenje in/ali izbiro najustreznejših jedrnih funkcij. Metodo smo ovrednotili na splošnih podatkovnih zbirkah, zgrajenih za namen vrednotenja regresijskih modelov. Predlagana metoda *Mklaren* izboljša trenutne metode za aproksimacijo splošnih jedrnih matrik ter je primerljiva z bolj specifičnimi metodami, ki delujejo na podrazredih jedrnih funkcij, pri čemer vrednotimo:

- minimalni rang aproksimacije za dosego primerljive napovedne točnosti v primerjavi z uporabo celotne jerdne matrike,
- napovedno točnost v diskretnih in zveznih prostorih vhodnih podatkov,

- iskanje smiselnih jeder v obdelavi besedil in bioloških zaporedjih,
- časovno zahtevnost.

Metodo prav tako uporabimo v praksi za modeliranje interakcij med proteini in RNA, pri čemer se osredotočimo na modeliranje bioloških zaporedij, predstavljenimi kot nizi znakov končne abecede. Poleg izboljšanje napovedne točnosti uspemo odkriti najprimerjenše dolžine RNA motivov, ki jih prepoznava posamezni RNA vezavni protein. Metoda tako omogoča modeliranje linearnih in nelinearnih izhodnih funkcij, ki so podane s kovariačno strukturo (jedrno funkcijo) v odvisnosti od podatkov. Z uporabo aproksimacije jedrnih matrik tako omogočimo modeliranje podatkovnih zbirk z milijoni vhodnih primerov in zapis modelov z več tisoč jedrnimi funkcijami.

BIBLIOGRAPHY

[1] Tinghui Zhou, H Shan, A Banerjee, and G Sapiro. Kernelized Probabilistic Matrix Factorization: Exploiting Graphs and Side Information. *SDM*, 2012.

[2] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940. ACM, 2008. ISBN 9781595939913.

[3] Stein Aerts, Diether Lambrechts, Sunit Maity, Peter Van Loo, Bert Coessens, Frederik De Smet, Leon-Charles Tranchevent, Bart De Moor, Peter Marynen, Bassem Hassan, Peter Carmeliet, and Yves Moreau. Gene prioritization through genomic data fusion. *Nature biotechnology*, 24(5):537–44, may 2006. ISSN 1087-0156. doi: 10.1038/nbt1203.

[4] Genevieve D Erwin, Nir Oksenberg, Rebecca M Truty, Dennis Kostka, Karl K Murphy, Nadav Ahituv, Katherine S Pollard, and John a Capra. Integrating diverse datasets improves developmental enhancer prediction. *PLoS computational biology*, 10(6):e1003677, jun 2014. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1003677.

[5] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[6] RP Adams, GE Dahl, and Iain Murray. Incorporating Side Information in Probabilistic Matrix Factorization with Gaussian Processes. *arXiv preprint arXiv:1003.4944*, (1999), 2010.

[7] Marinka Zitnik and Blaz Zupan. Data Fusion by Matrix Factorization. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2014.

[8] Weiyong Li. Supervised principal component analysis, 2013.

[9] Gert Lanckriet and Nello Cristianini. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[10] J. M. Hillis. Combining Sensory Information: Mandatory Fusion Within, but Not Between, Senses. *Science*, 298(5598):1627–1630, 2002. ISSN 00368075. doi: 10.1126/science.1075396.

[11] Henrik Boström, Sten F Andler, Marcus Brohede, Ronnie Johansson, Alexander Karlsson, Joeri Van Laere, Lars Niklasson, Maria Nilsson, Anne Persson, and Tom Ziemke. On the Definition of Information Fusion as a Field of Research. *IKI Technical Reports*, pages 1–8, 2007. doi: HS-IKI-TR-07-006.

[12] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Stafford Noble. Learning gene functional classifications from multiple data types. *Journal of computational biology*, 9(2):401–411, 2002.

[13] D D Lee and H S Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–91, oct 1999. ISSN 0028-0836. doi: 10.1038/44565.

[14] Shihua Zhang, Chun-Chi Liu, Wenyuan Li, Hui Shen, Peter W Laird, and Xianghong Jasmine Zhou. Discovery of multi-dimensional modules by integrative analysis of cancer genomic data. *Nucleic acids research*, 40(19):9379–91, oct 2012. ISSN 1362-4962. doi: 10.1093/nar/gks725.

