TECHNISCHE UNIVERSITÄT WIEN

DISSERTATION

Sufficient Dimension Reduction for Structured and Big Data

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften unter der Leitung von

Univ. Prof. Ph.D Efstathia Bura
E105 – Institut für Stochastik und Wirtschaftsmathematik, TU Wien
Angewandte Statistik

eingereicht an der Technischen Universität Wien
Fakultät für Mathematik und Geoinformation

von

Daniel Kapla
[redacted]

Diese Dissertation haben begutachtet:

1. **Univ. Prof., Ph.D. Efstathia Bura**
   Institut für Stochastik und Wirtschaftsmathematik, Technische Universität Wien, Österreich
2. **Prof., Ph.D. Liliana Forzani**
   Facultad de Ingeniería Química, Universidad Nacional del Litoral, Argentina
3. **Prof., Ph.D. Xin Zhang**
   Department of Statistics, Florida State University, USA

Wien, am 15. Mai 2024

Kurzfassung

Regression modelliert die bedingte Verteilung einer oder mehrerer Zufallsvariablen (Antwortvariablen) gegeben den Prädiktoren. Wenn die Prädiktoren hochdimensional sind, wird die Regressionsaufgabe noch herausfordernder. *Sufficient Dimension Reduction* (SDR; auf Deutsch: Ausschöpfende Dimensions Reduktion) begegnet diesem Problem, indem die Dimensionalität der Prädiktoren reduziert wird, während sämtliche relevante Information über die Zielvariable erhalten bleibt. Unter den SDR Methoden tragen lineare Projektionen auf niedrigdimensionale Teilräume nicht nur dazu bei, das Problem handhabbarer zu machen, sondern auch – was besonders wichtig ist – die Interpretierbarkeit zu verbessern.

In dieser Arbeit werden drei lineare SDR Methoden oder Methodenklassen vorgestellt. Die erste Methode, *Neural Network SDR* (NNSDR), behandelt effizient Datensätze mit enormem Datenvolumen und hoher Prädiktor-Dimensionalität. NNSDR kombiniert die lineare Projektion der Prädiktoren mit dem gleichzeitigen Lernen neuronaler Netzwerke. Dabei wird sowohl von der Anpassung des neuronalen Netzwerks als auch von vorwärts SDR gleichzeitig profitiert, um einen Prozess aufzubauen, der eine Vorhersagegenauigkeit bietet, die mit Neuronalen Netzwerken vergleichbar ist. Gleichzeitig bleibt die Transparenz bezüglich der Relevanz der Prädiktoren und der intrinsischen Regressionsdimension erhalten.

Die heute gesammelten Daten können strukturelle Merkmale aufweisen, die von früheren Regressionsmethoden nicht berücksichtigt werden und zu einem Verlust in Inferenz führt. Unsere zweite Methodenklasse führt SDR Inferenz bei Regressionen mit matrixwertigen Prädiktoren durch. Das allgemeine Problem eines Inferenzmodels für die bedingte Verteilung der Antwortvariablen gegeben tensorwertige Prädiktoren wird in unserer dritten Methodenklasse behandelt. *Generalized Multi-Linear Model* SDR ermöglicht flexible lineare Reduktionen, die die Informationen in beliebigen tensorwertigen Daten mit Verteilung in der quadratischen Exponentialfamilie erfassen. Wir zeigen die Konsistenz und asymptotische Normalität der ausschöpfenden Reduktion. Für kontinuierliche tensorwertige Prädiktoren entwickeln wir ein rechnerisch effizientes Schätzverfahren für ihre ausschöpfenden Reduktionen, das auch auf Situationen anwendbar ist, in denen die Dimension der Reduktion die verfügbare Stichprobengröße übersteigt. Für Regressionen mit binären tensorwertigen Prädiktoren orientiert sich das Schätzverfahren an Algorithmen, die zum Trainieren neuronaler Netzwerke verwendet werden, um große Mengen von Beobachtungen zu verarbeiten, die in allgemeinen hochdimensionalen diskreten Umgebungen oft erforderlich sind.

Abstract

Regression models the conditional distribution of a random variable(s) (response(s)) given a set of predictors. When the predictors are high-dimensional, the regression task becomes even more challenging. *Sufficient Dimension Reduction* (SDR) addresses this issue by reducing the dimensionality of the predictors while retaining all relevant information about the response variable. Among SDR methods, linear projections to lower dimensional subspaces make the problem not only more manageable but, importantly, also more interpretable.

Three linear SDR methods, or class of methods, are presented in this thesis. The first method, *Neural Network SDR*, efficiently handles datasets of huge data size and high predictor dimensionality. NNSDR combines linear projection of the predictors with neural network learning simultaneously borrowing strength from NN fitting and forward SDR to build a process that enjoys predictive accuracy comparable to NN methods while being transparent about predictor importance and intrinsic regression dimension.

The data collected today can exhibit structural features that past regression techniques do not take into consideration and that may result in inferential loss. Our second class of methods carry out SDR inference on regressions with matrix-valued predictors. The general problem of building inference models for the conditional distribution of a response given multi-way array-valued predictors is addressed in our third class of methods. *Generalized Multi-Linear Model* SDR, allows for flexible linear reductions capturing the information in arbitrary tensor-valued data with distribution in the quadratic exponential family. We prove the consistency and asymptotic normality of the sufficient reduction. For continuous tensor-valued predictors, we develop a computationally efficient estimation procedure of their sufficient reductions, which is also applicable to situations where the dimension of the reduction exceeds the available sample size. For regressions with binary tensor-valued predictors, the estimation procedure draws inspiration from algorithms used for training neural networks to be able to process large amounts of observations often required in general high-dimensional discrete settings.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 15. Mai 2024

Daniel Kapla

Contents

1 Introduction 1
- 1.1 Contribution of this Thesis . . . 1
- 1.2 General Notation . . . 2

2 Sufficient Dimension Reduction (SDR) 3
- 2.1 Central Subspace . . . 5
- 2.2 Central Mean Subspace . . . 6
- 2.3 Classic SDR methods . . . 7
  - 2.3.1 Sliced Inverse Regression (SIR) . . . 8
  - 2.3.2 Sliced Average Variance Estimation (SAVE) . . . 8
  - 2.3.3 Parametric Inverse Regression (PIR) . . . 10
  - 2.3.4 Principal Fitted Components (PFC) . . . 10
  - 2.3.5 Minimum Average Variance Estimation (MAVE) . . . 11
  - 2.3.6 Outer Product of Gradients (OPG) . . . 13
- 2.4 SDR methods for Matrix- or Tensor-Valued Predictors . . . 14
  - 2.4.1 Notes on Multilinear Algebra . . . 15
  - 2.4.2 Multilinear Algebra in R . . . 18
  - 2.4.3 Central Dimension-Folding Subspace . . . 19
  - 2.4.4 Longitudinal Sliced Inverse Regression (LSIR) . . . 20
  - 2.4.5 Tensor Sliced Inverse Regression (TSIR) . . . 22

3 Fusing SDR with Neural Networks 25
- 3.1 Multi Layer Perceptron (MLP) . . . 25
  - 3.1.1 A simple Neural Network Framework in R . . . 28
- 3.2 Neural Network OPG (NNOPG) . . . 34
- 3.3 Neural Network SDR (NNSDR) . . . 35
- 3.4 Dimension Estimation . . . 36
- 3.5 Algorithm . . . 37
  - 3.5.1 Implementation in R . . . 39
  - 3.5.2 MNIST handwritten digits dataset . . . 40

4 Kronecker Parametric Inverse Regression (KPIR) 43
- 4.1 Matrix-Valued Inverse Regression Models . . . 43
- 4.2 Estimation Procedures . . . 44
  - 4.2.1 Least Squares Kronecker PIR (KPIR (ls)) . . . 45
    - 4.2.1.1 Implementation in R . . . 46
  - 4.2.2 Maximum Likelihood Kronecker PIR (KPIR (mle)) . . . 46
    - 4.2.2.1 Implementation in R . . . 47

4.2.3 Kronecker PFC (KPFC) . . . 48
4.2.3.1 Implementation in R . . . 50

5 Generalized Multilinear Models **53**
5.1 The Generalized Multi-Linear Model (GMLM) . . . 53
5.2 Maximum Likelihood Estimation . . . 56
5.2.1 Tensor Normal . . . 60
5.2.1.1 Implementation in R . . . 62
5.2.2 Ising Model . . . 64
5.2.2.1 Initial Values . . . 66
5.2.2.2 Gradient Optimization . . . 67
5.2.2.3 Small Data Sets . . . 68
5.2.2.4 Slightly Bigger Dimensions . . . 68
5.2.2.5 Implementation in R . . . 69
5.3 Manifolds . . . 71
5.3.1 Matrix Manifolds . . . 74
5.3.2 Kronecker Product Manifolds . . . 75
5.4 Statistical Properties . . . 79
5.4.1 Asymptotic Consistency . . . 79
5.4.2 Asymptotic Normality . . . 81
5.5 Simulations . . . 87
5.5.1 Tensor Normal . . . 88
5.5.2 Ising Model . . . 89
5.6 Data Analysis . . . 93
5.6.1 EEG . . . 93
5.6.2 Chess . . . 94

6 Summary and Future Directions **99**

Notation Index **101**

References **103**

1 Introduction

Sufficient Dimension Reduction (SDR) is a statistical technique that combines statistical *sufficiency* with *dimension reduction* in regression and classification. The goal of SDR is to find a mapping of the predictors to a lower dimension space without changing the conditional distribution of the response. If such lower dimensional image of the predictors exists, it is *sufficient* for the regression or classification of the response. In other words, SDR replaces the input variable with a reduced version of itself while retaining all the information about the target variable.

The phrase "sufficient dimension reduction" with the meaning herein was introduced in the late 1990's [Coo98] in the context of regression graphics and a search of a "sufficient summary plot" of a response Y versus a transformation $\boldsymbol{R}(\boldsymbol{X})$ of the input (predictor) variables $\boldsymbol{X} \in \mathbb{R}^p$ that retains all of the relevant regression information. Professor Cook draws the distinction between SDR and the term "sufficient dimensionality reduction" in the machine learning literature by [GT03], where the underlying ideas are quite different from those associated with SDR and the methodology is not dimension reduction in the usual sense [see Bur10].

SDR methodology can be based on both likelihood (model) and non-likelihood (model-free) approaches. In the model-based approach, which was developed relatively more recently [BDF16; BF15; Coo07; CF08; CF09], either the joint family of distributions of $(Y, \boldsymbol{X})$, or the conditional family of distributions for $\boldsymbol{X} \mid Y$ is assumed to be known.

In the model-free approach, which is the most researched branch of SDR, reductions are typically constrained to be linear and the goal is to estimate the *mean subspace* [CL02] or *central subspace* [CN94] subspace. Model-free SDR comprises of three classes of methods: Inverse regression based, semi-parametric and nonparametric. For a review, see, e.g., [AC09; MZ13; Li18; GGK+21].

This thesis contains model-free and model-based SDR methods for a variety of regression and classification settings.

1.1 Contribution of this Thesis

This thesis starts with an introduction into SDR Chapter 2. In Chapter 3 we present the joint work with [KFB22] Lukas Fertl under the supervision of Prof. Efstathia Bura. Continuing in Chapter 4 we present [PKB21], where the author had both theoretical and application contributions. He devised all implementation algorithms, carried out the simulations, checked all theoretical results and revised and extended Theorem 1 in [PKB21], for which he also provided a new proof. The author also made publicly available the `R` implementation code for the methods and the simulations. This work also led to the main part of this thesis, the yet unpublished new method in Chapter 5 for regressions with tensor-valued

predictors, which generalizes the ideas from [PKB21] while exploiting some lessons learned in the development of [KFB22].

1.2 General Notation

Here we present the general notation used throughout. Specific notation linked to particular concepts introduced later is provided in conjunction with those concepts. For a full notation overview see the Notation Index.

Let $\boldsymbol{X}$ represent a p-dimensional measurable random vector. Unless specified otherwise, non-boldface X denotes a univariate random variable, that is $p = 1$.

For two random variables $\boldsymbol{X}$ and $\boldsymbol{Z}$, *independence* is indicated by $\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Z}$, while *equality in distribution* is denoted as $\boldsymbol{X} \sim \boldsymbol{Z}$notation1$\boldsymbol{X} \sim \boldsymbol{Z}$equal in distribution.

We use to the following convention for non-random objects, with exceptions. *Scalars* are represented by lower-case letters, such as a, b, p, q. *Vectors* are denoted by lower-case boldface letters, like $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{p}$, $\boldsymbol{q}$. *Matrices* are indicated by upper-case boldface letters, such as $\boldsymbol{A}$, $\boldsymbol{B}$.

For a matrix $\boldsymbol{A}$, the *transpose* is $\boldsymbol{A}^T$. If applied to a vector $\boldsymbol{a} \in \mathbb{R}^p$, the vector is treated as a $p \times 1$ matrix. For invertible matrices $\boldsymbol{A}$, the inverse is represented as $\boldsymbol{A}^{-1}$, while for all matrices, the notation $\boldsymbol{A}^\dagger$ denotes the *Moore-Penrose inverse* [AM05].

The *span* of a $p \times q$ dimensional matrix $\boldsymbol{A}$ is given by $\operatorname{span}(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{R}^q\}$, which is a subset of $\mathbb{R}^p$, representing the set of all linear combinations of the columns of $\boldsymbol{A}$.

The *vectorization* $\operatorname{vec}(\boldsymbol{A})$ of a $p \times q$ dimensional matrix $\boldsymbol{A}$ is a pq vector constructed by stacking the columns of $\boldsymbol{A}$. The *half vectorization* of a square matrix $\boldsymbol{A}$ of dimension $p \times p$ is a vector $\operatorname{vech}(\boldsymbol{A})$ of size $p(p+1)/2$, defined as the stacked columns of the lower triangular part of $\boldsymbol{A}$, including the diagonal elements.

For square matrices $\boldsymbol{A}$ we also have the *determinant* $\det(\boldsymbol{A})$ and its *trace* $\operatorname{tr}(\boldsymbol{A})$. Another conventient operation is the overloaded diagonal operator diag. There are two scenarios to consider. First, when the argument $\boldsymbol{x} \in \mathbb{R}^p$ is a vector, $\operatorname{diag}(\boldsymbol{x})$ represents a $p \times p$ diagonal matrix with its diagonal elements being $\boldsymbol{x}$. Second, for a square matrix $\boldsymbol{A} \in \mathbb{R}^{p \times p}$, $\operatorname{diag}(\boldsymbol{A})$ denotes a vector of dimension p comprising of the diagonal elements of $\boldsymbol{A}$.

The $p \times p$ *identity matrix* is represented by $\boldsymbol{I}_p$, while the *standard unit vectors* are denoted as $\boldsymbol{e}_j$, corresponding to the jth column of the identity matrix $\boldsymbol{I}_p$. Usually, the dimension of the standard unit vector is not explicitly provided if it is clear from the context.

2 Sufficient Dimension Reduction (SDR)

For an informal illustration of the general idea, consider the following regression models. Let Y and $\boldsymbol{X}$ be jointly distributed, ϵ be a mean zero error with finite variance. If Y is continuous, linear regression conjectures the model

$$Y = a + \boldsymbol{b}^T \boldsymbol{X} + \epsilon,$$

where is ϵ is a random variable with mean zero, constant variance and stochastically independent from $\boldsymbol{X}$. If Y is binary, the logistic regression model typically used is

$$\text{logit}(P(Y = 1 \mid \boldsymbol{X})) = a + \boldsymbol{b}^T \boldsymbol{X} + \epsilon,$$

where $\text{logit}(x) = \log x - \log(1 - x)$ is the logit function. Another example is the *generalized additive model* (GAM) [HT90],

$$\log(Y) = a + g_1(\boldsymbol{b}_1^T \boldsymbol{X}) + ... + g_q(\boldsymbol{b}_q^T \boldsymbol{X}) + \epsilon$$

where g_j are smooth functions for $j = 1, \ldots, q$. An example of heteroscedastic linear regression is

$$Y = a + \boldsymbol{b}_1^T \boldsymbol{X} + g(\boldsymbol{b}_2^T \boldsymbol{X})\epsilon,$$

where g is a real-valued function.

What all of those regression models have in common is their functional dependence on linear combinations of $\boldsymbol{X}$. In general, the assumption is that there exists a function g depending on $\boldsymbol{B}^T \boldsymbol{X}$, for $\boldsymbol{B} \in \mathbb{R}^{p \times q}$, describing the relation to the response Y with an error term ϵ independent of $\boldsymbol{X}$;

$$Y = g(\boldsymbol{B}^T \boldsymbol{X}, \epsilon). \tag{2.1}$$

Assuming that $q \leq p$, relation (2.1) means that the lower q-dimensional $\boldsymbol{B}^T \boldsymbol{X}$ contains all the information about Y in $\boldsymbol{X}$. We can replace $\boldsymbol{X}$ with the $q \leq p$ linearly transformed covariates without incurring any information loss.

In an abstract regression or classification problem, which is inference about the conditional distribution of the response Y given the predictors $\boldsymbol{X}$, (2.1) is equivalent to

$$Y \mid \boldsymbol{X} \sim Y \mid \boldsymbol{B}^T \boldsymbol{X}. \tag{2.2}$$

The goal of *linear* SDR is to estimate a linearly reduced set of variables $\boldsymbol{B}^T \boldsymbol{X}$ that satisfy (2.2). Relation (2.2) requires the conditional distribution $Y \mid \boldsymbol{X}$ and is referred to as *forward regression*. This allows the interpretation of $\boldsymbol{X}$ as observed data; that is, $\boldsymbol{X}$ can be considered fixed.

Provided $(Y, \boldsymbol{X})$ has a joint distribution, [Coo07] pointed out that

$$\boldsymbol{X} \mid (Y, \boldsymbol{B}^T \boldsymbol{X}) \sim \boldsymbol{X} \mid \boldsymbol{B}^T \boldsymbol{X} \tag{2.3}$$

is equivalent to (2.2). This formulation is referred to as *inverse regression* and requires only the inverse conditional distribution $\boldsymbol{X} \mid Y$. It is the basis for the word "sufficient" in SDR by relating SDR to statistical sufficiency [Fis22] by treating Y as non-random model parameter while the predictors $\boldsymbol{X}$ are considered as data. In this setup, the reduced variables $\boldsymbol{B}^T\boldsymbol{X}$ in (2.3) are a *sufficient statistic* for the *parameter* Y and, because of the equivalence of (2.3) and (2.2), it is also a *sufficient reduction* for the regression of Y on $\boldsymbol{X}$.

Another variation is the formulation given joint distribution $(Y, \boldsymbol{X})$ as

$$(Y \perp\!\!\!\perp \boldsymbol{X}) \mid \boldsymbol{B}^T\boldsymbol{X} \tag{2.4}$$

which is equivalent to (2.2) and (2.3) under its requirement of joint distribution.

Up to now, we only considered linear reductions. The methodology of SDR is applicable to non-linear reductions as well. For completeness, we define a sufficient reduction in the general setting.

Definition 1 (Sufficient Reduction [Coo07])**.** A reduction $\boldsymbol{R} : \mathbb{R}^p \to \mathbb{R}^q$ for $q \leq p$ is a *sufficient reduction* for the regression or classification problem $Y \mid \boldsymbol{X}$ if at least one of the following statements hold.

$$\begin{aligned} \text{Forward reduction: } & Y \mid \boldsymbol{X} \sim Y \mid \boldsymbol{R}(\boldsymbol{X}), \\ \text{Inverse reduction: } & \boldsymbol{X} \mid (Y, \boldsymbol{R}(\boldsymbol{X})) \sim \boldsymbol{X} \mid \boldsymbol{R}(\boldsymbol{X}), \\ \text{Joint reduction: } & (Y \perp\!\!\!\perp \boldsymbol{X}) \mid \boldsymbol{R}(\boldsymbol{X}). \end{aligned}$$

Remark 1. In Definition 1 there are no explicit assumptions about the nature of Y or $\boldsymbol{X}$. The response may be any random variable, univariate or multivariate, continuous or discrete. The predictors are treated here as p-dimensional vectors, but this is *not* limiting as they can be any structured random object such as matrix or tensor (multidimensional array), as will be done later. This is due to the structural equivalence (isomorphic spaces, inducing the "same" probability structure) of the underlying probability spaces induced by bijections between the vectorized versions of matrices (tensors) and vectors (with possibly additional structure. See Section 2.4.3 and Chapters 4 and 5).

This thesis focuses on *linear* SDR, which we simply refer to as SDR in the sequel. Specifically, we assume there exists a matrix $\boldsymbol{B} \in \mathbb{R}^{p\times q}$ such that the sufficient reduction in Definition 1 has the form $\boldsymbol{R}(\boldsymbol{X}) = \boldsymbol{B}^T\boldsymbol{X}$.

In the linear SDR setting, the coordinate matrix $\boldsymbol{B}$ is *not* identifiable. This can be seen easily from (2.1) as follows. Let $\boldsymbol{A}$ be an invertible $q \times q$ matrix, then

$$Y = g(\boldsymbol{B}^T\boldsymbol{X}, \epsilon) = g(\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{X}, \epsilon) = \tilde{g}(\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{X}, \epsilon) = \tilde{g}(\widehat{\boldsymbol{B}}^T\boldsymbol{X}, \epsilon)$$

where $\widehat{\boldsymbol{B}} = \boldsymbol{B}\boldsymbol{A}^T$. Replacing $\boldsymbol{B}$ with $\widehat{\boldsymbol{B}}$ leads to the same statements as before, just with different coordinates. Only the subspace $\operatorname{span}\boldsymbol{B} = \operatorname{span}\widetilde{\boldsymbol{B}}$ is invariant. Abstracting this from a specific coordinate matrix $\boldsymbol{B}$ gives the actual target of SDR as the estimation of a subspace $\mathcal{S} \subseteq \mathbb{R}^p$. If $\mathcal{S} = \operatorname{span}\boldsymbol{B}$ such that $\boldsymbol{R}(\boldsymbol{X}) = \boldsymbol{B}^T\boldsymbol{X}$ is a sufficient reduction by Definition 1, then the linear subspace $\mathcal{S}$ is called a *dimension reduction subspace* (Li [Li91] introduced this term).

Definition 2 (Dimension Reduction Subspace)**.** Let $\boldsymbol{B} \in \mathbb{R}^{p\times q}$ for $q \leq p$ and suppose $\boldsymbol{R}(\boldsymbol{X}) = \boldsymbol{B}^T\boldsymbol{X}$ is a sufficient reduction of the regression problem $Y \mid \boldsymbol{X}$. The subspace $\mathcal{S} = \operatorname{span}\boldsymbol{B}$ is called a *dimension reduction subspace*.

2.1 Central Subspace

Ideally, the maximum reduction in dimension is sought. Therefore, the aim is to infer the *smallest* dimension reduction subspace or *central subspace* $\mathcal{S}_{Y|\boldsymbol{X}}$ [CN94; Coo98].