[15] Fei Wang, Tao Li, and Changshui Zhang. Semi-supervised clustering via matrix factorization. In *SDM*, number 60675009, pages 1–12. SIAM, 2008.

[16] Nicolas Gillis and Robert Luce. Robust Near-Separable Nonnegative Matrix Factorization Using Linear Optimization. *arXiv preprint arXiv:1302.4385*, 15:1249–1280, 2014.

[17] Nicolas Gillis. The Why and How of Nonnegative Matrix Factorization. *Regularization, Optimization, Kernels, and Support Vector Machines*, 12:257, jan 2014.

[18] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix

factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, 2013.

[19] Samuel Kaski and Mehmet Gonen. Kernelized Bayesian matrix factorization. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 36(10): 2047–2060, 2014.

[20] C. Boutsidis and E. Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4): 1350–1362, apr 2008. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2007.09.010.

[21] Russell Albright, James Cox, David Duling, A Langville, and C Meyer. Algorithms, initializations, and convergence for the nonnegative matrix factorization. *arXiv preprint arXiv:1407.7299*, (919), 2006.

[22] Nathan Srebro. Maximum-margin matrix factorization. *Advances in neural information processing systems*, pages 1329—-1336, 2004.

[23] S.Z. Li. Local non-negative matrix factorization as a visual representation. *Proceedings 2nd International Conference on Development and Learning. ICDL 2002*, pages 178–183, 2002. doi: 10.1109/DEVLRN.2002.1011835.

[24] Stan Z Li, Xinwen Hou, Hongjiang Zhang, and Qiansheng Cheng. Learning spatially localized, parts-based representation. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I—-207. IEEE, 2001. ISBN 0769512720.

[25] PO O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, nov 2004. ISSN 1532-4435.

[26] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. Online Nonnegative Matrix Factorization With Robust Stochastic Approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1087–1099, jul 2012. ISSN 2162-237X. doi: 10.1109/TNNLS.2012.2197827.

[27] Chris Ding, Tao Li, and Michael I Jordan. Convex and Semi-Nonnegative Matrix Factorizations. *IEEE Transactions on Pattern analysis and Machine Intelligence*, 32(1):45–55, jan 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.277.

[28] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, 2002.

[29] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006. ISBN 9780387310732.

[30] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. *Twenty-first international conference on Machine learning - ICML '04*, page 6, 2004. doi: 10.1145/1015330.1015424.

[31] Mehmet Gönen and Ethem Alpaydin. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, 2011.

[32] Shai Fine and Katya Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

[33] Ali Rahimi and Ben Recht. Weighted sums of random kitchen sinks: replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 1, pages 1177–1184, 2009.

[34] Francis R. Bach and Michael I. Jordan. Predictive low-rank decomposition for kernel methods. In *International Conference on Machine learning (ICML)*, pages 33–40, New York, New York, USA, 2005. ACM Press. ISBN 1595931805.

[35] Brian Kulis, Mátyás Sustik, and Inderjit S. Dhillon. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research*, 10: 341–376, 2009.

[36] Bradley Efron and Trevor Hastie. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

[37] Alain Rakotomamonjy. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008. ISSN 15324435. doi: 10.1.1.139.1778.

[38] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Two-stage learning kernel algorithms. In *International Conference on Machine learning (ICML)*, pages 239–246, 2010.

[39] Francis Bach. Sharp analysis of low-rank kernel matrix approximations. *arXiv preprint arXiv:1208.2015*, aug 2012.

[40] Steven K. Tjoa and K. J.Ray Liu. Multiplicative update rules for nonnegative matrix factorization with co-occurrence constraints. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 449–452, 2010. ISSN 15206149. doi: 10.1109/ICASSP.2010.5495734.

[41] Vincent Y F Tan and Cédric Févotte. Automatic relevance determination in nonnegative matrix factorization with the β-divergence. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1592–605, jul 2013. ISSN 1939-3539. doi: 10.1109/TPAMI.2012.240.

[42] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[43] Gene H Golub and Charles F Van Loan. *Matrix Computations*, volume 3. JHU Press, 2012.

[44] Megasthenis Asteris and Alexandros G Dimakis. Orthogonal NMF through Subspace Exploration. *Advances in Neural Information Processing Systems (NIPS)*, (1):1–20, 2015. ISSN 10495258.

[45] Chris Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. *Proceedings of SDM'05*, pages 606–610, 2005.

[46] DD Daniel D Lee, HS Seung, Bell Laboratories, Murray Hill, and H Sebastian Seung Ý. Algorithms for Non-negative Matrix Factorization. In *NIPS*, number 1, pages 548–562, 2001.

[47] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–79, oct 2007. ISSN 0899-7667. doi: 10.1162/neco.2007.19.10.2756.

[48] Rafal Zdunek and Andrzej Cichocki. Non-negative matrix factorization with quasi-newton optimization. *Artificial Intelligence and Soft Computing*, pages 870–879, 2006.

[49] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[50] Jiho Yoo and Seungjin Choi. Weighted nonnegative matrix co-tri-factorization for collaborative prediction. *Advances in Machine Learning*, pages 396–411, 2009.

[51] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, page 315, 2011. doi: 10.1145/2009916.2009961.

[52] Swapna Joshi, S. Karthikeyan, B. S. Manjunath, Scott Grafton, and Kent a. Kiehl. Anatomical parts-based regression using non-negative matrix factorization. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2863–2870, jun 2010. doi: 10.1109/CVPR.2010.5540022.

[53] Jean-Philippe Brunet, Pablo Tamayo, Todd R Golub, and Jill P Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences of the United States of America*, 101(12):4164–9, mar 2004. ISSN 0027-8424. doi: 10.1073/pnas.0308531101.

[54] Julian König, Kathi Zarnack, Gregor Rot, Tomaz Curk, Melis Kayikci, Blaz Zupan, Daniel J Turner, Nicholas M Luscombe, and Jernej Ule. iCLIP reveals the function of hnRNP particles in splicing at individual nucleotide resolution. *Nature structural & molecular biology*, 17(7):909–15, jul 2010. ISSN 1545-9985. doi: 10.1038/nsmb.1838.

[55] T Hubbard, Daniel Barker, Ewan Birney, Graham Cameron, Yuan Chen, L Clark, Tony Cox, J Cuff, Val Curwen, Thomas Down, and Others. The Ensembl genome database project. *Nucleic acids research*, 30(1): 38–41, 2002.

[56] R B Denman. Using RNAFOLD to predict the activity of small catalytic RNAs. *Biotechniques*, 15(6): 1090–1095, 1993.

[57] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, and Others. Gene Ontology: tool for the unification of biology. *Nature genetics*, 25(1): 25–29, 2000.

[58] Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics (Oxford, England)*, 23(12):1495–502, jun 2007. ISSN 1367-4811. doi: 10.1093/bioinformatics/btm134.

[59] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[60] Samprit Chatterjee and Ali S Hadi. *Regression analysis by example*. John Wiley & Sons, 2015.

[61] Hilal Kazan, Debashish Ray, Esther T Chan, Timothy R Hughes, and Quaid Morris. RNAcontext: a new method for learning the sequence and structure binding preferences of RNA-binding proteins. *PLoS computational biology*, 6(7):e1000832, jan 2010. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000832.

[62] Xiao Li, Gerald Quon, HD Lipshitz, and Quaid Morris. Predicting in vivo binding sites of RNA-binding proteins using mRNA secondary structure. *RNA*, pages 1096–1107, 2010. doi: 10.1261/rna.2017210.sequence.

[63] Yoichiro Sugimoto, Julian König, Shobbir Hussain, Blaž Zupan, Tomaž Curk, Michaela Frye, and Jernej Ule. Analysis of CLIP and iCLIP methods for nucleotide-resolution studies of protein-RNA interactions. *Genome Biol*, 13(8):R67, 2012.

[64] Serena L Chan, Ina Huppertz, Chengguo Yao, Lingjie Weng, James J Moresco, John R Yates, Jernej Ule, James L Manley, and Yongsheng Shi. Cpsf30 and wdr33 directly bind to aauaaa in mammalian mrna 3 processing. *Genes & development*, 28(21):2370–2380, 2014.

[65] Marvin Jens and Nikolaus Rajewsky. Competition between target sites of regulators shapes post-transcriptional gene regulation. *Nature Reviews Genetics*, 16(2):nrg3853, 2014.

[66] Martin Stražar, Marinka Žitnik, Blaž Zupan, Jernej Ule, and Tomaž Curk. Orthogonal matrix factorization enables integrative analysis of multiple RNA binding proteins. *Bioinformatics*, page btw003, 2016.