Definition 3 (Central Subspace)**.** The *central subspace* $\mathcal{S}_{Y|\boldsymbol{X}}$ for the regression or classification problem $Y \mid \boldsymbol{X}$ is defined to be the intersection of all dimension reduction subspaces,

$$\mathcal{S}_{Y|\boldsymbol{X}} = \bigcap \{\mathcal{S} \subseteq \mathbb{R}^p : \mathcal{S} \text{ is a dimension reduction subspace for } Y \mid \boldsymbol{X}\}$$

if and only if it is itself a dimension reduction subspace.

In Definition 3 the central subspace in defined under the condition that the intersection of dimension reduction subspaces is itself a dimension reduction subspace. This raises a valid first question; does the central subspace exist? The general answer is *no*. Without any additional assumptions the existence *cannot* be guaranteed. Consider the following counter example; Let $\boldsymbol{X} = (X_1, X_2) \in \mathbb{R}^2$ be uniformly distributed on the unit circle $\{\boldsymbol{x} \in \mathbb{R}^2 : x_1^2 + x_2^2 = 1\}$, and $Y = X_1^2 + \epsilon$ with $\epsilon \perp\!\!\!\perp \boldsymbol{X}$ being standard normal distributed $\epsilon \sim \mathcal{N}(0,1)$. Then $Y = (\boldsymbol{e}_1^T\boldsymbol{X})^2 + \epsilon = 1 - (\boldsymbol{e}_2^T\boldsymbol{X})^2 + \epsilon$ which means that both $\operatorname{span}(\boldsymbol{e}_1)$ and $\operatorname{span}(\boldsymbol{e}_2)$ are dimension reduction subspaces. Therefore, the intersection of the dimension reduction subspaces is a subset of the singleton $\{\boldsymbol{0}\} = \operatorname{span}(\boldsymbol{e}_1) \cap \operatorname{span}(\boldsymbol{e}_2)$, which is *not* a dimension reduction subspace.

To ensure the existence of the central subspace, some (mild) assumptions are needed. For example [Coo98, Prop. 6.4] gives the existence of the central subspace for continuous $\boldsymbol{X}$ by assuming that $\boldsymbol{X}$ has a density with convex support. For more details see [Coo98; CL95; YLC08; CC02].

If the central subspace exists, it is unique by construction. Methods for estimating the central subspace $\mathcal{S}_{Y|\boldsymbol{X}}$ often require conditions with the most prominent being the following two [Coo18];

Condition 1 (Linearity Condition)**.** Let $\boldsymbol{B} \in \mathbb{R}^{p\times q}$ be a basis for the central subspace $\mathcal{S}_{Y|\boldsymbol{X}}$, that is $\operatorname{span}\boldsymbol{B} = \mathcal{S}_{Y|\boldsymbol{X}}$, then $\mathbb{E}[\boldsymbol{X} \mid \boldsymbol{B}^T\boldsymbol{X}]$ is a linear function of $\boldsymbol{B}^T\boldsymbol{X}$.

Condition 2 (Constant Variance Condition)**.** Let $\boldsymbol{B}$ as in Condition 1, then $\operatorname{Var}(\boldsymbol{X} \mid \boldsymbol{B}^T\boldsymbol{X})$ is constant (not random).

Cook [Coo98, Prop. 4.2] shows the equivalence of Condition 1 to

$$\mathbb{E}[\boldsymbol{X} \mid \boldsymbol{B}^T\boldsymbol{X}] - \mathbb{E}[\boldsymbol{X}] = \boldsymbol{P}_{Y|\boldsymbol{X}}(\boldsymbol{\Sigma}_{\boldsymbol{X}})^T(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])$$

where $\boldsymbol{\Sigma}_{\boldsymbol{X}} = \operatorname{Var}(\boldsymbol{X})$ and $\boldsymbol{P}_{Y|\boldsymbol{X}}(\boldsymbol{\Sigma}_{\boldsymbol{X}})$ is the projection onto $\mathcal{S}_{Y|\boldsymbol{X}}$ with respect to the inner product corresponding to the positive definite matrix $\boldsymbol{\Sigma}_{\boldsymbol{X}}$, that is

$$\boldsymbol{P}_{Y|\boldsymbol{X}}(\boldsymbol{\Sigma}_{\boldsymbol{X}}) = \boldsymbol{B}(\boldsymbol{B}^T\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{B})^{-1}\boldsymbol{B}^T\boldsymbol{\Sigma}_{\boldsymbol{X}}.$$

A usefull implication [Li91, Thm. 3.1] of the linearity Condition 1 is that the *first moment SDR* subspace $\mathcal{S}_{\text{FMSDR}}$ defind as

$$\mathcal{S}_{\text{FMSDR}} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \operatorname{span}(\mathbb{E}[\boldsymbol{X} \mid Y] - \mathbb{E}[\boldsymbol{X}]) \tag{2.5}$$

is a subset of the central subspace; i.e., $\mathcal{S}_{\text{FMSDR}} \subseteq \mathcal{S}_{Y|\boldsymbol{X}}$. By this relation the first moment SDR subspace $\mathcal{S}_{\text{FMSDR}}$ is the direct target of many inverse regression methods.

Another useful relation [Li18; Coo98] is

$$\mathcal{S}_{Y|\boldsymbol{X}} = \boldsymbol{A}^T \mathcal{S}_{Y|(\boldsymbol{A}\boldsymbol{X}-\boldsymbol{b})} \tag{2.6}$$

for any invertible $\boldsymbol{A} \in \mathbb{R}^{p\times p}$ and arbitrary vectors $\boldsymbol{b} \in \mathbb{R}^p$. This allows to work with standardized data $\boldsymbol{Z} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1/2}(\boldsymbol{X} - \mathbb{E}\,\boldsymbol{X})$ instead as

$$\mathcal{S}_{Y|\boldsymbol{X}} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1/2} \mathcal{S}_{Y|\boldsymbol{Z}}.$$

2.2 Central Mean Subspace

A major disadvantage of the central subspace is the challenge of estimation. This is due to the dependence of Y on $\boldsymbol{X}$ which can manifest itself in any conditional moment. Depending on the needs of the data analysis, those higher moments may not be of interest. A classic example of this is to use SDR as a preprocessing tool for further prediction models which struggle with the curse of dimensionality. If the goal of the prediction is purely to give an estimate of the expected value, only the information about the first moment $\mathbb{E}[Y \mid \boldsymbol{X}]$ is needed. This is the purpose of *mean dimension reduction subspaces* introduced in Cook and Li [CL02].

Definition 4 (Mean Dimension Reduction Subspace [CL02, Def. 1])**.** Let $\mathcal{S} \subseteq \mathbb{R}^p$ be a subspace with basis matrix $\boldsymbol{B} \in \mathbb{R}^{p\times q}$, that is $\mathcal{S} = \operatorname{span} \boldsymbol{B}$. if

$$Y \perp\!\!\!\perp \mathbb{E}[Y \mid \boldsymbol{X}] \mid \boldsymbol{B}^T\boldsymbol{X},$$

then $\mathcal{S}$ is a *mean dimension reduction subspace* for the regression of Y on $\boldsymbol{X}$.

The defining property $Y \perp\!\!\!\perp \mathbb{E}[Y \mid \boldsymbol{X}] \mid \boldsymbol{B}^T\boldsymbol{X}$ is equivalent to the following statements [CL02, Thm. 1];

$$\operatorname{Cov}(Y, \mathbb{E}[Y \mid \boldsymbol{X}] \mid \boldsymbol{B}^T\boldsymbol{X}) = 0,$$
$$\mathbb{E}[Y \mid \boldsymbol{X}] \text{ is a function of } \boldsymbol{B}^T\boldsymbol{X}.$$

As in the case of the central subspace, the smallest (if it exists) mean dimension reduction subspace is the main goal of inference.

Definition 5 (Central Mean Subspace [CL02, Def. 2])**.** The *central mean subspace* $\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$ for the regression or classification problem $Y \mid \boldsymbol{X}$ is defined to be the intersection of all mean dimension reduction subspaces,

$$\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]} = \bigcap \{\mathcal{S} \subseteq \mathbb{R}^p : \mathcal{S} \text{ is a mean dimension reduction subspace for } Y \mid \boldsymbol{X}\}$$

if and only if it is itself a mean dimension reduction subspace.

Similar to the central subspace the existence of the central mean subspace can not be guaranteed. Similar assumptions to those required for the existence of the central subspace, for example if $\boldsymbol{X}$ has open and convex domain, ensure the central mean subspace exists.

In general, any dimension reduction subspace is also a mean dimension reduction subspace. Moreover, the central mean subspace is always a subset of the central subspace $\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]} \subseteq \mathcal{S}_{Y|\boldsymbol{X}}$, if they exist.

Example 1 (Different mean and central subspace)**.** *Let* $\boldsymbol{X} = (X_1, X_2, X_3) \sim \mathcal{N}_3(\boldsymbol{0}, \boldsymbol{I}_3)$, $\epsilon \sim \mathcal{N}(0,1)$, *and*

$$Y = X_1 + \epsilon X_2$$

By $\boldsymbol{X} \perp\!\!\!\perp \epsilon$, *the conditional distribution of* Y *given* $\boldsymbol{X}$ *is an affine linear transformation of a standard normal variable* ϵ, *obtaining* $Y \mid \boldsymbol{X} \sim \mathcal{N}(X_1, X_2^2)$. *With* $\mathbb{E}[Y \mid \boldsymbol{X}] = X_1 = \boldsymbol{e}_1^T \boldsymbol{X}$ *we get the mean subspace* $\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]} = \operatorname{span} \boldsymbol{e}_1$. *The central subspace, on the other hand, captures all the information about the conditional distribution, includes the variance, therefore* $\mathcal{S}_{Y|\boldsymbol{X}} = \operatorname{span}(\boldsymbol{e}_1, \boldsymbol{e}_2)$.

A common assumption is to assume a *mean regression model* with additive error

$$Y = g(\boldsymbol{B}^T \boldsymbol{X}) + \epsilon \tag{2.7}$$

with zero conditional mean $\mathbb{E}[\epsilon \mid \boldsymbol{X}] = 0$ and finite variance $\operatorname{Var}(\epsilon) < \infty$. This is the case for the first three of the four examples at the beginning of this chapter. Under this model the central subspace and the central mean subspace, if they exist, are identical $\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]} = \mathcal{S}_{Y|\boldsymbol{X}}$.

2.3 Classic SDR methods

Before we describe some of the classic SDR methods, we introduce some convenient notation.

Let $\boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{\Sigma})$ be the projection onto the span of a full column rank matrix $\boldsymbol{A}$ with respect to the inner product induced by a symmetric positive definite (PSD) matrix $\boldsymbol{\Sigma}$ given by

$$\boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{\Sigma}) = \boldsymbol{A}(\boldsymbol{A}^T \boldsymbol{\Sigma} \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{\Sigma}.$$

When $\boldsymbol{B}$ is a basis of the central subspace; that is, $\mathcal{S}_{Y|\boldsymbol{X}} = \operatorname{span}(\boldsymbol{B})$, then we write $\boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{\Sigma})$ also as $\boldsymbol{P}_{Y|\boldsymbol{X}}(\boldsymbol{\Sigma})$. This avoids the need to specify $\boldsymbol{B}$. We write $\boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{I}_p)$ simply as $\boldsymbol{P}_{\boldsymbol{A}}$, which in addition to being idempotent is always symmetric. The projection onto the orthogonal complement under the inner product corresponding to $\boldsymbol{\Sigma}$ is writen as $\boldsymbol{Q}_{\boldsymbol{A}}(\boldsymbol{\Sigma}) = \boldsymbol{\Sigma} - \boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{\Sigma})$ with the same abbreviations as for $\boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{\Sigma})$.

Observing that the conditioning on $\boldsymbol{B}^T \boldsymbol{X}$ is equivalent to conditioning on $\boldsymbol{P}_{\boldsymbol{B}} \boldsymbol{X}$ we get $\mathbb{E}[\boldsymbol{X} \mid \boldsymbol{B}^T \boldsymbol{X}] = \mathbb{E}[\boldsymbol{X} \mid \boldsymbol{P}_{\boldsymbol{B}} \boldsymbol{X}]$. The same applies to other conditional expressions.

In many methods it is assumed that the parameter matrix $\boldsymbol{B}$ identifying a subspace through its span is semi-orthogonal. The set of all $p \times q$ semi-orthogonal matrices with $q \leq p$ is called the *Stiefel manifold* (see Section 5.3.1)

$$\mathrm{St}^{p \times q} = \{\boldsymbol{A} \in \mathbb{R}^{p \times q} : \boldsymbol{A}^T \boldsymbol{A} = \boldsymbol{I}_q\}. \tag{2.8}$$

The set for $q > p$ with the condition $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}_p$, is also called the Stiefel manifold. To avoid confusion, we only use the term or notation for the Stiefel manifold in the case of $q \leq p$.

2.3.1 Sliced Inverse Regression (SIR)

The method deemed to kick start the modern understanding of SDR is *Sliced Inverse Regression* (SIR) [Li91]. SIR is based on the idea of the centered *inverse regression curve* (IRC) $\mathbb{E}[\boldsymbol{X} \mid Y] - \mathbb{E}[\boldsymbol{X}]$ being contained in a lower dimensional subspace $\mathcal{S}_{\text{SIR}}$. With the linearity Condition 1, [Li91, Prop. 3.1] states that the centered IRC is contained in $\operatorname{span}(\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{B})$, where $\boldsymbol{\Sigma}_{\boldsymbol{X}} = \operatorname{Var}\boldsymbol{X}$. As a consequence, the variance of the IRC, $\operatorname{Var}(\mathbb{E}[\boldsymbol{X} \mid Y])$, is singular. The eigenvectors of $\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\operatorname{Var}(\mathbb{E}[\boldsymbol{X} \mid Y])$ with non-zero eigenvalues span the linear subspace $\mathcal{S}_{\text{SIR}} \subseteq \operatorname{span}\boldsymbol{B} = \mathcal{S}_{Y|\boldsymbol{X}}$.

To estimate $\boldsymbol{B}$, SIR first standardizes the i.i.d. samples $(Y_i, \boldsymbol{X}_i)$ as $\widehat{\boldsymbol{Z}}_i = \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}^{-1/2}(\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}})$. The estimates $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}$ and $\widehat{\boldsymbol{\mu}}$ are the usual moment estimates of the marginal variance $\operatorname{Var}(\boldsymbol{X})$ and the mean $\mathbb{E}[\boldsymbol{X}]$, respectively. With the standardized predictors $\widehat{\boldsymbol{Z}}_i$, we need to estimate $\mathbb{E}[\boldsymbol{Z} \mid Y]$, which is done by *slicing*. The response domain is split into a finite set of mutually disjoint *slices* and replace the continuous responce Y with a discrete version encoding slice membership. This approach will occur multiple times and deserves its own definition.

Definition 6 (Slices)**.** Let Y be a univariate random variable. For continuous Y, a *slicing* of size s of n observations $Y_i \in \mathbb{R}$ of Y is a disjoint cover of $\{Y_i : i = 1, \dots n\} \subset \mathbb{R}$ consisting of s intervals S_j, called *slices*, each containing at least one observation Y_i. For categorical Y, the slices typically coincide with the values of Y.

In both cases, n_j is the number of observations contained in the slice

$$n_j = \sum_{\substack{i=1 \\ Y_i \in S_j}}^{n} 1 = \sum_{i \in S_j} 1$$

where the summation over $i \in S_j$ is a short hand notation for the summation over all $i = 1, \dots, n$ such that $Y_i \in S_j$.

With s slices S_j of Y, we compute the *within slice* means $\widehat{\boldsymbol{m}}_j = n_j^{-1}\sum_{i \in S_j}\widehat{\boldsymbol{Z}}_i$. The estimate $\widehat{\boldsymbol{\Sigma}}_{\mathbb{E}[\boldsymbol{Z}|Y]}$ of $\operatorname{Var}(\mathbb{E}[\boldsymbol{Z} \mid Y])$ is then defined to be

$$\widehat{\boldsymbol{\Sigma}}_{\mathbb{E}[\boldsymbol{Z}|Y]} = \frac{1}{n}\sum_{j=1}^{s}(\widehat{\boldsymbol{m}}_j - \widehat{\boldsymbol{\mu}})(\widehat{\boldsymbol{m}}_j - \widehat{\boldsymbol{\mu}})^T.$$

where $\widehat{\boldsymbol{\mu}}$ is the overall sample mean. The q dimensional SIR estimate is then given by $\mathcal{S}_{\text{SIR}} = \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}^{-1/2}\operatorname{span}(\boldsymbol{v}_1, \dots, \boldsymbol{v}_q)$ with $\boldsymbol{v}_1, \dots, \boldsymbol{v}_q$ being the first q eigenvectors of $\widehat{\boldsymbol{\Sigma}}_{\mathbb{E}[\boldsymbol{Z}|Y]}$.

Remark 2. Including discrete random variables in Definition 6 is a simple way to extend all slicing based methods to discrete random variables without any additional nomenclature. This is a natural extension because slicing is nothing else than converting a continuous random variable into a discrete random variable.

2.3.2 Sliced Average Variance Estimation (SAVE)

[CW91] pointed out that SIR is non-exhaustive for the central subspace. The problem is illustrated with a slightly adapted example. Let $(X_1, X_2) = \boldsymbol{X} \sim \mathcal{N}_2(\boldsymbol{0}, \boldsymbol{I}_2)$ be bivariate

standard normal and $Y = X_1^2 + \epsilon$ with $\epsilon \perp\!\!\!\perp \boldsymbol{X}$, zero mean and finite variance. The central subspace is $\mathcal{S}_{Y|\boldsymbol{X}} = \operatorname{span} \boldsymbol{e}_1$ and the regression model fulfills all the requirements of SIR, but the conditional mean $\mathbb{E}[\boldsymbol{X} \mid Y] = \boldsymbol{0}$. As a result, the SIR subspace $\mathcal{S}_{\text{SIR}} = \{\boldsymbol{0}\}$, "which is not a very interesting subspace" [CW91, direct citation].

To overcome this issue, [CW91] also introduced a new method called *Sliced Average Variance Estimation* (SAVE). In addition to the assumptions of SIR the constant variance Condition 2 is required. Based on the same concept, SAVE uses the conditional variance $\operatorname{Var}(\boldsymbol{X} \mid Y)$ instead of the conditional mean $\mathbb{E}[\boldsymbol{X} \mid Y]$. For the standardized variables $\boldsymbol{Z} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1/2}(\boldsymbol{X} - \mathbb{E}\,\boldsymbol{X})$, the law of total variance gives

$$\begin{aligned}
\boldsymbol{I}_p = \operatorname{Var}(\boldsymbol{Z}) &= \operatorname{Var}(\mathbb{E}[\boldsymbol{Z} \mid \boldsymbol{P}_{Y|\boldsymbol{Z}}\boldsymbol{Z}]) + \mathbb{E}[\operatorname{Var}(\boldsymbol{Z} \mid \boldsymbol{P}_{Y|\boldsymbol{Z}}\boldsymbol{Z})] \\
&= \boldsymbol{P}_{Y|\boldsymbol{Z}} \operatorname{Var}(\boldsymbol{Z}) \boldsymbol{P}_{Y|\boldsymbol{Z}} + \operatorname{Var}(\boldsymbol{Z} \mid \boldsymbol{P}_{Y|\boldsymbol{Z}}\boldsymbol{Z}) \\
&= \boldsymbol{P}_{Y|\boldsymbol{Z}} + \operatorname{Var}(\boldsymbol{Z} \mid \boldsymbol{P}_{Y|\boldsymbol{Z}}\boldsymbol{Z}).
\end{aligned}$$

where the linearity Condition 1 and constant variance Condition 2 was used in conjunction with $\boldsymbol{P}_{Y|\boldsymbol{Z}}$ being symmetric and idempotent. Rearranging gives

$$\operatorname{span}(\boldsymbol{I}_p - \operatorname{Var}(\boldsymbol{Z} \mid \boldsymbol{P}_{Y|\boldsymbol{Z}}\boldsymbol{Z})) = \operatorname{span} \boldsymbol{P}_{Y|\boldsymbol{Z}} \subseteq \mathcal{S}_{Y|\boldsymbol{Z}}.$$

Reversing the standardization on the central subspace with (2.6) yields the SAVE subspace.

$$\mathcal{S}_{\text{SAVE}} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1/2} \operatorname{span}(\boldsymbol{I}_p - \operatorname{Var}(\boldsymbol{Z} \mid \boldsymbol{P}_{Y|\boldsymbol{Z}}\boldsymbol{Z})) \subseteq \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1/2} \mathcal{S}_{Y|\boldsymbol{Z}} = \mathcal{S}_{Y|\boldsymbol{X}}.$$

The sample version is based on the same slicing idea as SIR. Let $(Y_i, \boldsymbol{X}_i)$ be n i.i.d. observations. We start by standardizing $\boldsymbol{X}_i$ as $\boldsymbol{Z}_i = \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}^{-1/2}(\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}})$ where $\widehat{\boldsymbol{\mu}}$ and $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}$ are the sample esimates of the mean and covariance of $\boldsymbol{X}$. Then, as in SIR, split the range of the Y_i's into s slices and let $\tilde{Y}_i = j$ if Y_i is in the j'th slice. For every slice $j = 1, \ldots, s$, the conditional covariance is estimated as

$$\widehat{\boldsymbol{\Sigma}}_j = \frac{1}{n_j} \sum_{\substack{i=1 \\ \tilde{Y}_i = j}} \boldsymbol{Z}_i \boldsymbol{Z}_i^T$$

where n_j is the number of responces Y_i in slice nr. j, that is $\tilde{Y}_i = j$. Next, we estimate $(\boldsymbol{I}_p - \operatorname{Var}(\boldsymbol{Z} \mid \boldsymbol{P}_{Y|\boldsymbol{Z}}\boldsymbol{Z}))^2$. The square comes from the idea that both matrices have the same span, but its square is guaranteed to be positive semidefinite. This is done by weighted combining the slice variances as

$$\widehat{\boldsymbol{V}} = \sum_{j=1}^{s} \frac{n_j}{n} (\boldsymbol{I}_p - \widehat{\boldsymbol{\Sigma}}_j)^2.$$

The final estimate is given by computing the q eigenvectors $\boldsymbol{v}_k$ of $\widehat{\boldsymbol{V}}$ for the largest eigenvalues with $k = 1, \ldots, q$. The SAVE estimate of a basis is then $\widehat{\boldsymbol{B}} = \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}^{-1/2}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_q)$.

2.3.3 Parametric Inverse Regression (PIR)

Proposed in [BC01] the method *Parametric Inverse Regression* (PIR) was developed to estimate the dimension of the central subspace $\mathcal{S}_{Y|\boldsymbol{X}}$. Setting up the scene by observing that the slicing performed in SIR results in p-univariate regressions on a discrete random variable. The idea is now to use smooth parametric curves instead of the indicator functions on the slices and use a multivariate linear model instead of $\operatorname{Var}(\mathbb{E}[\boldsymbol{X} \mid Y])$ to recover $\mathcal{S}_{\text{FMSDR}}$ in (2.5).