[67] Kathi Zarnack, Julian König, Mojca Tajnik, Inigo Iñigo Martincorena, Sebastian Eustermann, Isabelle Stévant, Alejandro Reyes, Simon Anders, Nicholas M Luscombe, Jernej Ule, Julian Koenig, and Isabelle Stevant. Direct competition between hnrnp c and u2af65 protects the transcriptome from the exonization of< i> alu</i> elements. *Cell*, 152(3):453–466, jan 2013. ISSN 0092-8674. doi: 10.1016/j.cell.2012.12.023.

[68] Alfredo Castello, Bernd Fischer, Katrin Eichelbaum, Rastislav Horos, Benedikt M Beckmann, Claudia Strein, Norman E Davey, David T Humphreys, Thomas Preiss, Lars M Steinmetz, Jeroen Krijgsveld, and Matthias W Hentze. Insights into RNA biology from an atlas of mammalian mRNA-binding proteins. *Cell*, 149(6):1393–406, jun 2012. ISSN 1097-4172. doi: 10.1016/j.cell.2012.04.031.

[69] Jernej Murn, Kathi Zarnack, Yawei J Yang, Omer Durak, Elisabeth A Murphy, Sihem Cheloufi, Dilenny M Gonzalez, Marianna Teplova, Tomaž Curk, Johannes Zuber, Dinshaw J Patel, Jernej Ule, Nicholas M Luscombe, Li-Huei Tsai, Christopher A Walsh, and Yang Shi. Control of a neuronal morphology program by an RNA-binding zinc finger protein Unkempt. *Genes & Development*, 29:501–512, 2015. ISSN 1549-5477. doi: 10.1101/gad.258483.115.

[70] Erwin Kreyszig. *Introductory functional analysis with applications*, volume 1. Wiley, New York, 1978.

[71] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[72] Carl Edward Rasmussen. Gaussian processes for machine learning. MIT Press, 2006.

[73] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Courier Corporation, 1964.

[74] Sören Sonnenburg, Gunnar Rätsch, and Bernhard Schölkopf. Large scale genomic sequence SVM classifiers. In *International Conference on Machine learning (ICML)*, pages 848–855. ACM, 2005.

[75] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13:795–828, mar 2012.

[76] Corinna Cortes. L-2 Regularization for Learning Kernels. *UAI*, 2009.

[77] John Von Neumann. Uber ein {ö}konomisches Gleichungssystem. In *Ergebn. Math. Kolloq. Wein*, volume 8, 1937.

[78] Ahmed El Alaoui and Michael W. Mahoney. Fast Randomized Kernel Methods With Statistical Guarantees. *Stat*, 1050:1–17, nov 2014.

[79] Dino Oglic and Thomas Gärtner. Nyström Method with Kernel K-means++ Samples as Landmarks. *Proceedings of the 34th International Conference on Machine Learning*, 70:2652–2660, 2017.

[80] Yanshuai Cao, Marcus Brubaker, David Fleet, and Aaron Hertzmann. Efficient optimization for sparse Gaussian process regression. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, pages 1–1, 2015. ISSN 0162-8828.

[81] Francis Bach, Julien Mairal, Jean Ponce, and Guillermo Sapiro. Sparse coding and dictionary learning for image analysis. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2010.

[82] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67 (2):301–320, apr 2005. ISSN 1369-7412.

[83] Carl D Meyer. *Matrix Analysis and Applied Linear Algebra*. Siam, 2000.

[84] Corinna Cortes. Can learning kernels help performance? In *Invited talk at International Conference on Machine Learning (ICML 2009). Montreal, Canada*, 2009.

[85] Pascal Vincent and Yoshua Bengio. Kernel matching pursuit. *Machine Learning*, 48(1-3):165–187, 2002. ISSN 08856125. doi: 10.1023/A:1013955821559.

[86] Jean-Francois Ton, Seth Flaxman, Dino Sejdinovic, and Samir Bhatt. Spatial Mapping with Gaussian Processes and Nonstationary Fourier Features. *Spatial Statistics*, 2017. ISSN 2211-6753. doi: https://doi.org/10.1016/j.spasta.2018.02.002.

[87] Jesús Alcalá-Fdez, Luciano Sanchez, Salvador Garcia, Maria Jose del Jesus, Sebastian Ventura, Josep Maria Garrell, Jose Otero, Cristóbal Romero, Jaume Bacardit, Victor M Rivas, and Others. KEEL: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 13(3):307–318, 2009.

[88] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics (ACL)*, volume 7, pages 440–447, 2007.

[89] Mo Chen and James L Manley. Mechanisms of alternative splicing regulation: insights from molecular and genomics approaches. *Nature reviews. Molecular cell biology*, 10(11):741–54, 2009. ISSN 1471-0080. doi: 10.1038/nrm2777.

[90] Bin Tian, Jun Hu, Haibo Zhang, and Carol S Lutz. A large-scale analysis of mRNA polyadenylation of human and mouse genes. *Nucleic acids research*, 33(1):201–12, jan 2005. ISSN 1362-4962. doi: 10.1093/nar/gki158.

[91] Marina M. Scotti and Maurice S. Swanson. RNA mis-splicing in disease. *Nature Reviews Genetics*, nov 2015. ISSN 1471-0056. doi: 10.1038/nrg.2015.3.

[92] Jonathan P Ling, Olga Pletnikova, Juan C Troncoso, and Philip C Wong. TDP-43 repression of nonconserved cryptic exons is compromised in ALS-FTD. *Science*, 349(6248):650–655, 2015.

[93] Debashish Ray, Hilal Kazan, Esther T Chan, Lourdes Peña Castillo, Sidharth Chaudhry, Shaheynoor Talukder, Benjamin J Blencowe, Quaid Morris, and Timothy R Hughes. Rapid and systematic analysis of the RNA recognition specificities of RNA-binding proteins. *Nature biotechnology*, 27(7):667–70, jul 2009. ISSN 1546-1696. doi: 10.1038/nbt.1550.

[94] Kate B Cook, Hilal Kazan, Khalid Zuberi, Quaid Morris, and Timothy R Hughes. RBPDB: a database of RNA-binding specificities. *Nucleic acids research*, 39(Database issue):D301–8, jan 2011. ISSN 1362-4962. doi: 10.1093/nar/gkq1069.

[95] Xiang-Dong Fu and Manuel Ares Jr. Context-dependent control of alternative splicing by rna-binding proteins. *Nature Reviews Genetics*, 15(10): 689, 2014.

[96] Shatakshi Pandit, Yu Zhou, Lily Shiue, Gabriela Coutinho-Mansfield, Hairi Li, Jinsong Qiu, Jie Huang, Gene W Yeo, Manuel Ares, and Xiang-Dong Fu. Genome-wide analysis reveals SR protein cooperation and competition in regulated splicing. *Molecular cell*, 50(2):223–35, apr 2013. ISSN 1097-4164. doi: 10.1016/j.molcel.2013.03.001.

[97] Julian König, Kathi Zarnack, NM Nicholas M Luscombe, Jernej Ule, and Julian Koenig. Protein–RNA interactions: new genomic technologies and perspectives. *Nature Reviews Genetics*, 13(February):77–83, feb 2012. ISSN 1471-0056. doi: 10.1038/nrg3141.

[98] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[99] Tim Hesterberg, Nam Hee Choi, Lukas Meier, and Chris Fraley. Least angle and l-1 penalized regression: A review. *Statistics Surveys*, 2:61–93, 2008. ISSN 1935-7516.

[100] Ali Rahimi and Ben Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1177–1184, 2007.

[101] Yves-Laurent Kom Samo and Stephen Roberts. Generalized Spectral Kernels. *arXiv preprint arXiv:1506.00236*, 2015. ISSN 22113797. doi: 10.1016/j.rinp.2015.06.002.

[102] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 34(3):480–92, mar 2012. ISSN 1939-3539. doi: 10.1109/TPAMI.2011.153.

[103] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *International Conference on Machine learning (ICML)*, 2006.

[104] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[105] Philip J Uren, Suzanne C Burns, Jianhua Ruan, Kusum K Singh, Andrew D Smith, and Luiz O F Penalva. Genomic Analyses of the RNA-binding Protein Hu Antigen R (HuR) Identify a Complex Network of Target Genes and Novel Characteristics of Its Binding Sites. *Journal of Biological Chemistry*, 286(43):37063–37066, oct 2011. ISSN 0021-9258. doi: 10.1074/jbc.C111.266882.

[106] Minna-Liisa Aenkoe, Michaela Mueller-McNicoll, Holger Brandl, Tomaz Curk, Crtomir Gorup, Ian Henry, Jernej Ule, Karla M Neugebauer, Minna-Liisa Änkö, and Michaela Müller-McNicoll. The RNA-binding landscapes of two SR proteins reveal unique functions and binding to diverse RNA classes. *Genome biology*, 13(3):R17, jan 2012. ISSN 1465-6914. doi: 10.1186/gb-2012-13-3-r17.