Let f_j be s known smooth functions collected in a vector valued function $\boldsymbol{f}(y)$, for example $\boldsymbol{f}(y) = (y, y^2, \ldots, y^s)$. The multivariate linear regression model is then

$$\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{A}(\boldsymbol{f}(Y) - \mathbb{E}[\boldsymbol{f}(Y)]) + \epsilon \tag{2.9}$$

where $\mathbb{E}[\epsilon \mid Y] = \boldsymbol{0}$ and $\operatorname{Var}(\epsilon \mid Y) = \operatorname{Var}(\boldsymbol{X} \mid Y) = \boldsymbol{\Sigma}_{\boldsymbol{X}|Y}$. Under model (2.9), the linearity Condition 1 holds, which yields $\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \operatorname{span}(\boldsymbol{A}) \subseteq \mathcal{S}_{Y|\boldsymbol{X}}$.

Solving the least squares problem at the population level yields

$$\boldsymbol{A} = \operatorname{Cov}(\boldsymbol{X}, \boldsymbol{f}(Y)) \operatorname{Var}(\boldsymbol{f}(Y))^{-1} \qquad \text{and} \qquad \boldsymbol{\mu} = \mathbb{E}\,\boldsymbol{X} - \boldsymbol{A}\,\mathbb{E}\,\boldsymbol{f}(Y)$$

leading to $\mathcal{S}_{\text{PIR}} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \operatorname{span}(\boldsymbol{A}) \subseteq \mathcal{S}_{Y|\boldsymbol{X}}$ [BC01; PKB21; Li18].

A method for estimation on the sample level given n i.i.d. observations $(\boldsymbol{X}_i, Y_i)$ proceeds as follows. Let $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}$, $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X},\boldsymbol{F}}$ and $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{F}}$ be the moment estimates of $\operatorname{Var}(\boldsymbol{X})$, $\operatorname{Cov}(\boldsymbol{X}, \boldsymbol{f}(Y))$ and $\operatorname{Var}(\boldsymbol{F})$, respectively. Then compute the q first left singular vectors of $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}^{-1} \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X},\boldsymbol{F}} \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{F}}^{-1}$ to get the estimate $\widehat{\boldsymbol{B}} \in \mathbb{R}^{p \times q}$ for $\operatorname{span}(\boldsymbol{B}) \subseteq \mathcal{S}_{Y|\boldsymbol{X}}$.

Remark 3. Given that $\operatorname{span}(\boldsymbol{A}) = \operatorname{span}(\boldsymbol{AC})$ for any invertible $q \times q$ matrix $\boldsymbol{C}$ we can set $\boldsymbol{C} = \operatorname{Var}(\boldsymbol{f}(Y))^{-1} \operatorname{Cov}(\boldsymbol{f}(Y), \boldsymbol{X}) \operatorname{Var}(\boldsymbol{X}^{-1})$. This leads to a generalized eigenvalue problem also usable as a basis for a sample level algorithm. For example [Li18] uses this approach.

Remark 4. Letting $\boldsymbol{f}(y) = y$, the univariate identity, PIR reduces to ordinary least squares (OLS) as $\widehat{\boldsymbol{B}} \propto \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}^{-1} \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X},Y} \in \mathbb{R}^{p \times 1}$, the well known OLS estimate.

Remark 5. PIR is also a generalization of SIR. This can be seen by setting the f_j's to indicator functions over the slices of SIR.

2.3.4 Principal Fitted Components (PFC)

The concept of *Principal Fitted Components* (PFC) was introduced in [Coo07] and extended in [CF08]. In a nutshell, PFC is a slightly restricted model-based version of PIR by assuming $\boldsymbol{X} \mid Y$ to be multivariate normal. The conditional inverse regression model is similar to (2.9) and given by

$$\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{\Gamma\gamma}(\boldsymbol{f}(Y) - \mathbb{E}[\boldsymbol{f}(Y)]) + \epsilon \tag{2.10}$$

with the additional assumption of $\epsilon \sim \mathcal{N}_p(\boldsymbol{0}, \boldsymbol{\Sigma}_\epsilon)$ and $\boldsymbol{\Gamma\gamma} \in \mathbb{R}^{p \times s}$ is of known rank q, that is $\boldsymbol{\Gamma}$ is a $p \times q$ full rank matrix and $\boldsymbol{\gamma}$ is $q \times s$ also full rank q with $q \leq \min(p, s)$. Note that $\boldsymbol{A} = \boldsymbol{\Gamma\gamma}$ in (2.9).

Given the full probabilistic model (2.10) and an i.i.d. sample $(\boldsymbol{X}_i, Y_i)$, in [CF08] an MLE estimate of $\mathcal{S}_{\text{PIR}} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \operatorname{span}(\boldsymbol{\Gamma}) \subseteq \mathcal{S}_{Y|\boldsymbol{X}}$ was derived. For the exact form, let $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}$, $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X},\boldsymbol{F}}$ and $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{F}}$ be the moment estimates of $\operatorname{Var}(\boldsymbol{X})$, $\operatorname{Cov}(\boldsymbol{X}, \boldsymbol{f}(Y))$ and $\operatorname{Var}(\boldsymbol{F})$, respectively.

Then, the OLS estimate for $\boldsymbol{A}$, as in the PIR model, for the regression of $\boldsymbol{X}$ on $\boldsymbol{f}(Y)$ is given by $\widehat{\boldsymbol{A}} = \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X},\boldsymbol{F}}^{-1}\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{F}}$. Then, define the fitted and residual covariance estimates as

$$\widehat{\boldsymbol{\Sigma}}_{\text{fit}} = \widehat{\boldsymbol{A}}\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{F}}\widehat{\boldsymbol{A}}^T, \qquad \widehat{\boldsymbol{\Sigma}}_{\text{res}} = \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}} - \widehat{\boldsymbol{\Sigma}}_{\text{fit}}.$$

The next step is to compute the SVD of $\widehat{\boldsymbol{\Sigma}}_{\text{res}}^{-1/2}\widehat{\boldsymbol{\Sigma}}_{\text{fit}}\widehat{\boldsymbol{\Sigma}}_{\text{res}}^{-1/2} = \widehat{\boldsymbol{U}}\,\text{diag}(\widehat{\lambda}_1,\dots,\widehat{\lambda}_p)\widehat{\boldsymbol{U}}^T$ to get

$$\widehat{\boldsymbol{\Sigma}}_{\text{MLE}} = \widehat{\boldsymbol{\Sigma}}_{\text{res}} + \widehat{\boldsymbol{\Sigma}}_{\text{res}}^{1/2}\,\text{diag}(0,\dots,0,\widehat{\lambda}_{q+1},\dots,\widehat{\lambda}_p)\widehat{\boldsymbol{\Sigma}}_{\text{res}}^{1/2}.$$

The MLE of $\mathcal{S}_{\text{PIR}} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\,\text{span}(\boldsymbol{\Gamma}) \subseteq \mathcal{S}_{Y|\boldsymbol{X}}$ under (2.10) is then

$$\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}^{-1}\,\text{span}(\widehat{\boldsymbol{\Gamma}}) = \widehat{\boldsymbol{\Sigma}}_{\text{MLE}}^{-1/2}\,\text{span}_q(\widehat{\boldsymbol{\Sigma}}_{\text{MLE}}^{-1/2}\widehat{\boldsymbol{\Sigma}}_{\text{fit}}\widehat{\boldsymbol{\Sigma}}_{\text{MLE}}^{-1/2})$$

where span_q denotes the span of the first q eigenvectors of its argument.

Remark 6. If $s = q$ the MLE covariance matrix $\widehat{\boldsymbol{\Sigma}}_{\text{MLE}} = \widehat{\boldsymbol{\Sigma}}_{\text{res}}$.

2.3.5 Minimum Average Variance Estimation (MAVE)

Introduced in Xia et al. [XTL+02], the *Minimum Average Variance Estimation* (MAVE) targets the central mean subspace $\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$. It operates under model (2.7) so that it is the first *forward regression* based SDR method. It requires neither the linearity nor the constant variance conditions on the marginal distribution of $\boldsymbol{X}$, but the marginal fourth moments to exists, that is $\mathbb{E}\,|Y|^4 < \infty$ and $\mathbb{E}\,\|\boldsymbol{X}\|^4 < \infty$. Also, g in (2.7) as well as both $\mathbb{E}[\boldsymbol{X} \mid Y]$ and $\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T \mid Y]$ are assumed to have bounded, continuous third derivatives. MAVE uses local linear regression to approximate g based on the following theorem shown by [XTL+02].

Theorem 1. *Assume model* (2.7)*, then*

$$\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]} = \text{span}\Big(\underset{\widetilde{\boldsymbol{B}}\in\text{St}^{p\times q}}{\arg\min}\,\mathbb{E}(Y - \mathbb{E}[Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X}])^2\Big). \tag{2.11}$$

Proof. The mean regression model (2.7) states $Y = g(\boldsymbol{B}^T\boldsymbol{X}) + \epsilon$ with $\text{span}(\boldsymbol{B}) = \mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$. Therefore,

$$\begin{aligned}
&\mathbb{E}(Y - \mathbb{E}[Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X}])^2 \\
&\quad= \mathbb{E}(g(\boldsymbol{B}^T\boldsymbol{X}) - \mathbb{E}(Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X}))^2 + 2\,\mathbb{E}([g(\boldsymbol{B}^T\boldsymbol{X}) - \mathbb{E}(Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X})]\epsilon) + \text{Var}(\epsilon) \\
&\quad= \mathbb{E}(g(\boldsymbol{B}^T\boldsymbol{X}) - \mathbb{E}(Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X}))^2 + \text{Var}(\epsilon) \geq \text{Var}(\epsilon)
\end{aligned} \tag{2.12}$$

where the middle term is zero because $\mathbb{E}(\epsilon \mid \boldsymbol{X}) = 0$ which yields

$$\mathbb{E}([g(\boldsymbol{B}^T\boldsymbol{X}) - \mathbb{E}(Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X})]\epsilon) = \mathbb{E}([g(\boldsymbol{B}^T\boldsymbol{X}) - \mathbb{E}(Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X})]\,\mathbb{E}(\epsilon \mid \boldsymbol{X})) = 0.$$

If $\widetilde{\boldsymbol{B}}$ is such that $\text{span}(\widetilde{\boldsymbol{B}}) = \text{span}(\boldsymbol{B})$, then (2.12) gives with $\mathbb{E}(Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X})] = \mathbb{E}(Y \mid \boldsymbol{B}^T\boldsymbol{X})] = g(\boldsymbol{B}^T\boldsymbol{X})$ that

$$\mathbb{E}(Y - \mathbb{E}[Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X}])^2 = \text{Var}(\epsilon).$$

On the other hand, if $\text{span}(\widetilde{\boldsymbol{B}}) \neq \text{span}(\boldsymbol{B})$, then $\mathbb{E}(Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X}) \neq g(\boldsymbol{B}^T\boldsymbol{X})$, and therefore

$$\mathbb{E}(Y - \mathbb{E}[Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X}])^2 = \mathbb{E}(g(\boldsymbol{B}^T\boldsymbol{X}) - \mathbb{E}(Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X}))^2 + \text{Var}(\epsilon) > \text{Var}(\epsilon).$$

This means that only matrices $\widetilde{\boldsymbol{B}}$ such that $\text{span}(\widetilde{\boldsymbol{B}}) = \mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$ solve the minimization problem in (2.11). □

Using the tower property, we rewrite the objective in Theorem 1 as

$$\mathbb{E}(Y - \mathbb{E}[Y \mid \boldsymbol{B}^T\boldsymbol{X}])^2 = \mathbb{E}[\mathbb{E}\{(Y - \mathbb{E}[Y \mid \boldsymbol{B}^T\boldsymbol{X}])^2 \mid \boldsymbol{B}^T\boldsymbol{X}\}] = \mathbb{E}[\sigma^2(\boldsymbol{B}^T\boldsymbol{X})]$$

where the average variance of the regression residuals $\mathbb{E}\,\sigma^2(\boldsymbol{B}^T\boldsymbol{X})$ gives the method its name.

MAVE estimation is carried out as follows. Suppose $(Y_i, \boldsymbol{X}_i)$ is a sample of n i.i.d. observations. We start with a local linear expansion

$$\mathbb{E}[Y \mid \boldsymbol{B}^T\boldsymbol{X}_i] \approx a_i(\boldsymbol{x}_0) + \boldsymbol{s}_i(\boldsymbol{x}_0)^T\boldsymbol{B}^T(\boldsymbol{X}_i - \boldsymbol{x}_0)$$

at a point $\boldsymbol{x}_0$. Since g in (2.7) is differentiable, the intercept is given by $a(\boldsymbol{x}_0) = g(\boldsymbol{B}^T\boldsymbol{X}_0)$ and the slope vector by $\boldsymbol{s}(\boldsymbol{x}_0) = \nabla g(\boldsymbol{B}^T\boldsymbol{X}_0)$. Because the function g is unknown, we need to approximate its value and slope at $\boldsymbol{x}_0$. [XTL+02] use *local linear regression smoothing* to approximate the residual variance at $\boldsymbol{x}_0$ by

$$\sigma^2(\boldsymbol{B}^T\boldsymbol{X}_i) = \sum_{i=1}^{n}(Y_i - \mathbb{E}[Y \mid \boldsymbol{B}^T\boldsymbol{X}_i])^2 \approx \sum_{i=1}^{n}(a_i(\boldsymbol{x}_0) + \boldsymbol{s}_i(\boldsymbol{x}_0)^T\boldsymbol{B}^T(\boldsymbol{X}_i - \boldsymbol{x}_0))^2 w_i(\boldsymbol{B}^T\boldsymbol{x}_0)$$

with $w_i(\boldsymbol{B}^T\boldsymbol{x}_0) \geq 0$ being weights that sum to 1, that is $\sum_{i=1}^{n} w_i(\boldsymbol{B}^T\boldsymbol{x}_0) = 1$ at any point $\boldsymbol{x}_0$. A common approach of selecting such weights is based on a symmetric kernel $K_h(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2/h)$ with $k : \mathbb{R}_+ \mapsto \mathbb{R}_+$ continuous and monotonically decreasing. The parameter h is called *bandwidth* and controlls the kernel scaling. The default choice in MAVE is the *Gaussian kernel* $K_h(\tilde{\boldsymbol{x}}) = k(\|\tilde{\boldsymbol{x}}\|_2) = \exp(-\|\tilde{\boldsymbol{x}}\|_2/2)$. The weights need to sum to 1, so that

$$w_i(\boldsymbol{B}^T\boldsymbol{x}_0) = \frac{K_h(\boldsymbol{B}^T\boldsymbol{X}_i, \boldsymbol{B}^T\boldsymbol{x}_0))}{\sum_{j=1}^{n} K_h(\boldsymbol{B}^T\boldsymbol{X}_i, \boldsymbol{B}^T\boldsymbol{x}_0)}.$$

This results in observations $\boldsymbol{X}_i$ close to $\boldsymbol{x}_0$ having a strong influence (big weights) on the local regression. The bigger the distance of an observation $\boldsymbol{X}_i$ to $\boldsymbol{x}_0$, the less its influence, where the decrease in influence is controlled by the bandwidth h.

Finally, MAVE estimates $\boldsymbol{B} \in \mathrm{St}^{p\times q}$ by performing local linear regression at all observations $\boldsymbol{X}_j$. That gives the sample version of the objective function as the sum of residual variance estimates,

$$\begin{aligned}\mathbb{E}(Y - \mathbb{E}[Y \mid \widetilde{\boldsymbol{B}}^T\boldsymbol{X}])^2 &\approx \frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{n}(Y_i - \mathbb{E}[Y \mid \boldsymbol{B}^T\boldsymbol{X}_i])^2 \\ &\approx \frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{n}\big(Y_i - a_i(\boldsymbol{X}_j) - \boldsymbol{s}_i(\boldsymbol{X}_j)^T\boldsymbol{B}^T(\boldsymbol{X}_i - \boldsymbol{X}_j)\big)^2 w_i(\boldsymbol{B}^T\boldsymbol{X}_j).\end{aligned}$$

An algorithmic solution to this objective requires estimating $a_i(\boldsymbol{X}_j) \in \mathbb{R}$, $\boldsymbol{s}_i(\boldsymbol{X}_j) \in \mathbb{R}^q$ for $i, j = 1, \ldots, n$ and $\boldsymbol{B} \in \mathbb{R}^{p\times q}$. Those are $n^2(q+1) + pq$ parameters of whose only the pq of $\boldsymbol{B}$ are of interest. A closed form solution for the complete problem does not exist. Alternatively, fixing either $\boldsymbol{B}$ or all the local linear regression parameters $a_i(\boldsymbol{X}_j) \in \mathbb{R}$, $\boldsymbol{s}_i(\boldsymbol{X}_j) \in \mathbb{R}^q$ gives a closed form solution for minimizing the objective. Then we alternate

between the marginal objectives till convergence of the parameter estimate $\widehat{\boldsymbol{B}}$ subject to $\widehat{\boldsymbol{B}} \in \mathrm{St}^{p\times q}$. The constraint that $\widehat{\boldsymbol{B}}$ be in the Stiefel manifold is simply for algorithmic stability as it resolves the ambiguity of $\boldsymbol{Bb}$ in a local region. The estimated subspace is $\operatorname{span}(\widehat{\boldsymbol{B}})$. See [XTL+02; Xia07; Li18] for more details on the estimation procedure.

The MAVE algorithm requires two user choices. First, we need a starting value for $\widehat{\boldsymbol{B}}$ in the alternating optimization routine. Any mean subspace estimate can be chosen. [XTL+02] came up with a better targeted initial value for $\widehat{\boldsymbol{B}}$ the process of which they called outer product of gradients (OPG) described in detail Section 2.3.6. Second, the bandwidth h needs a value. A good choice based on asymptotic results combined with exhaustive testing is provided in [Xia07] as $h = 2.34n^{-1/(\max(p,3)+6)}$. This particular choice is only valid if the procedure is applied to standardized data due to two reasons. First, it is a single bandwidth parameter for all axes with possibly different scaling. Second, this particular choice is based on marginal variance of 1. In contrast to SIR and SAVE, the predictors are standardized separately, as follows. If $\widehat{\boldsymbol{\mu}} = (\widehat{\mu}_1, \ldots, \widehat{\mu}_p)$ is the sample mean vector of $\boldsymbol{X}$ and $\widehat{\boldsymbol{\sigma}}^2 = (\widehat{\sigma}_1^2, \ldots, \widehat{\sigma}_p^2) = \operatorname{diag}(\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{X}})$ are the diagonal elements of the moment estimate of the variance of $\boldsymbol{X}$,

$$(\widehat{\boldsymbol{Z}}_i)_j = \frac{(\boldsymbol{X}_i)_j - \widehat{\mu}_j}{\widehat{\sigma}_j}, \qquad i = 1, \ldots, n. \tag{2.13}$$

2.3.6 Outer Product of Gradients (OPG)

This method was introduced in Xia et al. [XTL+02] as a component of their main method MAVE described in Section 2.3.5. It provides an initial estimate for the central mean subspace $\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$ and is independent of the specific method MAVE. The *Outer Product of Gradients* (OPG) method is based on Theorem 2 (see [XTL+02, Lemma 1] and [Li18, Thm. 11.1]) which is an observation dating back to [Li91].

Theorem 2. *Let* $\tilde{g}(\boldsymbol{x}) = \mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}]$ *be differentiable and* $\boldsymbol{\Sigma}_\nabla = \mathbb{E}[\nabla\tilde{g}(\boldsymbol{X})\nabla\tilde{g}(\boldsymbol{X})^T]$. *Then,*

$$\operatorname{span}(\boldsymbol{\Sigma}_\nabla) \subseteq \mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}. \tag{2.14}$$

Proof. Model (2.7) implies $\tilde{g}(\boldsymbol{x}) = \mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}] = \mathbb{E}[Y \mid \boldsymbol{B}^T\boldsymbol{x} = \boldsymbol{B}^T\boldsymbol{x}] = g(\boldsymbol{B}^T\boldsymbol{x})$. Differentiating yields

$$\nabla_{\boldsymbol{x}}\tilde{g}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} g(\boldsymbol{B}^T\boldsymbol{x}) = \frac{\partial(\boldsymbol{B}^T\boldsymbol{x})}{\partial\boldsymbol{x}}\nabla_{\boldsymbol{B}^T\boldsymbol{x}} g(\boldsymbol{B}^T\boldsymbol{x}) = \boldsymbol{B}\nabla g(\boldsymbol{B}^T\boldsymbol{x}) \tag{2.15}$$

Therefore,

$$\boldsymbol{\Sigma}_\nabla = \mathbb{E}\left(\nabla_{\boldsymbol{x}} g(\boldsymbol{B}^T\boldsymbol{X})\nabla_{\boldsymbol{x}} g(\boldsymbol{B}^T\boldsymbol{X})^T\right) = \boldsymbol{B}\,\mathbb{E}\left(\nabla g(\boldsymbol{B}^T\boldsymbol{X})\nabla g(\boldsymbol{B}^T\boldsymbol{X})^T\right)\boldsymbol{B}^T$$

which completes the proof. □

To derive a sample level estimation procedure, we need a method to estimate the gradients at the given observation positions. The solution provided in [XTL+02] is *local linear regression* [FG92; Fan93; RW94]. The target objective at the population level is to minimize

$$\mathbb{E}\left[(Y - (a(\boldsymbol{x}_0) + \boldsymbol{s}(\boldsymbol{x}_0)^T(X - \boldsymbol{x}_0)))^2 K\left(\frac{\boldsymbol{X} - \boldsymbol{x}_0}{h}\right)\right].$$

with respect to $a(\boldsymbol{x}_0) \in \mathbb{R}$ and $\boldsymbol{s}(\boldsymbol{x}_0) \in \mathbb{R}^p$. The function K is called *kernel*. The kernel choice of OPG is $K(\boldsymbol{u}) = \exp(-\|\boldsymbol{u}\|^2/2)$ which creates weights in the LLR objective with respect to the standard Gaussian distance to $\boldsymbol{x}_0$. The *bandwidth* $h \in \mathbb{R}_+$ controls how distant observations influence the local regression. The smaller the bandwidth the smaller the influence of distant observations. In OPG, local linear regression provides gradient estimates at $\boldsymbol{x}_0$ for $\tilde{g}$ by the slope vector $\boldsymbol{b}(\boldsymbol{x}_0)$.

The sample level algorithm for n i.i.d. observations $(Y_i, \boldsymbol{X}_i)$ is to first standardize the predictors $\boldsymbol{X}_i$ marginally as in (2.13). This allows to use a single bandwidth h for all components. [Xia07] suggest using $h = 2.34n^{-1/(\max(p,3)+6)}$. Then, fit local linear regression at every sample $\boldsymbol{X}_i$ (replacing $\boldsymbol{x}_0$ with $\boldsymbol{X}_i$) to obtain [XTL+02; Li18]

$$\begin{pmatrix} \widehat{a}_i \\ \widehat{\boldsymbol{s}}_i \end{pmatrix} = \left(\sum_{j=1}^{n} \begin{pmatrix} 1 \\ \boldsymbol{X}_j - \boldsymbol{X}_i \end{pmatrix} \begin{pmatrix} 1 \\ \boldsymbol{X}_j - \boldsymbol{X}_i \end{pmatrix}^T K\left(\frac{\boldsymbol{X}_j - \boldsymbol{X}_i}{h} \right) \right)^{-1} \sum_{j=1}^{n} \begin{pmatrix} 1 \\ \boldsymbol{X}_j - \boldsymbol{X}_i \end{pmatrix} Y_j K\left(\frac{\boldsymbol{X}_j - \boldsymbol{X}_i}{h} \right)$$

This computes a slope vector $\widehat{\boldsymbol{s}}_i$ at every observation. Considering the slope vectors $\widehat{\boldsymbol{s}}_i$ as local gradient approximations, the OPG method estimates the outer product of gradients as

$$\widehat{\boldsymbol{\Sigma}}_{\nabla} = \frac{1}{n} \sum_{i=1}^{n} \widehat{\boldsymbol{s}}_i \widehat{\boldsymbol{s}}_i^T .$$

The OPG estimate of the mean subspace is given by $\operatorname{span}(\widehat{\boldsymbol{B}})$, where $\widehat{\boldsymbol{B}}$ consists of the first q eigenvectors of $\widehat{\boldsymbol{\Sigma}}_{\nabla}$.