[107] Jérôme Saulière, Valentine Murigneux, Zhen Wang, Emélie Marquenet, Isabelle Barbosa, Olivier Le Tonquèze, Yann Audic, Luc Paillard, Hugues Roest Crollius, and Hervé Le Hir. CLIP-seq of eIF4AIII reveals transcriptome-wide mapping of the human exon junction complex. *Nature structural & molecular biology*, 19(11):1124–1131, 2012.

[108] Markus Hafner, Markus Landthaler, Lukas Burger, Mohsen Khorshid, Jean Hausser, Philipp Berninger, Andrea Rothballer, Manuel Ascano, Anna-Carina Jungkamp, Mathias Munschauer, Alexander Ulrich, Greg S Wardle, Scott Dewell, Mihaela Zavolan, Thomas Tuschl, Manuel Ascano Jr, and Others. Transcriptome-wide identification of RNA-binding protein and microRNA target sites by PAR-CLIP. *Cell*, 141(1):129–41, apr 2010. ISSN 1097-4172. doi: 10.1016/j.cell.2010.03.009.

[109] Shivendra Kishore, Lukasz Jaskiewicz, Lukas Burger, Jean Hausser, Mohsen Khorshid, and Mihaela Zavolan. A quantitative analysis of CLIP methods for identifying binding sites of RNA-binding proteins. *Nature methods*, 8(7):559–64, jul 2011. ISSN 1548-7105. doi: 10.1038/nmeth.1608.

[110] Ryan L Boudreau, Peng Jiang, Brian L Gilmore, Ryan M Spengler, Rebecca Tirabassi, Jay A Nelson, Christopher A Ross, Yi Xing, and Beverly L Davidson. Transcriptome-wide Discovery of microRNA Binding Sites in Human Brain. *Neuron*, 81(2): 294–305, 2014.

[111] Jessica I Hoell, Erik Larsson, Simon Runge, Jeffrey D Nusbaum, Sujitha Duggimpudi, Thalia A Farazi, Markus Hafner, Arndt Borkhardt, Chris Sander, and Thomas Tuschl. RNA targets of wild-type and mutant FET family proteins. *Nature structural & molecular biology*, 18(12):1428–1431, 2011.

[112] Jernej Ule, Mikhail s Gelfand, and Albrecht Bindereif. Crosslinking-immunoprecipitation (iCLIP) analysis reveals global regulatory roles of hnRNP L. *RNA biology*, 11(2):146–155, 2014.

[113] Cem Sievers, Tommy Schlumpf, Ritwick Sawarkar, Federico Comoglio, and Renato Paro. Mixture models and wavelet transforms reveal high confidence RNA-protein interaction sites in MOV10 PAR-CLIP data. *Nucleic acids research*, 40(20):e160—e160, 2012.

[114] Shobbir Hussain, Abdulrahim A Sajini, Sandra Blanco, Sabine Dietmann, Patrick Lombard, Yoichiro Sugimoto, Maike Paramor, Joseph G Gleeson, Duncan T Odom, Jernej Ule, and Michaela Frye. NSun2-mediated cytosine-5 methylation of vault noncoding RNA determines its processing into regulatory small RNAs. *Cell reports*, 4(2):255–61, jul 2013. ISSN 2211-1247. doi: 10.1016/j.celrep.2013.06.029.

[115] James R Tollervey, Tomaz Curk, Boris Rogelj, Michael Briese, Matteo Cereda, Melis Kayikci, Julian Koenig, Tibor Hortobagyi, Agnes L Nishimura, Vera Zupunski, Rickie Patani, Siddharthan Chandran, Gregor Rot, Blaz Zupan, Christopher E Shaw, and Jernej Ule. Characterizing the RNA targets and position-dependent splicing regulation by TDP-43. *Nature Neuroscience*, 14(4):452–U180, apr 2011. ISSN 1097-6256. doi: 10.1038/nn.2778.

[116] Zhen Wang, Melis Kayikci, Michael Briese, Kathi Zarnack, Nicholas M Luscombe, Gregor Rot, Blaz Blaž Zupan, Tomaz Tomaž Curk, and Jernej Ule. iCLIP Predicts the Dual Splicing Effects of TIA-RNA Interactions. *PLOS Biology*, 8(10):e1000530, oct 2010. ISSN 1544-9173. doi: 10.1371/journal.pbio.1000530.