2.4 SDR methods for Matrix- or Tensor-Valued Predictors

Tensors are a mathematical tool to represent data of complex structure in Statistics. In this thesis, *tensors* are considered as a generalization of matrices to higher dimensions: A tensor is a multi-dimensional array of numbers. For example, the special case of a second-order tensor is a matrix, while a third-order tensor can be represented as a cube of numbers.

Complex data are collected at different times and/or under several conditions often involving a large number of multi-indexed variables represented as tensor-valued data [KB09]. They occur in large-scale longitudinal studies [e.g. Hof15], in agricultural experiments and chemometrics and spectroscopy [e.g. LR92; Bur95]. Also, in signal and video processing where sensors produce multi-indexed data, e.g. over spatial, frequency, and temporal dimensions [e.g. DC07; KR05], in telecommunications [dFM07]. Examples of multiway data include 3D images of the brain, where the modes are the 3 spatial dimensions, and spatio-temporal weather imaging data, a set of image sequences represented as 2 spatial modes and 1 temporal mode.

Classic SDR methods (Section 2.3) are poorly equipped to handle such complex data structures. This is because SDR methods are applied to the vectorized data, thus losing information about the matrix- or tensor-structure. This loss of structural information results

in reduced estimation efficiency, and adversely affects interpretability. Moreover, because the dimension of the vectorized data is the product of the matrix- or tensor-axis dimensions, vectorization often leads to a "big p, small n" setting.

To properly handle matrix- or tensor-valued data we need to introduce some concepts multilinear algebra as well as our notation.

2.4.1 Notes on Multilinear Algebra

Let $\mathcal{A} \in \mathbb{R}^{q_1 \times \ldots \times q_r}$ denotes an order[1] r tensor, where $r \in \mathbb{N}$ is the number of modes or axes (dimensions) of $\mathcal{A}$ and $\mathcal{A}_{i_1,\ldots,i_r} \in \mathbb{R}$ is its $(i_1, \ldots, i_r)$th entry. For example, a $p \times q$ matrix $\boldsymbol{B}$ has two modes, the rows and columns. For matrices $\boldsymbol{B}_k \in \mathbb{R}^{p_k \times q_k}$, $k \in [r] = \{1, 2, \ldots, r\}$, the *multi-linear multiplication*, or *Tucker operator* [Kol06], is defined element wise as

$$(\mathcal{A} \times \{\boldsymbol{B}_1, \ldots, \boldsymbol{B}_r\})_{j_1,\ldots,j_r} = \sum_{i_1,\ldots,i_r=1}^{q_1,\ldots,q_r} \mathcal{A}_{i_1,\ldots,i_r} (\boldsymbol{B}_1)_{j_1,i_1} \cdots (\boldsymbol{B}_r)_{j_r,i_r}$$

which results in an order r tensor of dimension $p_1 \times \ldots \times p_k$. The *k-mode product* of the tensor $\mathcal{A}$ with the matrix $\boldsymbol{B}_k$ is

$$\mathcal{A} \times_k \boldsymbol{B}_k = \mathcal{A} \times \{\boldsymbol{I}_{q_1}, \ldots, \boldsymbol{I}_{q_{k-1}}, \boldsymbol{B}_k, \boldsymbol{I}_{q_{k+1}}, \ldots, \boldsymbol{I}_{q_r}\}.$$

The notation $\mathcal{A} \times_{k \in S} \boldsymbol{B}_k$ is short hand for the iterative application of the mode product for all indices in $S \subseteq [r]$. For example $\mathcal{A} \times_{k \in \{2,5\}} \boldsymbol{B}_k = \mathcal{A} \times_2 \boldsymbol{B}_2 \times_5 \boldsymbol{B}_5$. By only allowing S to be a set, this notation is unambiguous because the mode product commutes for different modes; i.e., $\mathcal{A} \times_j \boldsymbol{B}_j \times_k \boldsymbol{B}_k = \mathcal{A} \times_k \boldsymbol{B}_k \times_j \boldsymbol{B}_j$ for $j \neq k$.

Example 2. *Let $\boldsymbol{A}, \boldsymbol{B}_1, \boldsymbol{B}_2$ be matrices (of matching dimensions). In this bilinear setting the relation to the well known matrix-matrix multiplications is*

$$\boldsymbol{A} \times_1 \boldsymbol{B}_1 = \boldsymbol{B}_1 \boldsymbol{A}, \qquad \boldsymbol{A} \times_2 \boldsymbol{B}_2 = \boldsymbol{A}\boldsymbol{B}_2^T, \qquad \boldsymbol{A} \bigtimes_{k=1}^{2} \boldsymbol{B}_k = \boldsymbol{A} \bigtimes_{k \in \{1,2\}} \boldsymbol{B}_k = \boldsymbol{B}_1 \boldsymbol{A} \boldsymbol{B}_2^T.$$

The operator vec maps an array to a vector. Specifically, $\operatorname{vec}(\boldsymbol{B})$ stands for the $pq \times 1$ vector of the $p \times q$ matrix $\boldsymbol{B}$ resulting from stacking the columns of $\boldsymbol{B}$ one after the other. For a tensor $\mathcal{A}$ of order r and dimensions $q_1, \ldots, q_r$, $\operatorname{vec}(\mathcal{A})$ is the $q_1 q_2 \ldots q_r \times 1$ vector with the elements of $\mathcal{A}$ stacked one after the other in the order r then $r-1$, and so on. For example, if $\mathcal{A}$ is a 3-dimensional array, $\operatorname{vec}(\mathcal{A}) = (\operatorname{vec}(\mathcal{A}_{:,:,1})^T, \operatorname{vec}(\mathcal{A}_{:,:,2})^T, \ldots, \operatorname{vec}(\mathcal{A}_{:,:,q_3})^T)^T$. We use the notation $\mathcal{A} \equiv \mathcal{B}$ for objects $\mathcal{A}, \mathcal{B}$ of any shape if and only if $\operatorname{vec}(\mathcal{A}) = \operatorname{vec}(\mathcal{B})$.

The *inner product* between two tensors of the same order and dimensions is

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1,\ldots,i_r} \mathcal{A}_{i_1,\ldots,i_r} \mathcal{B}_{i_1,\ldots,i_r}$$

This leads to the definition of the *Frobenius norm* for tensors, $\|\mathcal{A}\|_F = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$ and is the straightforward extension of the Frobenius norm for matrices and vectors. The *outer*

[1] Also referred to as rank, therefore the variable name r, but this term is *not* used as it leads to confusion with the concept of rank of a matrix.

product between two tensors $\mathcal{A}$ of dimensions $q_1, \ldots, q_r$ and $\mathcal{B}$ of dimensions $p_1, \ldots, p_l$ is a tensor $\mathcal{A} \circ \mathcal{B}$ of order $r + l$ and dimensions $q_1, \ldots, q_r, p_1, \ldots, p_l$, such that

$$\mathcal{A} \circ \mathcal{B} \equiv (\operatorname{vec} \mathcal{A})(\operatorname{vec} \mathcal{B})^T.$$

Let $\mathcal{K} : \mathbb{R}^{q_1,\ldots,q_{2r}} \to \mathbb{R}^{q_1 q_{r+1},\ldots,q_r q_{2r}}$ be defined element wise with indices $1 \leq i_j + 1 \leq q_j q_{r+j}$ for $j = 1, ..., r$ as

$$\mathcal{K}(\mathcal{A})_{i_1+1,\ldots,i_r+1} = \mathcal{A}_{\lfloor i_1/q_{r+1} \rfloor + 1,\ldots,\lfloor i_r/q_{2r} \rfloor + 1,(i_1 \bmod q_{r+1})+1,\ldots,(i_r \bmod q_{2r})+1}$$

where $\lfloor . \rfloor$ is the floor operation and $a \bmod b$ is the integer division remainder of a/b. The mapping $\mathcal{K}$ is a linear operation and maps an order $2r$ tensor to an order r tensor by reshaping and permuting its elements. This operation allows defining a generalization of the *Kronecker product* to tensors, which we define as $\mathcal{A} \otimes \mathcal{B} = \mathcal{K}(\mathcal{A} \circ \mathcal{B})$. Based on the well known relation $\operatorname{vec}(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^T) = (\boldsymbol{B} \otimes \boldsymbol{A}) \operatorname{vec}(\boldsymbol{X})$, it is technically more convenient to consider a reversed variation on that theme. This is the "reverse" operation $\mathcal{B} \otimes \mathcal{A} = \mathcal{K}_r(\mathcal{A} \circ \mathcal{B})$. In Lemma 1 this relation between the outer product and the Kronecker product is generalized to the iterated outer and Kronecker product

$$\bigcirc_{k=1}^{r} \boldsymbol{A}_k = \boldsymbol{A}_1 \circ \ldots \circ \boldsymbol{A}_r, \qquad \bigotimes_{k=1}^{r} \boldsymbol{A}_k = \boldsymbol{A}_1 \otimes \ldots \otimes \boldsymbol{A}_r.$$

Note the reverse order of the Kronecker product in (2.16).

Lemma 1. *Given $r \geq 2$ matrices $\boldsymbol{A}_k$ of dimension $p_j \times q_j$ for $k = 1, \ldots, r$, then there exists a unique permutation matrix $\boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}$ such that*

$$\operatorname{vec} \bigotimes_{k=r}^{1} \boldsymbol{A}_k = \boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}} \operatorname{vec} \bigcirc_{k=1}^{r} \boldsymbol{A}_k. \tag{2.16}$$

The permutation $\boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}$ with indices $\boldsymbol{p} = (p_1, \ldots, p_r)$ and $\boldsymbol{q} = (q_1, \ldots, q_r)$ is defined recursively as

$$\boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}} = \boldsymbol{S}_{\left(\prod_{k=1}^{r-1} p_k, p_r\right), \left(\prod_{k=1}^{r-1} q_k, q_r\right)} \left(\boldsymbol{I}_{p_r q_r} \otimes \boldsymbol{S}_{(p_1,\ldots,p_{r-1}),(q_1,\ldots,q_{r-1})}\right) \tag{2.17}$$

with initial value

$$\boldsymbol{S}_{(p_1,p_2),(q_1,q_2)} = \boldsymbol{I}_{q_2} \otimes \boldsymbol{K}_{q_1,p_2} \otimes \boldsymbol{I}_{p_1}$$

where $\boldsymbol{K}_{p,q}$ is the commutation matrix *from Abadir and Magnus [AM05, Ch. 11], that is the permutation such that $\operatorname{vec} \boldsymbol{A}^T = \boldsymbol{K}_{p,q} \operatorname{vec} \boldsymbol{A}$ for every $p \times q$ dimensional matrix $\boldsymbol{A}$.*

Proof. Magnus and Neudecker [MN86, Lemma 7] states that

$$\begin{aligned} \operatorname{vec}(\boldsymbol{A}_2 \otimes \boldsymbol{A}_1) &= (\boldsymbol{I}_{q_2} \otimes \boldsymbol{K}_{q_1,p_2} \otimes \boldsymbol{I}_{p_1})(\operatorname{vec} \boldsymbol{A}_2 \otimes \operatorname{vec} \boldsymbol{A}_1) \\ &= (\boldsymbol{I}_{q_2} \otimes \boldsymbol{K}_{q_1,p_2} \otimes \boldsymbol{I}_{p_1}) \operatorname{vec}(\boldsymbol{A}_1 \circ \boldsymbol{A}_2). \end{aligned} \tag{2.18}$$

This proves the statement for $r = 2$. The general statement for $r > 2$ follows via induction. Assuming (2.16) holds for $r - 1$, the induction step is then;

$$\begin{aligned}
\operatorname{vec} \bigotimes_{k=r}^{1} \boldsymbol{A}_k &= \operatorname{vec}\Big(\boldsymbol{A}_r \otimes \bigotimes_{k=r-1}^{1} \boldsymbol{A}_k\Big) \\
&\overset{(2.18)}{=} \Big(\boldsymbol{I}_{q_r} \otimes \boldsymbol{K}_{\prod_{k=1}^{r-1} q_k, p_r} \otimes \boldsymbol{I}_{\prod_{k=1}^{r-1} p_k}\Big) \operatorname{vec}\Big((\operatorname{vec} \boldsymbol{A}_r) \otimes \operatorname{vec} \bigotimes_{k=r-1}^{1} \boldsymbol{A}_k\Big) \\
&= \boldsymbol{S}_{\left(\prod_{k=1}^{r-1} p_k, p_r\right), \left(\prod_{k=1}^{r-1} q_k, q_r\right)} \operatorname{vec}\Big[\Big(\operatorname{vec} \bigotimes_{k=r-1}^{1} \boldsymbol{A}_k\Big)(\operatorname{vec} \boldsymbol{A}_r)^T\Big] \\
&\overset{(2.16)}{=} \boldsymbol{S}_{\left(\prod_{k=1}^{r-1} p_k, p_r\right), \left(\prod_{k=1}^{r-1} q_k, q_r\right)} \operatorname{vec}\Big[\boldsymbol{S}_{(p_1,\ldots,p_{r-1}),(q_1,\ldots,q_{r-1})}\Big(\operatorname{vec} \bigcirc_{k=1}^{r-1} \boldsymbol{A}_k\Big)(\operatorname{vec} \boldsymbol{A}_r)^T\Big] \\
&\overset{(a)}{=} \boldsymbol{S}_{\left(\prod_{k=1}^{r-1} p_k, p_r\right), \left(\prod_{k=1}^{r-1} q_k, q_r\right)} \left(\boldsymbol{I}_{p_r q_r} \otimes \boldsymbol{S}_{(p_1,\ldots,p_{r-1}),(q_1,\ldots,q_{r-1})}\right) \operatorname{vec}\Big[\Big(\operatorname{vec} \bigcirc_{k=1}^{r-1} \boldsymbol{A}_k\Big)(\operatorname{vec} \boldsymbol{A}_r)^T\Big] \\
&= \boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}} \operatorname{vec} \bigcirc_{k=1}^{r} \boldsymbol{A}_k.
\end{aligned}$$

Equality (a) uses the relation $\operatorname{vec}(\boldsymbol{C}\boldsymbol{a}\boldsymbol{b}^T) = (\boldsymbol{I}_{\dim(\boldsymbol{b})} \otimes \boldsymbol{C}) \operatorname{vec}(\boldsymbol{a}\boldsymbol{b}^T)$ for a matrix $\boldsymbol{C}$ and vectors $\boldsymbol{a}, \boldsymbol{b}$. □

Remark 7. The permutation matrix $\boldsymbol{K}_{p,q}$ represents a perfect outer p-shuffle of pq elements.

For tensors of order at least 2, the *matricization* (or *flattening* or *unfolding*) is a reshaping of the tensor into a matrix along a particular mode. For a tensor $\mathcal{A}$ of order r and dimensions $q_1, \ldots, q_r$, the k-mode unfolding $\mathcal{A}_{(k)}$ is a $q_k \times \prod_{l=1, l\neq k} q_l$ matrix with elements

$$(\mathcal{A}_{(k)})_{i_k, j} = \mathcal{A}_{i_1,\ldots,i_r} \quad \text{with} \quad j = 1 + \sum_{\substack{l=1 \\ l\neq k}}^{r} (i_l - 1) \prod_{\substack{m=1 \\ m\neq k}}^{l-1} q_m.$$

A generalization of the well known identity $\operatorname{vec}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}) = (\boldsymbol{C}^T \otimes \boldsymbol{A}) \operatorname{vec} \boldsymbol{B}$ is given by

$$\operatorname{vec}\Big(\mathcal{A} \bigtimes_{k=1}^{r} \boldsymbol{B}_k\Big) = \Big(\bigotimes_{k=r}^{1} \boldsymbol{B}_k\Big) \operatorname{vec} \mathcal{A}.$$

Furthermore, we have

$$(\mathcal{A} \otimes \mathcal{B}) \bigtimes_{k=1}^{r} (\operatorname{vec} \boldsymbol{C}_k)^T = \Big\langle \mathcal{A} \bigtimes_{k=1}^{r} \boldsymbol{C}_k, \mathcal{B} \Big\rangle = \Big\langle \mathcal{A}, \mathcal{B} \bigtimes_{k=1}^{r} \boldsymbol{C}_k^T \Big\rangle = (\operatorname{vec} \mathcal{B})^T \Big(\bigotimes_{k=r}^{1} \boldsymbol{C}_k\Big) \operatorname{vec} \mathcal{A}$$

as well as for any tensor $\mathcal{A}$ of even order $2r$ and matching square matrices $\boldsymbol{B}_k$ holds

$$\mathcal{K}(\mathcal{A}) \bigtimes_{k=1}^{r} (\operatorname{vec} \boldsymbol{B}_k)^T = (\operatorname{vec} \mathcal{A})^T \operatorname{vec}\Big(\bigotimes_{k=r}^{1} \boldsymbol{B}_k^T\Big)$$

The gradient of a function $\mathcal{F}(\mathcal{X})$ of any shape, univariate, multivariate or tensor valued, with argument $\mathcal{X}$ of any shape is defined as the $p \times q$ matrix

$$\nabla_{\mathcal{X}} \mathcal{F} = \frac{\partial (\operatorname{vec} \mathcal{F}(\mathcal{X}))^T}{\partial (\operatorname{vec} \mathcal{X})},$$

where the vectorized quantities $\operatorname{vec} \mathcal{X} \in \mathbb{R}^p$ and $\operatorname{vec} \mathcal{F}(\mathcal{X}) \in \mathbb{R}^q$. This is consistent with the gradient of a real-valued function $f(\boldsymbol{x})$ where $\boldsymbol{x} \in \mathbb{R}^p$ as $\nabla_{\boldsymbol{x}} f \in \mathbb{R}^{p \times 1}$ [Har97, ch. 15].

2.4.2 Multilinear Algebra in R

The vectorization $\operatorname{vec}(\mathcal{A})$ is simply `as.vector(A)` (or `c(A)`). Matricization can be implemented using *axis permutation* `aperm` followed by a change of dimensions. An implementation of $\mathcal{A}_{(k)}$ for an multidimensional array `A` is given by;

```
mat <- function(A, k) {
    matrix(aperm(A, c(k, seq_along(dim(A))[-k])), prod(dim(A)[k]))
}
```

Using the relation $(\mathcal{A} \times_k \boldsymbol{B})_{(k)} = \boldsymbol{B}\mathcal{A}_{(k)}$ alows for a simple implementation of the k-mode product of a tensor $\mathcal{A}$ and a matrix $\boldsymbol{B}$. We only need to "undo" the matricization after the matrix matix product $\boldsymbol{B}\mathcal{A}_{(k)}$.

```
mode.product <- function(A, B, k) {
    C <- B %*% mat(A, k)
    dim(C) <- c(nrow(B), dim(A)[-k])
    aperm(C, order(c(k, seq_along(dim(A))[-k])))
}
```

Given an implementation of the mode product $\mathcal{A} \times_k \boldsymbol{B}$ it is a simple loop to implement the multi-linear multiplication $\mathcal{A} \times_{k \in M} \boldsymbol{B}_k$ for M being the set of `modes` for the iterated mode product between the multidimensional array `A` and a list of matrices `Bs`.

```
mlm <- function(A, Bs, modes = seq_along(Bs)) {
    for (k in seq_along(modes)) {
        A <- mode.product(A, Bs[[k]], modes[k])
    }
    A
}
```

The outer products $\mathcal{A} \circ \mathcal{B}$ is exactly what the base `R` function `outer` computes and `kronecker` computes the generalized Kronecker product $\mathcal{A} \otimes \mathcal{B}$ as described above (`R`'s implementation of `kronecker` is also based on the relation $\mathcal{A} \otimes \mathcal{B} = \mathcal{K}(\mathcal{A} \circ \mathcal{B})$.) An (even more general) implementation of $\mathcal{K}$ (that is for `ncomp =2`) is the following.

```
kronperm <- function(A, dims = dim(A), ncomp = 2, revers = FALSE) {
    # force ‘A‘ to have a multiple of ‘ncomp‘ dimensions
    dim(A) <- c(dims, rep(1L, length(dims) %% ncomp))
    # Shuffle ‘A‘’s axis
    perm <- matrix(seq_along(dim(A)), ncol = ncomp)
    perm <- t(if (revers) perm else perm[, ncomp:1])
    K <- aperm(A, as.vector(perm), resize = FALSE)
    # collapse consecutive dimensions
    dim(K) <- apply(matrix(dim(K), ncol = ncomp), 1, prod)
```

```
    K
}
```

The `kronperm` function allows for some tricks. Specifically, it can compute the perumtation of the *comutation matrix* $\boldsymbol{K}_{p,q}$ from [MN99]. At the same time it provides the permutation

```
shuffle <- function(...) as.vector(t(cbind(...)))
kronperm(seq_len(prod(p, q)), shuffle(p, q), ncomp = length(p), TRUE)
```

correponsing to the permutation matrix $\boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}$ from Lemma 1.

Remark 8. The functions provided are simplified versions of the implementations used for the simulations in Sections 5.5.1 and 5.5.2, as well as the data examples in Sections 5.6.1 and 5.6.2. While the simplified code performs adequately in terms of performance, there is certainly room for improvement.

2.4.3 Central Dimension-Folding Subspace

In the context of (linear) SDR, the first SDR approach [LKA10] performs a bi-linear or multi-linear reduction of the matrix- or tensor-valued predictors, respectively. To clarify this, consider the matrix-valued predictor case. Let $(Y, \boldsymbol{X})$ be jointly distributed, Y is univariate and the predictors $\boldsymbol{X}$ are $p_1 \times p_2$ random matrices. Assume there are matrices $\boldsymbol{B}_j \in \mathbb{R}^{p_j \times q_j}$ where $q_j \leq p_j$ for $i = 1, 2$ such that

$$Y \perp\!\!\!\perp \boldsymbol{X} \mid \boldsymbol{B}_1^T \boldsymbol{X} \boldsymbol{B}_2. \tag{2.19}$$

By the well known relation $\operatorname{vec}(\boldsymbol{AXC}) = (\boldsymbol{C}^T \otimes \boldsymbol{A}) \operatorname{vec} \boldsymbol{X}$, statement (2.19) is equivalent to

$$Y \perp\!\!\!\perp \boldsymbol{X} \mid (\boldsymbol{B}_2 \otimes \boldsymbol{B}_1)^T \operatorname{vec} \boldsymbol{X}$$

which makes $(\boldsymbol{B}_2 \otimes \boldsymbol{B}_1)^T \operatorname{vec} \boldsymbol{X}$ a sufficient reduction of $\boldsymbol{X}$ for the regression on Y by Definition 3. Those two statements are identical. For example, suppose $\boldsymbol{X}$ contains *longitudinal data*, that is $\boldsymbol{X} = (\boldsymbol{X}_{,1}, \ldots, \boldsymbol{X}_{,p_2})$ where $\boldsymbol{X}_{,t}$ are measurements of the same features over multiple time points $t = 1, \ldots, p_2$. The interpretation of $\boldsymbol{X}$ is then different for the rows which correspond to different features and columns indexing time. The bi-linear reduction $\boldsymbol{B}_1^T \boldsymbol{X} \boldsymbol{B}_2$ keeps this relation of rows to features and columns to time. This interpretation is lost by vectorizing without the Kronecker structure. This is where the bi- or multi-linear reductions gain estimation efficiency through the Kronecker constraint while preserving interpretability, opposed to an unconstraint vectorized reduction $\boldsymbol{B} \operatorname{vec} \boldsymbol{X}$.

In the spirit of Definitions 2 and 4, we provide an equivalent definition of the generalization of [LKA10, Def. 1] as discussed in the same paper [LKA10, Sec. 6].

Definition 7 (Dimension-Folding Subspace)**.** Let $\mathcal{X}$ be a tensor-valued random variable of dimension $p_1 \times \ldots \times p_r$ and $\boldsymbol{B}_k \in \mathbb{R}^{p_k \times q_k}$ with $p_k \leq q_k$ for $k = 1, \ldots, r$. If

$$Y \perp\!\!\!\perp \boldsymbol{X} \mid \boldsymbol{X} \bigtimes_{k=1}^{r} \boldsymbol{B}_k$$

then $\mathcal{S} = \operatorname{span}(\boldsymbol{B}_r) \otimes \ldots \otimes \operatorname{span}(\boldsymbol{B}_1)$ is a *dimension-folding subspace* for the regression or classification problem of Y on $\mathcal{X}$.

Again, in analogy to the central (mean) subspace Definitions 3 and 5, we are usually interested in the smallest dimension-folding subspace, if it exists.

Definition 8 (Central Dimension-Folding Subspace)**.** The *central dimension-folding subspace* $\mathcal{S}_{Y|\mathcal{X}^{\otimes}}$ for the regression or classification problem $Y \mid \mathcal{X}$ is defined to be the intersection of all dimension-folding subspaces,

$$\mathcal{S}_{Y|\mathcal{X}^{\otimes}} = \bigcap \{\mathcal{S} \subseteq \mathbb{R}^{p_r} \otimes \ldots \otimes \mathbb{R}^{p_1} : \mathcal{S} \text{ is a dimension-folding subspace for } Y \mid \mathcal{X}\}$$

if and only if it is itself a dimension-folding subspace.

Every dimension-folding subspace is also a dimension reduction subspace[2] [LKA10], since $\operatorname{vec}\left(\mathcal{X} \times_{k=1}^{r} \boldsymbol{B}_k\right) = \left(\bigotimes_{k=r}^{1} \boldsymbol{B}_k\right) \operatorname{vec} \mathcal{X}$. The other direction is *not* the case, in general. This is well illustrated by a counter-example. Let $\boldsymbol{X}$ be a 3×3 random matrix with standard normal entries $X_{ij} \sim \mathcal{N}(0,1)$, and $Y = X_{1,1} + X_{2,1}^2 + X_{1,2}^2 + \epsilon$ for $\epsilon \perp\!\!\!\perp \boldsymbol{X}$, $\epsilon \sim \mathcal{N}(0,1)$. The central subspace $\mathcal{S}_{Y|\boldsymbol{X}} = \operatorname{span}(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_4) \subset \mathbb{R}^9$ is of dimension 3, but the central dimension-folding subspace $\mathcal{S}_{Y|\boldsymbol{X}^{\otimes}} = \operatorname{span}(\boldsymbol{e}_1, \boldsymbol{e}_2) \otimes \operatorname{span}(\boldsymbol{e}_1, \boldsymbol{e}_2) = \operatorname{span}(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_4, \boldsymbol{e}_5) \subset \mathbb{R}^9$ is 4 dimensional. In summary, if they exist,

$$\mathcal{S}_{Y|\operatorname{vec}\mathcal{X}} \subseteq \mathcal{S}_{Y|\mathcal{X}^{\otimes}}$$

is always the case, but equality is *not* guaranteed.

Similar to the central (mean) subspace relations, the relation [LKA10; DC15]

$$\mathcal{S}_{Y|\mathcal{X}^{\otimes}} = \left(\bigotimes_{k=r}^{1} \boldsymbol{A}_k^T\right) \mathcal{S}_{Y|\mathcal{Z}^{\otimes}} \tag{2.20}$$

where $\mathcal{Z} = (\mathcal{X} - \mu) \times_{k=1}^{r} \boldsymbol{A}_k$ holds for invertible matrices $\boldsymbol{A}_k$ and arbitrary offset $\boldsymbol{\mu}$.

The first formal SDR methods for r-mode tensor-valued data where folded-SIR, folded-SAVE and folded-DR, introduced in [LKA10]. They directly generalized and adapted the classic SDR methods SIR, SAVE, and DR to accommodate tensor predictor structures. Their approach is applicable to inverse regression methods for regressions with the marginal predictor distribution satisfying the tensor versions of the linearity Condition 1 and/or constant variance Condition 2, depending on the needs of the underlying conventional method (see [LKA10] for details). A major drawback is that their approach requires the vectorized variance-covariance structure $\boldsymbol{\Sigma} = \operatorname{Var}(\operatorname{vec} \boldsymbol{X})$. This leads to high computational costs while demanding more observations to provide a reasonable estimate of $\boldsymbol{\Sigma}$. We concentrate on later methods that circumvent this drawback, described next.

2.4.4 Longitudinal Sliced Inverse Regression (LSIR)

Aimed specifically at longitudinal data, *Longitudinal Sliced Inverse Regression* (LSIR) [PFB12] reduces matrix-valued predictors $\boldsymbol{X}$ given a univariate response Y. The matrix structure comes from arranging multi-variate features, measured at different time points, in a matrix structure. One axis of the matrix indexes features, the other time points.

[2] Identified, in our notation, $\mathbb{R}^{p_r} \otimes \ldots \otimes \mathbb{R}^{p_1}$ with $\mathbb{R}^{p}$ for $p = \prod_{k=1}^{r} p_k$ and setting $\boldsymbol{X} = \operatorname{vec} \mathcal{X}$. We will continue with similar identifications, if the meaning is clear, without any further mention.

We drop the specific interpretation of the data as longitudinal and simply consider matrix-valued predictors $\boldsymbol{X} \in \mathbb{R}^{p_1 \times p_2}$. This allows to extend the method to arbitrary r-mode tensor $\mathcal{X}$, in a straightforward manner.

[PFB12] requires the linearity Condition 1. Assuming additionally a Kronecker structure of $\boldsymbol{B} = \boldsymbol{B}_2 \otimes \boldsymbol{B}_1$ and $\operatorname{Var}(\operatorname{vec} \boldsymbol{X}) = \boldsymbol{\Sigma} = \boldsymbol{\Sigma}_2 \otimes \boldsymbol{\Sigma}_1$, they show that the SIR subspace has also a Kronecker structure. In other words, under those assumptions, $\mathcal{S}_{\text{SIR}} = \mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]} = \mathcal{S}_{Y|\boldsymbol{X}^{\otimes}} \subseteq \mathcal{S}_{Y|\boldsymbol{X}}$.

The sample level LSIR estimate for n i.i.d. observations $(Y_i, \boldsymbol{X}_i)$ where Y_i is univariate and $\boldsymbol{X}_i$ are matrix valued of dimension $p_1 \times p_2$ is as follows. The first step is to perform a mode-wise standardization. This requires computing the mode-wise covariance estimates

$$\widehat{\boldsymbol{\Sigma}}_1 = \frac{1}{np_2} \sum_{i=1}^{n} (\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}})(\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}})^T, \qquad \widehat{\boldsymbol{\Sigma}}_2 = \frac{1}{np_1} \sum_{i=1}^{n} (\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}})^T (\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}})$$

where $\widehat{\boldsymbol{\mu}} = n^{-1} \sum_{i=1}^{n} \boldsymbol{X}_i$ is the sample mean of the $\boldsymbol{X}_i$'s. With those we standardize the observations as

$$\widehat{\boldsymbol{Z}}_i = \widehat{\boldsymbol{\Sigma}}_1^{-1/2} (\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}}) \widehat{\boldsymbol{\Sigma}}_2^{-1/2}.$$

As in SIR, the responses are discretized by slicing (see Section 2.3.1). Let $\tilde{Y}_i = j$ if Y_i is in the j'th of $j = 1, \ldots, s$ slices and denote with n_j the number of observations in the j'th slice. For every slice we compute its (within slice) mean

$$\widehat{\boldsymbol{\mu}}_j = \frac{1}{n_j} \sum_{\substack{i=1 \\ \tilde{Y}_i = j}}^{n} \widehat{\boldsymbol{Z}}_i$$

used to compute the (in-between slices) mode-wise covariances

$$\widehat{\boldsymbol{\Omega}}_1 = \frac{1}{sp_2} \sum_{j=1}^{s} \widehat{\boldsymbol{\mu}}_j \widehat{\boldsymbol{\mu}}_j^T, \qquad \widehat{\boldsymbol{\Omega}}_2 = \frac{1}{sp_1} \sum_{j=1}^{s} \widehat{\boldsymbol{\mu}}_j^T \widehat{\boldsymbol{\mu}}_j.$$

Letting $\widehat{\boldsymbol{U}}_1$, $\widehat{\boldsymbol{U}}_2$ denote the $p_1 \times q_1$, $p_2 \times q_2$ dimensional matrices consisting of the first q_1, q_2 eigenvectors of $\widehat{\boldsymbol{\Omega}}_1$, $\widehat{\boldsymbol{\Omega}}_2$, respectively. It remains to reverse the standardization to get the LSIR estimate

$$\widehat{\boldsymbol{B}} = \widehat{\boldsymbol{B}}_2 \otimes \widehat{\boldsymbol{B}}_1 = \widehat{\boldsymbol{\Sigma}}_2^{-1/2} \widehat{\boldsymbol{U}}_2 \otimes \widehat{\boldsymbol{\Sigma}}_1^{-1/2} \widehat{\boldsymbol{U}}_1$$

for $\operatorname{span}(\boldsymbol{B}) \subseteq \mathcal{S}_{Y|\boldsymbol{X}^{\otimes}}$.

Generalizing this procedure to r-mode tensors $\mathcal{X}$ is straightforward. One only needs to matricize appropriately and use multi-linear instead of bilinear operations. Letting $p = \prod_{k=1}^{r} p_k$, the mode-wise covariances $\widehat{\boldsymbol{\Sigma}}_k$ for $k = 1, \ldots, r$ have the form

$$\widehat{\boldsymbol{\Sigma}}_k = \frac{p_k}{np} \sum_{i=1}^{n} (\mathcal{X}_i - \widehat{\boldsymbol{\mu}})_{(k)} (\mathcal{X}_i - \widehat{\boldsymbol{\mu}})_{(k)}^T.$$

The bilinear standardization is replaced by its multi-linear analog

$$\widehat{\mathcal{Z}}_i = (\mathcal{X}_i - \widehat{\boldsymbol{\mu}}) \bigtimes_{k=1}^{r} \widehat{\boldsymbol{\Sigma}}_k^{-1/2}$$

which are used to compute the (within slice) means $\widehat{\boldsymbol{\mu}}_j$, for $j = 1, \ldots, s$, computed as in the matrix case with the difference they are now r-tensors instead of matrices. The (in-between slice) mode-wise covariances are given by

$$\widehat{\boldsymbol{\Omega}}_k = \frac{p_k}{sp} \sum_{j=1}^{s} (\widehat{\boldsymbol{\mu}}_j)_{(k)} (\widehat{\boldsymbol{\mu}}_j)_{(k)}^T$$

Suppose the $p_k \times q_k$ matrices $\widehat{\boldsymbol{U}}_k$ consist of the first q_k eigenvectors of $\widehat{\boldsymbol{\Omega}}_k$, for every mode $k = 1, \ldots, r$. Then, the generalized LSIR estimate is

$$\widehat{\boldsymbol{B}} = \bigotimes_{k=r}^{1} \widehat{\boldsymbol{\Sigma}}_k^{-1/2} \widehat{\boldsymbol{U}}_k$$

for estimating $\operatorname{span}(\boldsymbol{B}) \subseteq \mathcal{S}_{Y|\mathcal{X}^\otimes}$.

2.4.5 Tensor Sliced Inverse Regression (TSIR)

Another variation of SIR, adapted specifically for r-mode tensor data, is *Tensor Sliced Inverse Regression* (TSIR) from [DC15]. It requires a weaker version of the linearity Condition 1. For $\mathcal{S}_{Y|\mathcal{X}^\otimes} = \operatorname{span}(\boldsymbol{B}_r) \otimes \ldots \otimes \operatorname{span}(\boldsymbol{B}_1)$ with $\boldsymbol{B}_k \in \mathbb{R}^{p_k \times q_k}$ where $q_k \leq p_k$, the assumption is that $\mathbb{E}[\mathcal{X} \mid \mathcal{X} \times_k \boldsymbol{B}_k]$ is a linear function of $\mathcal{X} \times_k \boldsymbol{B}_k$ for all $k = 1, \ldots, r$. Under this assumption, they show that

$$\mathbb{E}[\boldsymbol{X} \mid Y] = \mathbb{E}[\boldsymbol{X} \mid Y] \bigtimes_{k=1}^{r} \boldsymbol{P}_{\boldsymbol{B}_k} (\boldsymbol{\Sigma}_k)^T \tag{2.21}$$

where $\boldsymbol{\Sigma}_k = \mathbb{E}[\mathcal{X}_{(k)} \mathcal{X}_{(k)}^T]$ are the k'th-mode variance-covariance matrices. Letting $\boldsymbol{\Gamma}_k$ be a basis of $\operatorname{span}(\boldsymbol{\Sigma}_k \boldsymbol{B}_k)$, for $k = 1, \ldots, r$, we can reformulate (2.21) as

$$\mathbb{E}[\boldsymbol{X} \mid Y] = \mathbb{E}[\boldsymbol{X} \mid Y] \bigtimes_{k=1}^{r} \boldsymbol{P}_{\boldsymbol{\Gamma}_k}$$

with $\mathcal{S}_{Y|\mathcal{X}^\otimes} = \bigotimes_{k=r}^{1} \operatorname{span}(\boldsymbol{\Sigma}_k^{-1} \boldsymbol{\Gamma}_k)$. Note that $\boldsymbol{P}_{\boldsymbol{\Gamma}_k}$ is symmetric which makes transposing unnecessary. Based on this, TSIR estimates the central dimension-folding subspace $\mathcal{S}_{Y|\mathcal{X}^\otimes}$ by minimizing

$$\left\| \mathbb{E}[\boldsymbol{X} \mid Y] - \mathbb{E}[\boldsymbol{X} \mid Y] \bigtimes_{k=1}^{r} \boldsymbol{P}_{\boldsymbol{\Gamma}_k} \right\|_F^2.$$

Let $(\mathcal{X}_i, Y_i)$ be an i.i.d. sample of n realizations of $(\mathcal{X}, Y)$ for $\mathcal{X}$ being $p_1 \times \ldots \times p_r$ dimensional and Y univariate. The suggested procedure in [DC15] for TSIR proceeds as follows.

Split the range of Y into s disjoint and contiguous slices S_j (Definition 6), and initialize $\widehat{\boldsymbol{\Gamma}}_j$ for $j = 1, \ldots, r$ (for example, using the LSIR estimates). Next, compute the within slice means

$$\widehat{\boldsymbol{\mu}}_j = \frac{1}{n_j} \sum_{i \in S_j} \boldsymbol{X}_i$$

where n_j is the j'th slice size, and perform cyclic updates of $\widehat{\boldsymbol{\Gamma}}_k$ for $k = 1, \ldots, r$ by setting the columns of $\widehat{\boldsymbol{\Gamma}}_k$ to the q_k leading eigenvectors of

$$\widehat{\boldsymbol{\Omega}}_k = \frac{1}{n} \sum_{j=1}^{s} n_j (\widehat{\boldsymbol{\mu}}_j)_{(k)} \Big(\widehat{\boldsymbol{\mu}}_j \bigtimes_{l=1:l\neq k}^{r} \boldsymbol{P}_{\widehat{\boldsymbol{\Gamma}}_l} \Big)_{(k)}^T$$

until the objective function

$$\frac{1}{n} \sum_{j=1}^{s} n_j \Big\| \widehat{\boldsymbol{\mu}}_j - \widehat{\boldsymbol{\mu}}_j \bigtimes_{k=1}^{r} \boldsymbol{P}_{\widehat{\boldsymbol{\Gamma}}_k} \Big\|_F^2$$

converges. The TSIR estimate is then given by $\operatorname{span}(\widehat{\boldsymbol{B}}) = \bigotimes_{k=r}^{1} \operatorname{span}(\widehat{\boldsymbol{\Omega}}_k^{-1} \widehat{\boldsymbol{\Gamma}}_k)$ where $\widehat{\boldsymbol{\Omega}}_k^{-1} = n^{-1} \sum_{i=1}^{n} (\boldsymbol{X}_i)_{(k)} (\boldsymbol{X}_i)_{(k)}^T$.

An interesting follow-up is given in a later section [DC15, Sec. 3.2] of the same paper. Based on the fact that SIR estimate is the MLE for the central subspace if $\operatorname{vec} \mathcal{X} \mid Y$ is multivariate normal, a slightly different method, TSIR-K, is proposed. The K stands for *Kronecker* for the additional assumption of a Kronecker structure of the vectorized covariance. Meaning, $\operatorname{Cov}(\operatorname{vec} X) = \bigotimes_{k=r}^{1} \boldsymbol{\Sigma}_k$ is assumed in addition to the weakened linearity condition. Under this additional assumption, TSIR-K works by first standardizing the observations as

$$\widehat{\mathcal{Z}}_i = (\mathcal{X}_i - \widehat{\boldsymbol{\mu}}) \bigtimes_{k=1}^{r} \widehat{\boldsymbol{\Sigma}}_k^{-1/2}$$

where

$$\widehat{\boldsymbol{\Sigma}}_k = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}})(\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}})^T$$

are (unscaled, scaling is of no concern for this to work) estimates of $\boldsymbol{\Sigma}_k$, for $k = 1, \ldots, r$. Then, apply TSIR to the standardized data $\widehat{\boldsymbol{Z}}_i$ to get an estimate $\widehat{\mathcal{S}}_{Y|\mathcal{Z}^\otimes}$. The subspace estimate for $\mathcal{S}_{Y|\mathcal{X}^\otimes}$ then uses (2.20) to get the estimate $\widehat{\mathcal{S}}_{Y|\mathcal{X}^\otimes} = \Big(\bigotimes_{k=1}^{r} \widehat{\boldsymbol{\Sigma}}_k^{-1/2} \Big) \widehat{\mathcal{S}}_{Y|\mathcal{Z}^\otimes}$. Listing this as a separate method has two reasons.

3 Fusing SDR with Neural Networks

In this chapter, we consider the forward regression model (2.7),

$$Y = g(\boldsymbol{B}^T\boldsymbol{X}) + \epsilon,$$

where Y can be a real valued multivariate or categorical random variable, $\boldsymbol{X}$ is a p-variate random vector, $\boldsymbol{B} \in \mathbb{R}^{p\times q}$ of rank $q < p$, $\epsilon \in \mathbb{R}$ is a random variable with $\mathbb{E}[\epsilon \mid \boldsymbol{X}] = 0$ and $\mathrm{Var}(\epsilon) < \infty$, and g is an unknown continuously differentiable non-constant function. The projection $\boldsymbol{B}^T\boldsymbol{X}$ is a *linear sufficient dimension reduction* since $\mathbb{E}[Y \mid \boldsymbol{X}] = \mathbb{E}[Y \mid \boldsymbol{B}^T\boldsymbol{X}]$. Our goal is to estimate the linear projection.

We present *Neural Network SDR* (NNSDR; [KFB22]) and its sibling method *Neural Network OPG* (NNOPG; [KFB22]). The key idea behind both methods is to model the dependence of Y on $\boldsymbol{X}$ with a *neural network* (NN). This solves a particular drawback of forward regression methods like MAVE, OPG or CVE which are computationally demanding, rendering them infeasible for big data regression or classification problems. Specifically, if the number of features p and the sample size n increase, which is a setting encountered increasingly frequently and where neural networks come into play. The latter can handle gigantic amounts of data while proven to be well equipped to perform almost any predictive regression or classification task. Moreover, this leads to a straightforward way to predict the response at the same time via NNs, something missing from most SDR methods where dimension reduction is separate from model fitting.

In NNOPG, we use the relation (2.15), which states that $\nabla_{\boldsymbol{x}} g(\boldsymbol{B}^T\boldsymbol{x}) = \boldsymbol{B}^T \nabla_{\boldsymbol{z}} g(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{B}^T\boldsymbol{x}}$. The gradients are based on the neural network fitting the conditional expected value $\boldsymbol{x} \mapsto \mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}] = g(\boldsymbol{B}^T\boldsymbol{X})$. The network itself is oblivious of any reduction, it is only trained to be as predictive as possible. On the other hand, NNSDR explicitly models the reduction into the neural network by incorporating the reduction as a part of the network itself. This is realized by fitting $\boldsymbol{x} \mapsto \mathbb{E}[Y \mid \boldsymbol{B}^T\boldsymbol{X} = \boldsymbol{B}^T\boldsymbol{x}] = g(\boldsymbol{B}^T\boldsymbol{x})$ where the reduction $\boldsymbol{B}$ is a part of the network.

While the general approach is applicable to a wide range of different neural network architectures, NNOPG and NNSDR are built upon the simplest of neural networks, introduced next.

3.1 Multi Layer Perceptron (MLP)

Historically, neural networks draw inspiration from the inner workings of the human brain. Analogous to the brain's neural network, (artificial) neural networks consist of interconnected nodes (neurons), linked by weighted connections (synapses). In the brain, the synapses serve as wires for electrical or chemical signals, each possessing a specific strength. The different strengths of synapses controll the electrical flow between neurons. When a neuron, connected to multiple others, receives electrical signals through its inbound synapses,

it does nothing untill an inbound signal thresholds is reached. Then, it fires, which sends a signal into its outbound synapses. This in turn propagates signals though the brain, controlled by neurons acting as signal gates and the synapses regulate the signal through their strength.

Given this oversimplified view of the human brain, a *Multi Layer Perceptron* (MLP; [Gur97; MP43; GBC16; JWH+21]) is the simplest of modern neural networks. It derives its name from the *Rosenblatt perceptron* [Ros58], a variant of the *McCulloch-Pitts neuron* [MP43], which is a linear binary classifier modeling a single neuron. For a modern neural network, the binary output is replaced by a univariate real-valued function called an *activation function*. Putting multiple perceptrons in parallel forms a single *layer*. Chaining *multiple* layers leads to an MLP, a "network of neurons".

Definition 9 (Layer, MLP and Activation Function)**.** A *Multi Layer Perceptron* with L layers is a function $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ with the structure

$$f(\boldsymbol{x}; \boldsymbol{\Theta}) = (f_L \circ f_{L-1} \circ \ldots \circ f_1)(\boldsymbol{x})$$

where $\boldsymbol{\Theta} = (\boldsymbol{W}_1, \boldsymbol{b}_1, \ldots, \boldsymbol{W}_L, \boldsymbol{b}_L)$ are the network parameters. Each *layer* f_l for $l = 1, \ldots, L$ is of the form

$$f_l(\boldsymbol{x}; \boldsymbol{W}_l, \boldsymbol{b}_l) = \phi_l(\boldsymbol{W}_l \boldsymbol{x} + \boldsymbol{b}_l)$$

which is an affine linear transformation with *weights* $\boldsymbol{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$, *bias* $\boldsymbol{b}_l \in \mathbb{R}^{n_l}$ and a *component-wise* applied *activation function* $\phi_l : \mathbb{R} \to \mathbb{R}$ which is non-constant, continuous and almost everywhere differentiable.

The input $\boldsymbol{x}$ is the *input layer*, that can be seen as a placeholder for the inputs features while the last layer f_L is denoted *output layer*. All layers f_l inbetween, that is for $l = 1, \ldots, L-1$, are known as *hidden layers*. It is also common in the machine learning literature (even though not consistently applied) to count the number of hidden layers. This means that a 1 (hidden) layer MLP has two layers $f = f_1 \circ f_2$ in reference to Definition 9. See Figure 3.1 for an example of a 2 layer MLP with $n_0 = 4$ input features $\boldsymbol{x} \in \mathbb{R}^4$, $n_1 = 5$ neurons in the first hidden layer, $n_2 = 4$ neurons in the second hidden layer and univariate output $n_L = n_3 = 1$.

To specify an MLP, one must choose a number of hyperparameters. Those are the number of layers L, with the number of neurons n_l in each hidden layer as well as the activation function ϕ_l, of each layer. Usually, the only non-free hyperparameters are the input and output dimensions n_0 and n_L and the output activation function ϕ_L, as they are inherent to the problem on hand. A common choise for the activation function is the *Rectified Linear Unit* defined as

$$\mathrm{ReLU}(x) = \max(0, x)$$

where it is common to use the ReLU in all hidden layers. From now on we opt for ReLU as the hidden activation function throughout the remainder of this chapter. Meaning $\phi = \phi_l = \mathrm{ReLU}$ for all hidden layers $l = 1, \ldots, L-1$. It has proven itself as the number one choise due to its simplicity while performing extremely well in almost all tasks. Alternative widely-used options involve sigmoid functions such as the hyperbolic tangent (tanh).

In Definition 9 the output can be multivariate, but for the sake of simplicity we focus on univariate output only, even though the methodology is applicable to multi-variate outputs.

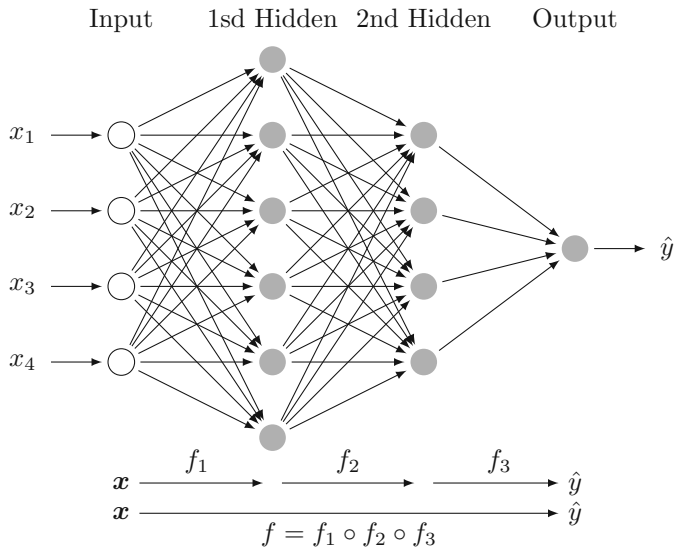

Figure 3.1: Example Architecture of a 2 layer MLP (that is 2 hidden and 1 output layer).

As justification for the following approach we refer to a result from [Hor91] known as the *Universal Approximator Theorem* which established that 1-layer MLPs can approximate continuously differentiable functions on a compact set arbitrarily well.

Theorem 3 ([Hor91, Thm 3]). *Let* $\mathrm{MLP}_\infty(\phi)$ *be the set of all one layer MLPs with arbitrarily many neurons in the first layer and an activation function* ϕ *which is non-constant and bounded, then* $\mathrm{MLP}_\infty(\phi)$ *is uniformly* m *dense in* $C^m(\mathbb{R}^p)$ *on compact sets, where* $C^m(\mathbb{R}^p)$ *is the space of all* m*-times differentiable functions on* $\mathbb{R}^p$*.*

For $m = 1$, applied to the forward regression model (2.7) with $\boldsymbol{x} \mapsto g(\boldsymbol{B}^T\boldsymbol{x}) \in C^1(\mathbb{R}^p)$, we get that for every $\mu > 0$ there exists an $f \in \mathrm{MLP}_\infty(\phi)$ such that

$$\sup_{\boldsymbol{x}\in K} \left(|g(\boldsymbol{B}^T\boldsymbol{x}) - f_{\mathrm{MLP}_2}(\boldsymbol{x};\boldsymbol{\Theta})| + \|\nabla_{\boldsymbol{x}} g(\boldsymbol{B}^T\boldsymbol{x}) - \nabla_{\boldsymbol{x}} f_{\mathrm{MLP}_2}(\boldsymbol{x};\boldsymbol{\Theta})\|\right) \leq \nu.$$

where $K \subset \mathbb{R}^p$ is compact. This means that the conditional expectation $\mathbb{E}(Y \mid \boldsymbol{X} = \boldsymbol{x}) = g(\boldsymbol{B}^T\boldsymbol{x})$ as well as its gradients can be approximated arbitrarily well by an MLP as long as we make it big enough.

Remark 9. We stated before that our default activation function would be the *rectified linear unit* ReLU. The ReLU is *not* bounded and as such Theorem 3 is *not* applicable. We still choose the ReLU function due to experience showing no difference in performance. Moreover, [Hor91, Thm 1] states that $\mathrm{MLP}_\infty(\phi)$ is uniformly dense in $L^k(\mu)$ on every compact set for any finite measure μ in $\mathbb{R}^p$ and requires only bounded activation function ϕ. This is applicable to ReLU by observing that ReLU can be used to construct a bounded activation function as $\phi(x) = \mathrm{ReLU}(x-1) - 2\,\mathrm{ReLU}(x) + \mathrm{ReLU}(x+1)$. This is a linear combination which can be mimicked by a ReLU MLP by combining three neurons to mimic a single neuron with ϕ as activation function. As such MLPs with ϕ as activation function are a subset of $\mathrm{MLP}_\infty(\phi) \subset \mathrm{MLP}_\infty(\mathrm{ReLU})$, which makes it dense in $L^k(\mu)$.

3.1.1 A simple Neural Network Framework in R

Here we implement a simple neural network framework in base R from scatch. Using this simple framework we implement NNOPG (see Section 3.2) and NNSDR (see Section 3.3) pinpointing the exact workings of the estimation procedure.

Remark 10. The following implementation is intended for deductive and research purposes. It allows for a comprehensive understanding and manipulation of every aspect. The implementations used in Section 3.5.2 is written in TensorFlow [MAP+15] using the R package `tensorflow` [AT20]. While the TensorFlow framework is user-friendly and very powerful, some minor details may become complex when facing specific research questions or new concepts not yet addressed by the frameworks API. This can result in patchwork code, obscuring the underlying concepts and challenging to understand. Something we encountered during the development of NNSDR.

Before we start implementing the framework we provide a demonstraton of the intended usage of our framework. This demonstration serves as a guide for the implementation process and motivates our design decisions.

```
# Generate some toy data
sample.size <- 101
X <- matrix(rnorm(4 * sample.size), 4)
y <- sin(X[1, ]) + rnorm(sample.size, 0, 0.1)

# Construct the neural network f(x) = f3(f2(f1(x))) from Figure 3.1
network <- NN(
    Layer(c(4, 6), activation = relu),   # f1 : x ↦ ReLU(W1 x + b1) = h1
    Layer(c(6, 4), activation = relu),   # f2 : h1 ↦ ReLU(W2 h1 + b2) = h2
    Layer(c(4, 1))                       # f3 : h2 ↦ W3 h2 + b3 = ŷ
)

# Train the network using RMSprop
history <- rmsprop(network, X, y, loss = mse)

# Plot the training loss over epochs
plot(history, main = "history", xlab = "epochs", ylab = "loss")

# Predict response
y_hat <- network(X)

# Compute gradients ∇x f(x) evaluated at all samples
G <- grad(network)(X)
```

We start with the implementation of a single layer as in Definition 9, that is a function $\boldsymbol{x} \mapsto \phi(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b})$, which is the result of the constructor `Layer`. For convenience we only want to provide the input and output dimensions as well as the activation function and let the constructor initialize the weights for us. But we also want to be able to set the parameters of a layer if needed. In foresight we add two additional parameters, `dropout` and `constraint`. The layer itself simply remembers their values which are used in the routines for training the neural network.

```
Layer <- function(weights, bias = 0, activation = identity,
    dropout = 0, constraint = NULL
```

```
) {
    if (!is.matrix(weights)) {
        # Xavier Uniform Initialization
        bound <- sqrt(6 / sum(weights))
        weights <- matrix(runif(prod(weights), -bound, bound), weights[2])
    }
    # ensures to remembered
    force(bias)
    force(activation)
    force(dropout)
    force(constraint)
    # return function x ↦ φ(Wx + b)
    function(x) activation(weights %*% x + bias)
}
```

Next we define several activation functions commonly used in neural networks and their derivatives. We assigns these derivatives as attributes to their respective functions, allowing to compute gradients of the network using the backpropagation algorithm discussed later.

The `relu` function implements the ReLU activation function, while `logit` implements the logistic function. Other built-in functions such as the identity, exponential function, and hyperbolic tangent are also given derivatives.

```
relu <- function(x) pmax(x, 0)
attr(relu, "deriv") <- function(x) 0 < x

logit <- function(x) 1 / (1 + exp(-x))
attr(logit, "deriv") <- function(x) {
    ex <- exp(x)
    ex / (1 + ex)^2
}

# Add derivatives to builtins
attr(identity, "deriv") <- function(x) 1 + (0 * x)
attr(exp, "deriv") <- exp
attr(tanh, "deriv") <- function(x) cosh(x)^(-2)
attr(abs, "deriv") <- function(x) 1 - 2 * (x < 0)
```

With the derivatives provided by the functions themself, we have a convenient way of differentiation as

```
grad <- function(fun) attr(fun, "deriv")
```

which we can use for any activation function ϕ as `grad(activation)` to get its gradient $\nabla\phi$. Evaluation of the gradient $\nabla\phi(\boldsymbol{x})$ at $\boldsymbol{x}$ is then `grad(activation)(x)`.

These derivative functions are critical for the *backpropagation* algorithm. Backpropagation enables the computation of gradients of a neural network $f(\,.\,;\boldsymbol{\theta})$ with respect to its input $\nabla_{\boldsymbol{x}} f(\boldsymbol{x};\boldsymbol{\theta})$ or its parameters $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta})$. This algorithm is a clever application of the chain rule. It begins with a *forward pass*, where the network is evaluated, storing intermediate values. Then, during the subsequent *backward pass*, the gradient is propagated backward through the network using the chain rule in conjunction with the intermediate values stored in the forward pass, effectively computing the desired gradient.

Given layers f_l with differentiable activation functions, we can construct an MLP as in Definition 9. The differentiability of the activation functions enables the computation of

the gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; \boldsymbol{\theta})$ via backpropagation.

By attaching the gradient to the `"deriv"` attribute of the network, we create a differentiable network function, say `network <-NN(...)`. This network can be evaluated on input data `x` by `network(x)`, and its gradients are obtained using `grad(network)(x)`.

```
NN <- function(...) {
    layers <- unlist(list(...))
    # MLP as concatenation of layers
    network <- function(x) {
        for (layer in layers) x <- layer(x)
        x
    }
    # Network Derivative using Backprop algorithm
    attr(network, "deriv") <- function(x) {
        G <- apply(matrix(x, NROW(x)), 2, function(x_i) {
            # Forward Pass
            z <- list()     # layer output (befor activation)
            for (l in seq_along(layers)) {
                layer <- environment(layers[[l]])
                z[[l]] <- layer$weights %*% x_i + layer$bias
                x_i <- layer$activation(z[[l]])
            }
            # Backward Pass
            G_i <- diag(nrow(x_i))
            for (l in rev(seq_along(layers))) {
                layer <- environment(layers[[l]])
                phi.prime <- as.vector(grad(layer$activation)(z[[l]]))
                G_i <- G_i %*% (phi.prime * layer$weights)
            }
            G_i
        })
        drop(`dim<-`(G, c(NROW(x), nrow(G) / NROW(x), ncol(G))))
    }

    network
}
```

At this point we have implemented all the required functionality except for training a neural network. To develop training routines, we need loss functions that define the training objective. Similar to activation functions, the loss functions need to be differentiable to facilitate gradient based optimization. Therefore, we use the same pattern of attaching the derivatives to the loss itself. The derivative is computed with respect to the second argument of the bivariate loss function, which, by convention is the predicted response in contrast to the observed response expected as the first argument.

The code next defines the *mean squared error* and the *binary cross-entropy* loss. The latter can be used for binary classification tasks expecting 0 or 1 class labels as its first argument and the second argument is in the open $(0, 1)$ interval interpreted as the probability of being a member of the class with label 1.

```
mse <- function(y, y_hat) mean((y - y_hat)^2)
attr(mse, "deriv") <- function(y, y_hat) (2 / length(y)) * (y_hat - y)

binary.cross.entropy <- function(y, y_hat) {
```

```
    -(y * log(y_hat) + (1 - y) * log(1 - y_hat))
}
attr(binary.cross.entropy, "deriv") <- function(y, y_hat) {
    (1 - y) / (1 - y_hat) - y / y_hat
}
```

The loss $\mathcal{L}$ specifies the training objective $\mathcal{L}(y, f(\boldsymbol{x}; \boldsymbol{\theta}))$ to be minimized. For gradient based optimization, we need to compute the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(y, f(\boldsymbol{x}; \boldsymbol{\theta}))$. Similar to computing the gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; \boldsymbol{\theta})$ of a neural network with respect to its input, we implement the gradient of the objective $\nabla_{\boldsymbol{\theta}} \mathcal{L}(y, f(\boldsymbol{x}; \boldsymbol{\theta}))$ with respect to the network parameters via backpropagation.

The additional parameter, `dropout`, determines whether *dropout* should be used for gradient computation. Dropout, introduced in [SHK+14], is a regularization technique commonly used in neural networks to prevent overfitting and improve generalization performance. Dropout randomly "drops out" a subset of neurons from the network, effectively generating a sparse network. This is equivalent to setting the "dropped" neurons output and the gradient elements corresponding to the parameters of that neuron to zero. When applied iteratively, the trained network essentially becomes a moving average over a sequence of sparser networks. We add this functionality directly into the backpropagation algorithm, enabling dropout to be easily applied in different optimization algorithms by setting the `dropout` flag to `TRUE`.

```
backprop <- function(network, x, y, loss = mse, training = FALSE) {
    layers <- environment(network)$layers
    z <- vector("list", length(layers)) # layer output (befor activation)
    h <- vector("list", length(layers)) # layer output (after activation)
    active <- vector("list", length(layers))    # dropout active neurons

    # Forward Pass
    y_hat <- x <- as.matrix(x)
    for (l in seq_along(layers)) {
        layer <- environment(layers[[l]])
        z[[l]] <- layer$weights %*% y_hat + layer$bias
        if (training && layer$dropout) {
            prob <- c(layer$dropout, 1 - layer$dropout)
            active[[l]] <- sample(0:1, nrow(layer$weights), TRUE, prob)
        } else {
            active[[l]] <- 1
        }
        y_hat <- h[[l]] <- active[[l]] * layer$activation(z[[l]])
    }

    # Backward Pass
    gradients <- vector("list", length(layers))
    G <- grad(loss)(y, y_hat)
    for (l in rev(seq_along(layers))) {
        layer <- environment(layers[[l]])
        grad.phi.z <- active[[l]] * grad(layer$activation)(z[[l]])
        h_lm1 <- if (l > 1) h[[l - 1]] else x
        gradients[[l]] <- list(
            weights = tcrossprod(G * grad.phi.z, h_lm1) / ncol(x),
               bias = rowMeans(G * grad.phi.z)
        )
```

```
32          G <- crossprod(layer$weights, G * grad.phi.z)
33      }
34
35      gradients
36  }
```

The gradient provided by `backprop` is a nested list of partial gradients. This approach has the advantage of avoiding the need to unpack parameter subsets, but it also has the disadvantage of not allowing direct use of vectorized operations. Therefore, we implement a recursive version of `Map` called `rMap` which provides a multivariate, flexible and recursive way to apply a function to elements of one or more vectors or lists, with the capability to handle nested lists and apply the same function to each level of nesting. For example `rMap('+',list(1,list(2,3)),list(10,list(20,30)))` computes `list(11,list(22,33))`.

```
1  rMap <- function(fun, ...) {
2      if (is.list(..1)) {
3          Map(rMap, ..., MoreArgs = list(fun = fun))
4      } else {
5          fun(...)
6      }
7  }
```

Before we proceed with our chosen training routine, we want to perform a single parameter update step given a gradient or gradient-like object. Extracting this into a separate function allows different optimization algorithms to update parameters without needing to understand the internal details of the network. This simplifies the optimization routines while making them more general at the same time.

Two additional features are needed. First, we want to ensure that any constraints (e.g., semi-orthogonality) imposed on the layer weights are preserved. These constraints are specified by the second additional parameter, `constraint`, in the `Layer` constructor, which has been ignored until now. Additionally, we use a trick to drop the bias term when required. This means configuring a layer in the form of $\boldsymbol{x} \mapsto \phi(\boldsymbol{W}\boldsymbol{x})$ without the additive bias term. This works because the internal value of `FALSE` is equal to 0, effectively configures the layer to be the function $\boldsymbol{x} \mapsto \phi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{0})$, while the update routine checks if the bias term is of logical type and has value `FALSE`. If this is the case, the update ignores the bias term.

```
1  gradient.step <- function(network, gradient, step.size = 1e-2) {
2      layers <- environment(network)$layers
3      for (l in seq_along(layers)) {
4          layer <- environment(layers[[l]])
5          layer$weights <- layer$weights - step.size * gradient[[l]]$weights
6          if (!identical(layer$bias, FALSE)) {
7              layer$bias <- layer$bias - step.size * gradient[[l]]$bias
8          }
9          if (is.function(layer$constraint)) {
10             layer$weights <- layer$constraint(layer$weights)
11         }
12     }
13 }
```

The final step is to implement RMSprop, which stands for *root mean squared propagation*

[Hin12]. It is a variation of stochastic gradient descent, a gradient based iterative optimization routine with gradients computed from mini-batches. Building on other similar methods, RMSprop intends to accelerate learning while eliminating the need to manually tune the step size for parameter updates. This is accomplished by an adaptive learning rate applied to every parameter. The basic idea is to keep track of the moving average of the element wise squared gradient, and then use the square root of the moving average to scale the gradient for every parameter, essentially providing a per parameter adaptive learning rate. Provided the current values of the element wise squared gradients $\boldsymbol{G}_2$ and a gradient of the loss $\boldsymbol{G} = \nabla_{\boldsymbol{\theta}} \mathcal{L}(y_b, f(\boldsymbol{X_b}))$, where $\boldsymbol{X}_b, y_b$ is a random subset of the full data set (a mini-batch), the update rule for the parameters $\boldsymbol{\theta}$ of the network has the form (see [Hin12; GBC16] for more details.)

$$\boldsymbol{G}_2 \leftarrow \nu \boldsymbol{G}_2 + (1 - \nu) \boldsymbol{G} \odot \boldsymbol{G},$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\lambda}{\sqrt{\boldsymbol{G}_2} + \epsilon} \odot \boldsymbol{G}.$$

The parameters $\nu = 0.9$, $\lambda = 10^{-2}$, and $\epsilon = 10^{-8}$ are fixed, while the initial value of $\boldsymbol{G}_2$ is set to $\mathbf{0}$. The symbol $\odot$ denotes the *Hadamard product*, representing element wise multiplication. Similarly, element wise operations are applied for division and square root calculations.

```
rmsprop <- function(network, x, y, loss = mse,
    epochs = 100, batch.size = 32
) {
    if (!is.matrix(y)) y <- t(y)

    # initialize squared gradient accumulator
    G2 <- rep(list(weights = 0, bias = 0),
        length(environment(network)$layers))

    # main training loop
    sapply(seq_len(epochs), function(.) {
        # generate random batches
        batches <- matrix(sample.int(ncol(x),
            (ncol(x) %/% batch.size) * batch.size
        ), nrow = batch.size)

        # iterate (almost) entire data set in batches
        for (batch in seq_len(ncol(batches))) {
            # sub-set the batch from the data set
            x_batch <- x[, batches[, batch], drop = FALSE]
            y_batch <- y[, batches[, batch], drop = FALSE]
            # compute the gradient with current batch
            G <- backprop(network, x_batch, y_batch, loss, TRUE)
            G2 <<- rMap(function(g, g2) 0.9 * g2 + 0.1 * g^2, G, G2)
            step <- rMap(function(g, g2) g / (sqrt(g2) + 1e-8), G, G2)
            gradient.step(network, step)
        }

        loss(y, network(x))
    })
}
```

With all of that in place we are done with our framework providing all the needed functionality to provide a simple and clean implementation of the following introduced methods NNOPG and NNSDR.

3.2 Neural Network OPG (NNOPG)

The *Neural Network Outer Product of Gradients* (NNOPG) method, introduced in [KFB22], is the first phase of the NNSDR method (see Section 3.3). Yet, it is a self standing method for the estimation of the central mean subspace $\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$ based on Theorem 2. It uses the same objective as OPG, but instead of local linear regression it uses a neural network to model the conditional distribution $\mathbb{E}[Y \mid \boldsymbol{B}^T\boldsymbol{X}]$ under the forward regression model (2.7). Similar to local linear regression in OPG, which provides gradient estimates at the sample points, any neural network framework (e.g., [MAP+15]) provides gradients of the network approximating any function at the requested input, in our case the sample points.

Assume the mean forward model $Y = g(\boldsymbol{B}^T\boldsymbol{X})+\epsilon$ as in (2.7) and let $f_{\text{OPG}}(\,.\,;\boldsymbol{\Theta}) : \mathbb{R}^p \to \mathbb{R}$ be an MLP modeling the conditional mean function $\boldsymbol{x} \mapsto \mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}]$. Moreover, let $\mathcal{L} : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}_+$ be a differentiable loss function. Assuming $(Y, \boldsymbol{X})$ to be jointly distributed for arbitrary Y and multivariate $\boldsymbol{X} \in \mathbb{R}^p$, we let

$$T_{\text{OPG}}(\boldsymbol{\Theta}) = \mathbb{E}\,\mathcal{L}(Y, f_{\text{OPG}}(\boldsymbol{X};\boldsymbol{\Theta}))$$

be the objective for the NNOPG estimator on the population level. In this setup the MLP f_{OPG} is parameterized by the weights and biases collected in $\boldsymbol{\Theta}$ as in Definition 9.

The selection of the loss function $\mathcal{L}$ is guided by the structure of model (2.7) and the conditional distribution of Y given $\boldsymbol{X}$. If, for example, the error term in (2.7) is normally distributed, the squared error loss function agrees with the loss induced by the likelihood function as a measure of discordance between observed data and the assumed distribution. For situations where Y follows a Bernoulli or multinomial distribution, the cross entropy loss function is appropriate. Conversely, if Y is Poisson distributed, the deviance serves as the natural choice for the loss function. Generally, the loss function can be thought of as the likelihood in the sense that the latter captures the information in the data about a parameter of interest. This also agrees with the loss function employed in generalized linear models for conditional distributions within the exponential family.

For estimation of $\operatorname{span}(\boldsymbol{B}) = \mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$, we minimize the objective function T_{OPG}, that is

$$\boldsymbol{\Theta}_{\text{OPG}} = \arg\min_{\Theta} T_{\text{OPG}}(\boldsymbol{\Theta}). \tag{3.1}$$

The parameters $\boldsymbol{\Theta}_{\text{OPG}}$ then parameterize the neural network f_{OPG} such that the conditonal mean $\mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}] \approx f_{\text{OPG}}(\boldsymbol{x}; \widehat{\boldsymbol{\Theta}}_{\text{OPG}})$. Then, let

$$\boldsymbol{\Sigma}_{\text{OPG}} = \mathbb{E}[\nabla_{\boldsymbol{x}} f_{\text{OPG}}(\boldsymbol{X}, \widehat{\boldsymbol{\Theta}}_{\text{OPG}})(\nabla_{\boldsymbol{x}} f_{\text{OPG}}(\boldsymbol{X}, \widehat{\boldsymbol{\Theta}}_{\text{OPG}}))^T] \tag{3.2}$$

which is an approximation to $\boldsymbol{\Sigma}_\nabla$ in Theorem 2 satisfying $\operatorname{span}(\boldsymbol{\Sigma}_\nabla) = \mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$. We obtain the NNOPG subspace $\mathcal{S}_{\text{NNOPG}} = \operatorname{span}(\boldsymbol{\Sigma}_{\text{OPG}})$.

3.3 Neural Network SDR (NNSDR)

Based again on the mean forward regression model (2.7), the *Neural Network Sufficient Dimension Reduction* (NNSDR) method is introduced. In contrast to NNOPG that estimates $\mathbb{E}(Y \mid \boldsymbol{X})$, NNSDR approximates the conditional mean $\mathbb{E}[Y \mid \boldsymbol{B}^T\boldsymbol{x}] = g(\boldsymbol{B}^T\boldsymbol{x})$ where we incorporate the reduction matrix $\boldsymbol{B}$ directly into the neural network model. The idea is to recreate the function $\boldsymbol{x} \mapsto g(\boldsymbol{B}^T\boldsymbol{x})$ in a literal sense by modeling the reduction $\boldsymbol{B}^T\boldsymbol{x}$ as the first layer of an MLP with a linear activation function. This also resolves a problem of NNOPG, which suffers from the curse of dimensionality. The estimation accuracy of NNOPG is adversely affected by growing input dimension as learning a non-linear functions and their gradients with a high dimensional input space is difficult.

Let f_{NN} be a MLP by Definition 9 with $L+1$ layers of the form

$$f_{\text{NN}}(\boldsymbol{x}; \boldsymbol{\Theta}_{\text{NN}}) = (f_0 \circ f_1 \circ \dots f_L)(\boldsymbol{x})$$

with weight and bias parameters collected in $\boldsymbol{\Theta}_{\text{NN}}$. The first layer (number 0) has the simplified form

$$f_0(\boldsymbol{x}; \boldsymbol{B}) = \boldsymbol{B}^T\boldsymbol{x}$$

which corresponds to a layer with weights $\boldsymbol{B}^T$, bias $\boldsymbol{b} = \boldsymbol{0}$ and a linear activation function $\phi_0(x) = x$. The remaining layers for $l = 1, \dots, L$ have the usual form

$$f_l(\boldsymbol{x}; \boldsymbol{W}_l, \boldsymbol{b}_l) = \phi_l(\boldsymbol{W}_l\boldsymbol{x} + \boldsymbol{b}_l)$$

with weights $\boldsymbol{W}_l$, bias $\boldsymbol{b}_l$ and an activation function ϕ_l. A visualization is provided in Figure 3.2.

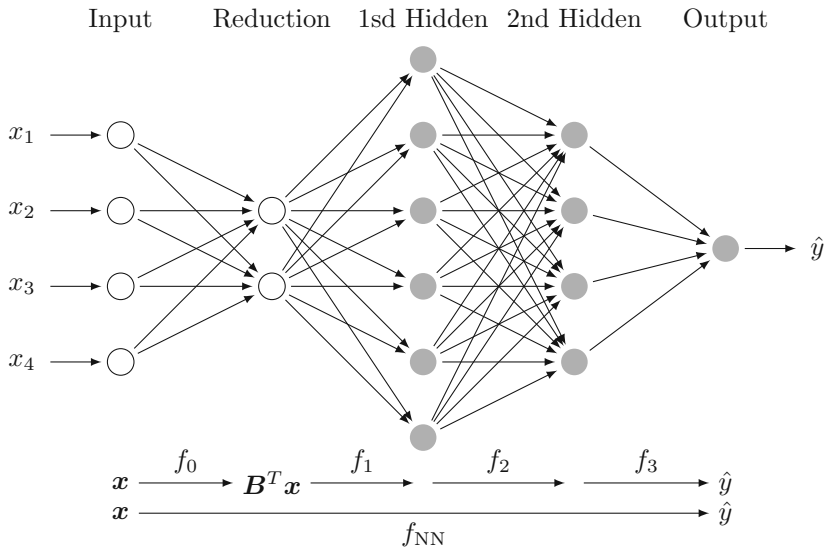

Figure 3.2: Illustration of the NNSDR neural network f_{NN} architecture.

Remark 11. Through the linear activation function $\phi_0(x) = x$ in the first layer, the first two layers collapse into a single affine transformation followed by the activation function of the second hidden layer. Because the input $\boldsymbol{x}$ and the output of the second hidden layer $f_1(f_0(x))$ are both in $\mathbb{R}^p$, but the in-between first hidden layer $f_0(x)$ is of lower dimension q, this is equivalent to a reduced rank constraint on the weights of the collapsed first two layers.

The population training objective is then

$$T_{\text{NN}}(\boldsymbol{\Theta}) = \mathbb{E}\,\mathcal{L}(Y, f_{\text{NN}}(\boldsymbol{X}; \boldsymbol{\Theta})) \tag{3.3}$$

where $\boldsymbol{\Theta}$ are collective parameters of the NNSDR neural network f_{NN} which includes the reduction matrix $\boldsymbol{B}$. With the reduction matrix $\boldsymbol{B}$ incorporated in the network parameters $\boldsymbol{\Theta}$ we get the NNSDR subspace as $\mathcal{S}_{\text{NN}} = \text{span}(\boldsymbol{B}_{\text{NN}})$ where $\boldsymbol{B}_{\text{NN}}$ is extracted from

$$\boldsymbol{\Theta}_{\text{NN}} = \arg\min_{\boldsymbol{\Theta}} T_{\text{NN}}(\boldsymbol{\Theta}).$$

Remark 12. The reduction in the first layer is conceptually similar to *autoencoders* (e.g. [Kra91; KW19]) with the defining difference that autoencoders are analogous to non-linear PCA and independent of the response.

3.4 Dimension Estimation

Up to this point, we have assumed to know the true reduction dimension q. However, in practice, this dimension remains unknown. This section describes the *ladle estimator*, proposed by [LL16], as it fits well into our method for deriving an estimate $\widehat{q}$ for the reduction dimension.

The ladle estimator relies on a sample-based approximation $\widehat{\boldsymbol{M}}$, of a population matrix $\boldsymbol{M}$ with dimensions $p \times p$, to estimate the rank of $\boldsymbol{M}$. It integrates information from the eigenvalues and the variability of the eigenvector directions of $\widehat{\boldsymbol{M}}$ to estimate q. The key idea is that the eigenvalues $\widehat{\lambda}_j$ of $\widehat{\boldsymbol{M}}$ tend towards zero for $j > q$, while the directions of the corresponding eigenvectors experience significant variability.

Let $\widehat{\boldsymbol{B}}_k = (\widehat{\boldsymbol{b}}_1, \ldots, \widehat{\boldsymbol{b}}_k)$ constitute the first k eigenvectors of $\widehat{\boldsymbol{M}}$. To assess the directional variability in the eigenvectors of $\widehat{\boldsymbol{M}}$, we employ bootstrapping on $\widehat{\boldsymbol{M}}$ to obtain N additional estimates, denoted as $\boldsymbol{M}_i^*$ for $i = 1, \ldots, N$, and $\boldsymbol{B}_{k,i}^*$ represents the first k eigenvectors of $\boldsymbol{M}_{k,i}^*$. Subsequently, we define $v_N(k)$ as

$$v_N(k) = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \left|\det \widehat{\boldsymbol{B}}_k^T \boldsymbol{B}_{k,i}^*\right|\right)$$

as a metric for the directional variability, where $k = 1, \ldots, p-1$. For $k = 0$, $v_N(0)$ is set to 0. The range of v_N is $[0, 1]$, with v_N being 0 when all $\boldsymbol{B}_{k,i}^*$ and $\widehat{\boldsymbol{B}}_k$ span the same space, and $v_N(k) = 1$ when the column spaces of $\boldsymbol{B}_{k,i}^*$ are orthogonal to $\widehat{\boldsymbol{B}}_k$.

The final dimension estimate is then

$$\widehat{q} = \arg\min_{k=0,\ldots,p-1} \left(\frac{\sum_{i=0}^{k} v_N(k)}{1 + \sum_{i=0}^{p-1} v_N(i)} + \frac{\widehat{\lambda}_{k+1}}{1 + \sum_{i=1}^{p} \widehat{\lambda}_i} \right).$$

In our context, $\widehat{\boldsymbol{\Sigma}}_{\text{OPG}}$ in (3.2) corresponds to $\widehat{\boldsymbol{M}}$ and serves as an estimate of $\boldsymbol{\Sigma}_\nabla$ in (2) corresponding to $\boldsymbol{M}$. The bootstrap estimates involve drawing N bootstrap samples and re-computing an estimate of $\boldsymbol{\Sigma}_\nabla$. To recompute we bootstrap the given data sample but *do not refit* the neural network. That means that the parameters $\widehat{\boldsymbol{\Theta}}_{\text{OPG}}$ are estimated only once. This way we get all N $\boldsymbol{\Sigma}^\star_{\text{OPG},i}$ for very cheap ($i = 1, ..., N$), compared to training costs. The k first eigenvectors of $\boldsymbol{\Sigma}^\star_{\text{OPG},i}$ are the bootstrapped estimates $\boldsymbol{B}^*_{k,i}$ used to compute v_N at every k.

Typically, considering all $k = 0, ..., p-1$ is neither necessary nor feasible. As recommended in [LL16], it is often sufficient to set an upper bound, e.g., $q \leq \lceil p/\log(p) \rceil$, which we have adopted as a default bound.

3.5 Algorithm

In this section we provide the technical details and needed tricks to solve the optimization problems (3.1) and (3.3) satisfactory. The methods are implemented in `R` using the `R`-package `tensorflow` [AT20] which provides an interface to the TensorFlow [MAP+15] machine learning framework providing almost all the basic algorithms and concepts needed for our approach.

For training neural networks we opted for the *Root Mean Squared Propagation* (RMSprop) [Hin12] algorithm, which is a variation of *Stochastic Gradient Descent* (SDG) [Bot98]. It provides adaptive learning rates for individual parameters during the stochastic optimization. This avoids the problem of a fixed learning rate used in SGD, particularly useful with parameters operating on different scales while at the same time avoids the need to search for a good learning rate. It is even advised for almost all applications to keep the default hyperparameters of the algorithm.

To overcome the problem of small sample sizes, a setting where neural networks usually struggle, we employ *dropout* [SHK+14] during training. Dropout is a regularization technique commonly used in neural networks to prevent overfitting and improve generalization performance. Dropout randomly "drops out" a subset of *neurons* in each layer, that is to set the output of "dropped" neurons to zero. This effectively creating a sparser network. Applied independently to each training sample this leads to some sort of moving average over the sequence of sparser networks. The result is a more robust network with better generalization performance. It is simple, yet powerful and widely used in practice to prevent overfitting, a crucial tool in our setting. We experimented with different default values and arrived to the conclusion that a dropout rate of 40% is a good default for "small" sample sizes.[1]

Another specific issue requires attention, specifically concerning the identifiability of the matrix $\boldsymbol{B}$ as a basis for the mean subspace $\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$ in conjunction with ensuring training stability. We can resolve the problem, in a manner similar to other methods, by constraining the matrix $\boldsymbol{B}$ to the Stiefel manifold (2.8). The constraint enforcement involves a straightforward approach: after each parameter update, we project the parameters associated with the matrix $\boldsymbol{B}$ back onto the Stiefel manifold. While a QR decomposition could serve this

[1] The term "small" is ambiguous and depends strongly on the problem on hand. A rough rule of thumb is that everything with a sample size up to 1,000 is definitely small.

purpose, it has the drawback of causing significant changes in the parameters, leading to catastrophic instabilities during network training. This instability arises because all parameters in the network are interconnected. To overcome this challenge, we employ a stable projection technique, one that minimally alters the parameters themselves. This is achieved by utilizing a *polar decomposition* [Con97, p. 242] to enforce the constraint.

Given a sample $(Y_i, \boldsymbol{X}_i)$ of i.i.d. samples drawn from the joint distribution of $(Y, \boldsymbol{X})$. The model specific configurations, such as the choice of the loss function $\mathcal{L}$, the number of layers L, the number of neurons in each layer n_l for $l = 1, \ldots, L$, and the activation function ϕ_l in each layer, are additional hyperparameters needed by the algorithm. Only one network needs to be specified, as the NNSDR network is constructed from the NNOPG network. Additionally, the batch size and the number of epochs for training must be defined. A reasonable default choice for the batch size is 32, and the number of epochs may be set to 200. The selection of 200 epochs is somewhat arbitrary, driven primarily by the observation that almost all experiments converge within this range. It is crucial to note that due to the high dropout rate, training can be extended as overfitting is prevented through dropout.

The NNSDR estimation procedure is a two stage process. The first stage is the NNOPG estimator providing initial value used by the second stage, which computes the NNSDR estimate.

Stage 1 (NNOPG) : Randomly initialize the parameter vector $\boldsymbol{\Theta}_{\text{OPG}}$ for weights as in Definition 9 for the MLP f_{OPG}. Optimize the sample version of the objective function (3.1) given by

$$T_{\text{OPG}}(\boldsymbol{\Theta}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(Y_i, f_{\text{OPG}}(\boldsymbol{X}_i; \boldsymbol{\Theta}_{\text{OPG}}))$$

using RMSprop (or similar optimization methods) resulting in parameters

$$\widehat{\boldsymbol{\Theta}}_{\text{OPG}} = (\widehat{\boldsymbol{W}}_1, \widehat{\boldsymbol{b}}_1, \ldots, \widehat{\boldsymbol{W}}_L, \widehat{\boldsymbol{b}}_L).$$

Estimte the OPG matrix $\widehat{\boldsymbol{\Sigma}}_{\text{OPG}}$ from (3.2) as

$$\widehat{\boldsymbol{\Sigma}}_{\text{OPG}} = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{x}} f_{\text{OPG}}(\boldsymbol{X}_i, \widehat{\boldsymbol{\Theta}}_{\text{OPG}})(\nabla_{\boldsymbol{x}} f_{\text{OPG}}(\boldsymbol{X}_i, \widehat{\boldsymbol{\Theta}}_{\text{OPG}}))^T. \tag{3.4}$$

If the true mean subspace dimension q is unknown, estimate q as the estimated rank of $\boldsymbol{\Sigma}_\nabla$ using the sample version (3.4) via the ladle estimator from Section 3.4 (keep the parameters $\widehat{\boldsymbol{\Theta}}_{\text{OPG}}$ fixed).

Compute the first q eigenvectors of $\widehat{\boldsymbol{\Sigma}}_{\text{OPG}}$ forming the columns of $\widehat{\boldsymbol{B}}_{\text{OPG}} \in \mathbb{R}^{p\times q}$.

Stage 2 (NNSDR) : Construct a new neural network of the form in Section 3.3 where all layers with index $l = 2, \ldots, L$ are identical to NNOPG network. Ensure that $\widehat{\boldsymbol{\Sigma}}_{\text{OPG}}$ is semi-orthogonal and initialize the parameter vector $\boldsymbol{\Theta}_{\text{NN}}$ as

$$(\widehat{\boldsymbol{B}}_{\text{OPG}}^T, \boldsymbol{0}, \widehat{\boldsymbol{W}}_1\widehat{\boldsymbol{B}}_{\text{OPG}}, \widehat{\boldsymbol{b}}_1, \ldots, \widehat{\boldsymbol{W}}_L, \widehat{\boldsymbol{b}}_L).$$

Minimize the sample version

$$T_{\text{NN}}(\boldsymbol{\Theta}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(Y_i, f_{\text{NN}}(\boldsymbol{X}_i; \boldsymbol{\Theta}))$$

of the objective (3.3) while ensuring that the first layer weights are semi-orthogonal and the bias is zero. After optimizing, the transpose of the first layer weights contains the estimate $\widehat{\boldsymbol{B}}_{\mathrm{NN}}$.

The NNOPG estimate requires only the first stage, whereas the NNSDR estimate relies crucially on the two-stage approach. If the second stage is initialized randomly, the probability of failure is very high. The optimization of T_{NN} is extremely challenging without well-chosen initial values. This is rooted in the fact that for big p and small mean subspace dimension q a random initialization of the parameters associated with $\boldsymbol{B} \in \mathbb{R}^{p\times q}$ often results in them being nearly orthogonal to the true mean subspace $\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$. Consequently, the first layer loses all information about the response, leading to a breakdown in the optimization. This is the reason for the development of NNOPG in the first place.

Under the squared error loss $\mathcal{L}(y,\widehat{y}) = (y-\widehat{y})^2$ for a univariate response Y, the NNOPG objective in (3.1) is identical to the OPG objective (see Section 2.3.6). Under certain regularity conditions [XTL+02], the OPG objective is known to be consistent for estimating $\operatorname{span}(\boldsymbol{\Sigma}_{\nabla}) = \mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$, as stated in Theorem 2. A similar scenario, under the square error loss for univariate Y, arises for the NNSDR objective in (3.1), which is identical to the MAVE objective (see Section 2.3.5). As demonstrated in Theorem 1, the MAVE estimate is consistent, and this consistency extends to the NNSDR estimate as well if $f_{\mathrm{NN}}(\,.\,;\widehat{\boldsymbol{\Theta}}_{\mathrm{NN}})$ is a consistent estimator of g in (2.7).

However, it is important to note that both cases hinge on establishing the functional approximation consistency of neural networks, a challenging problem that, to the best of our knowledge, remains an open question.

Remark 13. An appealing characteristic of NNSDR, unlike MAVE, is its inherent applicability for online training when new data become available, thanks to the SDG based optimization algorithm. More precisely, the method seamlessly adjusts the parameters of NNSDR through additional gradient steps given new data.

3.5.1 Implementation in R

Using our framework from Section 3.1.1 an implementation of NNOPG is.

```
nnopg <- function(X, y, dims, network = NULL,
    loss = mse, epochs = 200, batch.size = 32
) {
    if (is.null(network)) {
        # Use default network architecture
        network <- NN(
            Layer(c(nrow(X), 512), activation = relu, dropout = 0.4),
            Layer(c(512, 1), activation = identity)
        )
    }
    # train the network
    rmsprop(network, X, y, loss, epochs, batch.size)
    # Outer Product of Gradients (3.2)
    Sigma <- var(t(grad(network)(X)))
    # Reduction subspace basis as dims first eigenvectors
    eigen(Sigma)$vectors[, seq_len(dims), drop = FALSE]
}
```

To ensure the semi-orthogonality of the first layers weights, corresponding to the estimated reduction, we need to implement the polar decomposition based contraint.

```
semi.orthogonal <- function(A) with(La.svd(A), u %*% vt)
```

With NNOPG as first stage of NNSDR we have;

```
nnsdr <- function(X, y, dims, opg.net = NULL,
    loss = mse, epochs = 200, batch.size = 32
) {
    if (is.null(opg.net)) {
        # Use default OPG network architecture
        opg.net <- NN(
            Layer(c(nrow(X), 512), activation = relu, dropout = 0.4),
            Layer(c(512, 1), activation = identity)
        )
    }
    # Invoke NNOPG for an initial estimate
    B.opg <- nnopg(X, y, dims, opg.net, loss, epochs, batch.size)
    # Get first OPG network layer
    f_1 <- environment(environment(opg.net)$layers[[1]])
    new.layers <- c(
        # f_0 : x ↦ B_OPG^T x
        Layer(t(B.opg), bias = FALSE, constraint = polar),
        # f_1 : x ↦ φ_1((W_1 B_OPG)x + b_1)
        Layer(
            weights = f_1$weights %*% B.opg,
            bias = f_1$bias,
            activation = f_1$activation,
            dropout = f_1$dropout),
        # remaining OPG layers f_2, ..., f_L
        environment(opg.net)$layers[-1]
    )
    network <- do.call(NN, new.layers)
    # train NNSDR network
    rmsprop(network, X, y, loss, epochs, batch.size)
    # extract finetuned reduction matrix B from first layer
    t(environment(environment(network)$layers[[1]])$weights)
}
```

3.5.2 MNIST handwritten digits dataset

We demonstrate NNSDR on the MNIST dataset [LBB+98], which cannot be analyzed by other forward regression based SDR methods such as OPG, MAVE, or CVE, due to its size. However, for neural networks, this dataset can be considered the "Hello World" dataset.

The MNIST dataset [LBB+98] consists of 60,000 labeled grayscale images of handwritten digits, each with dimensions of 28 by 28 pixels, along with a test set containing 10,000 labeled images. It is a 10-class classification problem with a predictor dimension of 784. For the multi-class classification problem, we use the *categorical cross-entropy* loss

$$\mathcal{L}(y, \widehat{y}) = \sum_{i=1}^{10} y_i \log(\widehat{y}_i)$$

where $y \in \{0,1\}^{10}$ represents *one-hot encoded* labels and $\widehat{y} \in (0,1)^{10}$ denotes the neural network outputs. The neural network is a 2-layer MLP with 256 and 128 hidden units and ReLU activation, respectively, followed by 10 output neurons representing each digit with a *softmax* output activation

$$\text{softmax}(x) = \frac{\exp(\boldsymbol{x})}{\|\exp(\boldsymbol{x})\|_1}$$

where $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{p} |\boldsymbol{x}_i|$. This normalizes softmax($\boldsymbol{x}$) to sum to 1 with exp is applied element wise. We achieve a test set classification accuracy of approximately 97% after training for 15 epochs with a dropout rate of 1/3. Subsequently, we apply NNSDR which simultaneously fits a prediction model on the reduced data. Figure 3.3 illustrates the data projected onto a 2 dimensional subspace using the NNSDR estimated reduction, revealing visually distinguishable clusters. In contrast, PCA reduction appears to blur clusters other than those corresponding to digits 0 and 1. Table 3.1 shows the classification accuracy of the NNSDR network (Stage 2) as a prediction model using reduced data across various dimensions, yielding approximately 70% accuracy for the 2 dimensional reduction shown in Figure 3.3. Starting at dimension $q = 6$ for $p = 784$, which is a reduction to less than 1% of the original size, an classification accuracy of more than 90% is achieved.

q	1	2	3	4	5	6	7	8	9	10	784
Train	39.3	68.8	82.0	87.7	89.3	93.0	94.0	95.0	95.7	96.1	99.5
Test	40.2	69.5	82.0	87.5	89.3	92.9	93.6	94.4	94.9	95.4	97.9

Table 3.1: Training and test accuracy of the refinement network for reduction dimensions $q = 1, ..., 10$ for the MNIST dataset with the last column for $q = p$, meaning no reduction.

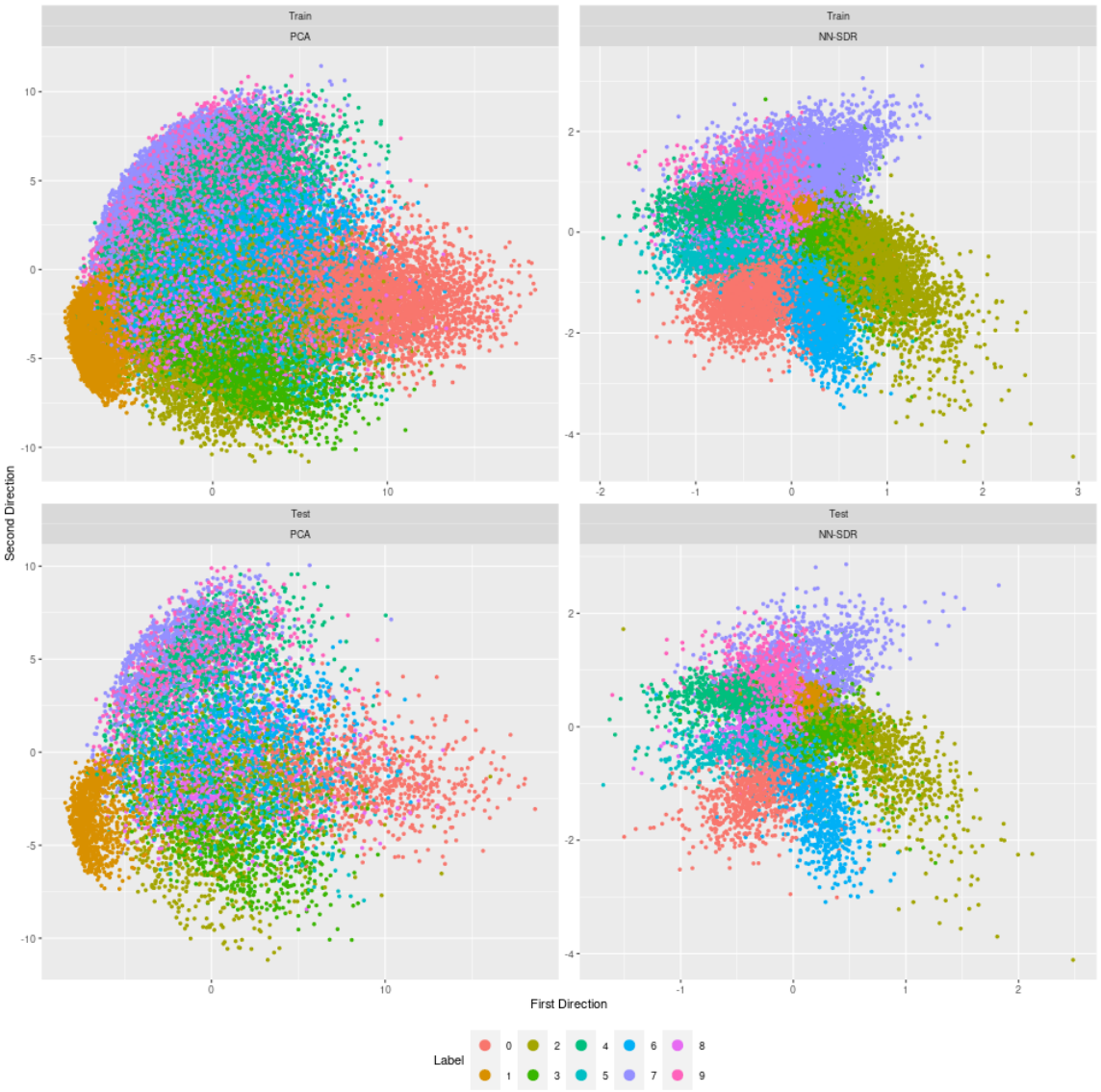

Figure 3.3: 2D reduction of the MNIST dataset using PCA (left panels) and NNSDR (right panels).

4 Kronecker Parametric Inverse Regression (KPIR)

In this chapter we present *Kronecker Parametric Inverse Regression* (KPIR) and *Kronecker Principal Fitted Components* (KPFC), extensions of PIR (see Section 2.3.3) and PFC (see Section 2.3.4), respectively, to matrix valued predictors $\boldsymbol{X}$ [PKB21]. Both methods built on the corresponding PIR model (2.9) and PFC model (2.10) by assuming a Kronecker structure of $\boldsymbol{A}$ or $\boldsymbol{\Gamma}$, $\boldsymbol{\gamma}$, respectively.

4.1 Matrix-Valued Inverse Regression Models

The setup is as in Section 2.3.3 except that the predictors $\boldsymbol{X}$ are matrix valued of dimension $p_1 \times p_2$. To accommodate the matrix structure we adapt the PIR model (2.9) by assuming that $\boldsymbol{X}$ decomposes into separate row and column components. This is accomplished by replacing the linear dependence of $\boldsymbol{X}$ in the inverse regression model PIR by a bilinear dependence of the form

$$\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{\alpha}_1 \boldsymbol{F}(Y) \boldsymbol{\alpha}_2^T + \boldsymbol{\epsilon} \tag{4.1}$$

where $\boldsymbol{F}(Y)$ is a known matrix valued function in Y of dimension $q_1 \times q_2$ and mean zero $\mathbb{E}\, \boldsymbol{F}(Y) = \boldsymbol{0}$. The $p_k \times q_k$ matrices $\boldsymbol{\alpha}_k$ are unconstrained for $k = 1, 2$. The first matrix $\boldsymbol{\alpha}_1$ describes the row mean structure of $\boldsymbol{X}$ and $\boldsymbol{\alpha}_2$ the column mean structure. The error is assumed to be centered $\mathbb{E}[\boldsymbol{\epsilon}] = \boldsymbol{0}$ and has conditional variance $\operatorname{Var}(\operatorname{vec} \boldsymbol{\epsilon} \mid Y) = \operatorname{Var}(\operatorname{vec} \boldsymbol{X} \mid Y) = \boldsymbol{\Delta}_Y$. The vectorized form of model (4.1) is

$$\operatorname{vec} \boldsymbol{X} = \operatorname{vec} \boldsymbol{\mu} + (\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1) \operatorname{vec} \boldsymbol{F}(Y) + \operatorname{vec} \boldsymbol{\epsilon}. \tag{4.2}$$

The vectorized model (4.2) and the bilinear model (4.1) are equivalent. In the vectorized form the relation to the PIR model (2.9) is clear under the constraint on the parameter matrix $\boldsymbol{A} = \boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1$. This is why we call it Kronecker PIR.

Remark 14. The indexing scheme, where the indices in $\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1$ are "reversed," align with the convention of Chapter 5. In (4.1) and (4.2), these indices correspond to the modes of the matrix $\boldsymbol{F}(Y)$. The rows count as the first mode and the columns as the second. This particular indexing scheme will be convenient in the multilinear setting of Chapter 5, and it is adopted here for the sake of consistency.

In [PFB12] was shown that the first moment based SDR subspace $\mathcal{S}_{\text{FMSDR}}$ under model (4.2) is given by

$$\mathcal{S}_{\text{FMSDR}} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \operatorname{span}(\mathbb{E}[\boldsymbol{X} \mid Y]) = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \operatorname{span}(\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1) = \boldsymbol{\Delta}^{-1} \operatorname{span}(\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1)$$

where $\boldsymbol{\Sigma}_{\boldsymbol{X}} = \operatorname{Var}(\operatorname{vec} \boldsymbol{X})$ and $\boldsymbol{\Delta} = \mathbb{E}(\boldsymbol{\Delta}_Y)$. The inclusion of the first moment SDR subspace in the central subspace is a direct result of the linearity Condition 1 on $\boldsymbol{X}$, but equality requires knowledge of the model. The dimension of the subspace is given by the rank of $\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1$, which is the product of the ranks of $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$. Therefore, $\dim(\mathcal{S}_{\text{FMSDR}}) = \operatorname{rank}(\boldsymbol{\alpha}_1) \operatorname{rank}(\boldsymbol{\alpha}_2)$.

The PFC model (2.10) adapted for matrix valued predictors $\boldsymbol{X}$ has the form

$$\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{\Gamma}_1 \boldsymbol{\gamma}_1 \boldsymbol{F}(Y) \boldsymbol{\gamma}_2^T \boldsymbol{\Gamma}_2^T + \boldsymbol{\epsilon} \tag{4.3}$$

where $\boldsymbol{\Gamma}_k \in \mathbb{R}^{p_k \times d_k}$ are semi-orthogonal, $\boldsymbol{\gamma}_k \in \mathbb{R}^{d_k \times q_k}$, for $k = 1, 2$, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Delta})$. The PFC model (4.3) is a restricted KPIR model in that it requires $\boldsymbol{\alpha}_k$ have a fixed rank $d_k = \operatorname{rank}(\boldsymbol{\alpha}_k)$, which in turn yields the factorization $\boldsymbol{\alpha}_k = \boldsymbol{\Gamma}_k \boldsymbol{\gamma}_k$ for $\boldsymbol{\gamma}_k$ of full rank d_k (which requires $q_k \leq d_k$) for $k = 1, 2$. Vectorization of (4.3) yields the equivalent model

$$\begin{aligned} \operatorname{vec} \boldsymbol{X} &= \operatorname{vec} \boldsymbol{\mu} + (\boldsymbol{\Gamma}_2 \boldsymbol{\gamma}_2 \otimes \boldsymbol{\Gamma}_1 \boldsymbol{\gamma}_1) \operatorname{vec} \boldsymbol{F}(Y) + \operatorname{vec} \boldsymbol{\epsilon} \\ &= \operatorname{vec} \boldsymbol{\mu} + (\boldsymbol{\Gamma}_2 \otimes \boldsymbol{\Gamma}_1)(\boldsymbol{\gamma}_2 \otimes \boldsymbol{\gamma}_1) \operatorname{vec} \boldsymbol{F}(Y) + \operatorname{vec} \boldsymbol{\epsilon} \end{aligned}$$

which shows the relation to the PFC model (2.10) as the matrices $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}_2 \otimes \boldsymbol{\Gamma}_1$ and $\boldsymbol{\gamma} = \boldsymbol{\gamma}_2 \otimes \boldsymbol{\gamma}_1$ both have Kronecker constraints.

The first moment SDR subspace follows directly from the KPIR model, of which the KPFC model is a constrained version. Letting $d_k = \operatorname{rank}(\boldsymbol{\alpha}_k)$, then

$$\mathcal{S}_{\text{FMSDR}} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \operatorname{span}(\boldsymbol{\Gamma}_2 \otimes \boldsymbol{\Gamma}_1) = \boldsymbol{\Delta}^{-1} \operatorname{span}(\boldsymbol{\Gamma}_2 \otimes \boldsymbol{\Gamma}_1)$$

as $\operatorname{span}(\boldsymbol{\alpha}_k) = \operatorname{span}(\boldsymbol{\Gamma}_k \boldsymbol{\gamma}_k) = \operatorname{span}(\boldsymbol{\Gamma}_k)$, for $k = 1, 2$.

Assuming additionally that $\boldsymbol{\Sigma}_{\boldsymbol{X}}$ is separable, that is $\boldsymbol{\Sigma}_{\boldsymbol{X}} = \boldsymbol{\Sigma}_2 \otimes \boldsymbol{\Sigma}_1$ for $\boldsymbol{\Sigma}_k \in \mathbb{R}^{p_k \times p_k}$, $k = 1, 2$, we get

$$\mathcal{S}_{\text{FMSDR}} = \operatorname{span}(\boldsymbol{\Sigma}_2^{-1} \boldsymbol{\Gamma}_2 \otimes \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\Gamma}_1).$$

A slightly less restrictive alternative assumption is the separability of $\boldsymbol{\Delta} = \mathbb{E}\, \boldsymbol{\Delta}_Y = \mathbb{E} \operatorname{Var}(\operatorname{vec} \boldsymbol{X} \mid Y) = \boldsymbol{\Delta}_2 \otimes \boldsymbol{\Delta}_1$ which yields

$$\mathcal{S}_{\text{FMSDR}} = \operatorname{span}(\boldsymbol{\Delta}_2^{-1} \boldsymbol{\Gamma}_2 \otimes \boldsymbol{\Delta}_1^{-1} \boldsymbol{\Gamma}_1).$$

4.2 Estimation Procedures

We provide multiple approaches for estimating the component matrices $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$ under model (4.1) and $\boldsymbol{\Gamma}_1$, $\boldsymbol{\Gamma}_2$ for model (4.3). We operate under the assumption that the dimensions d_1, d_2 are known.

Before we describe the different estimation procedures we introduce a notation for gathering an i.i.d. sample $(\boldsymbol{X}_i, Y_i)$ in a convenient matrix form, used throughout the remainder of this chapter. The observed predictors $\boldsymbol{X}_i$ are collected in the $p_1 p_2 \times n$ matrix $\mathbb{X}$ with the ith column containing the vectorized and centered observations $\operatorname{vec}(\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}})$ where $\widehat{\boldsymbol{\mu}} = n^{-1} \sum_{i=1}^n \boldsymbol{X}_i$. For the matrix valued function $\boldsymbol{F}(Y_i)$ of the responses Y_i, we collect the data similarly in an $q_1 q_2 \times n$ matrix $\mathbb{F}$ with columns consisting of the vectorized and centered functions $\operatorname{vec}(\boldsymbol{F}(Y_i) - \widehat{\boldsymbol{\nu}})$ with $\widehat{\boldsymbol{\nu}} = n^{-1} \sum_{i=1}^n \boldsymbol{F}(Y_i)$.

Remark 15. Storing the data in a matrix, with *columns indexing observations*, offers several advantages. First, there is no need to transpose sample-level versions of the model. Second, in multilinear models, such as those discussed in Chapter 5, it is very confusing on the sample level when the first mode indexes observations, requiring the addition of 1 in mode products. To eliminate this confusion, samples are arranged in the last mode, aligning with indexing observations in columns. Third, when using the programming language `R`, one can simplify many operations by leveraging the built-in recycling feature of most `R` functions, avoiding the need for cumbersome workarounds.

4.2.1 Least Squares Kronecker PIR (KPIR (ls))

We start with a *least squares* approach for the estimation of $\mathcal{S}_{\text{FMSDR}}$ under the KPIR model (4.2). Using the notation introduced at the beginning of this chapter, the sample level version of model (4.2) is

$$\mathbb{X} = (\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1)\mathbb{F} + \boldsymbol{\epsilon}$$

where the error term $\boldsymbol{\epsilon}$ is a matrix with i.i.d. columns $\boldsymbol{\epsilon}_i$ with zero mean $\mathbb{E}[\boldsymbol{\epsilon}_i] = \mathbf{0}$ and conditional variance $\operatorname{Var}(\boldsymbol{\epsilon}_i \mid Y = y_i) = \boldsymbol{\Delta}_{y_i}$.

The basis for the least squares estimate, as well as the other methods introduced later, is the following result.

Theorem 4. *Assume the data collected in* $\mathbb{X}$ *follow model* (4.2). *Let* $\widehat{\boldsymbol{A}} = \mathbb{X}\mathbb{F}^T(\mathbb{F}\mathbb{F}^T)^{-1}$ *denote the ordinary least squares estimate in the unconstrained model* $\mathbb{X} = \boldsymbol{A}\mathbb{F} + \boldsymbol{\epsilon}$. *The solutions* $\widehat{\boldsymbol{\alpha}}_1$, $\widehat{\boldsymbol{\alpha}}_2$ *of*

$$(\widehat{\boldsymbol{\alpha}}_2, \widehat{\boldsymbol{\alpha}}_1) = \underset{\boldsymbol{\alpha}_2, \boldsymbol{\alpha}_1}{\arg\min} \|\widehat{\boldsymbol{A}} - \widehat{\boldsymbol{\alpha}}_2 \otimes \widehat{\boldsymbol{\alpha}}_1\|_F^2 \tag{4.4}$$

converge in probability to $\boldsymbol{\alpha}_1$ *and* $\boldsymbol{\alpha}_2$ *under the KPIR model* (4.2); *i.e.,*

$$\widehat{\boldsymbol{\alpha}}_2 \otimes \widehat{\boldsymbol{\alpha}}_1 \xrightarrow{p} \boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1$$

and $\widehat{\boldsymbol{\alpha}}_2 \otimes \widehat{\boldsymbol{\alpha}}_1$ *is asymptotically normal.*

Proof. By [VP93, Thm. 2.1] we rewrite the optimization objective of (4.4) as

$$\|\widehat{\boldsymbol{A}} - \widehat{\boldsymbol{\alpha}}_2 \otimes \widehat{\boldsymbol{\alpha}}_1\|_F = \|\mathcal{R}(\widehat{\boldsymbol{A}}) - \operatorname{vec}(\widehat{\boldsymbol{\alpha}}_2)\operatorname{vec}(\widehat{\boldsymbol{\alpha}}_1)^T\|_F$$

where $\mathcal{R} : \mathbb{R}^{p_1 p_2 \times q_1 q_2} \to \mathbb{R}^{p_2 q_2 \times p_1 q_1}$ is a permutation and reshaping operation (that is $\operatorname{vec}(\mathcal{R}(\boldsymbol{A}))$ is a permutation of $\operatorname{vec}(\boldsymbol{A})$, see also Section 2.4.1, particularly Lemma 1). By [VP93, Corollary 2.2] the rank 1 approximation $\mathcal{R}_1(\widehat{\boldsymbol{A}})$ of $\mathcal{R}(\widehat{\boldsymbol{A}})$ based on SVD solves the least squares problem (4.4). Under model (4.2), which is equivalent to (4.1), the true parameter matrix $\boldsymbol{A} = \boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1$ is separable and the OLS estimate $\widehat{\boldsymbol{A}} \xrightarrow{p} \boldsymbol{A}$ is consistent and asymptotically normal. Since $\mathcal{R}$ is continuous, $\mathcal{R}(\widehat{\boldsymbol{A}}) \xrightarrow{p} \mathcal{R}(\boldsymbol{A})$ by the continuous mapping theorem. Moreover, in a neighborhood of $\boldsymbol{A}$ the rank 1 approximation $\mathcal{R}_1(\widehat{\boldsymbol{A}})$ is unique and continuous. Therefore, applying the delta method to the asymptotically normal least squares estimate $\widehat{\boldsymbol{A}}$, we also obtain that $\mathcal{R}_1(\widehat{\boldsymbol{A}})$ is asymptotically normal with mean $\mathcal{R}_1(\boldsymbol{A}) = \mathcal{R}(\boldsymbol{A})$. Applying the inverse operation $\mathcal{R}^{-1}$ gives

$$\widehat{\boldsymbol{\alpha}}_2 \otimes \widehat{\boldsymbol{\alpha}}_1 = \mathcal{R}^{-1}(\mathcal{R}_1(\widehat{\boldsymbol{A}})) \xrightarrow{p} \mathcal{R}^{-1}(\mathcal{R}_1(\boldsymbol{A})) = \mathcal{R}^{-1}(\mathcal{R}(\boldsymbol{A})) = \boldsymbol{A} = \boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1.$$

By the delta method, $\widehat{\boldsymbol{\alpha}}_2 \otimes \widehat{\boldsymbol{\alpha}}_1$ is asymptotically normal with mean $\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1$. □

All decompositions obtained as a solution to (4.4) will be referred to as VLP approximations. This is because it is based on the results obtained by Van Loan and Pitsianis in their paper [VP93].

We also need an estimate of $\boldsymbol{\Delta} = \mathbb{E}\,\boldsymbol{\Delta}_Y$, which, given estimates $\widehat{\boldsymbol{\alpha}}_1$, $\widehat{\boldsymbol{\alpha}}_2$, has a least squares estimate

$$\widehat{\boldsymbol{\Delta}}_{\text{ls}} = \frac{1}{n - \operatorname{rank}(\mathbb{F})}\big(\mathbb{X} - (\widehat{\boldsymbol{\alpha}}_2 \otimes \widehat{\boldsymbol{\alpha}}_1)\mathbb{F}\big)\big(\mathbb{X} - (\widehat{\boldsymbol{\alpha}}_2 \otimes \widehat{\boldsymbol{\alpha}}_1)\mathbb{F}\big)^T \tag{4.5}$$

leading to the least squares estimate $\widehat{\mathcal{S}}_{\text{KPIR(ls)}} = \widehat{\boldsymbol{\Delta}}_{\text{ls}}^{-1} \operatorname{span}(\widehat{\boldsymbol{\alpha}}_2 \otimes \widehat{\boldsymbol{\alpha}}_1)$ of $\mathcal{S}_{\text{FMSDR}}$.

4.2.1.1 Implementation in R

Next a short R implementation of [VP93, Framework. 1] with `p` and `q` are the dimension tuples (p_1, p_2) and (q_1, q_2), respectively. The matrix $\boldsymbol{A}$ to be VLP approximated has to have dimension $p_1 p_2 \times q_1 q_2$.

```
approx.kron <- function(A, p, q) {
    R <- aperm(`dim<-`(A, c(p, q)), c(2, 4, 1, 3))

    with(svd(`dim<-`(R, p * q), 1, 1), list(
        alpha2 = matrix(sqrt(d[1]) * u, p[2]),
        alpha1 = matrix(sqrt(d[1]) * v, p[1])
    ))
}
```

With the VLP approximation `approx.kron`, an implementation of KPIR(ls) is straightforward.

```
kpir.ls <- function(X, F, p, q, d) {
    # Solve least squares problem Â = 𝕏𝔽^T(𝔽𝔽^T)^-1
    A <- t(solve(tcrossprod(F), tcrossprod(F, X)))
    # VLP approximation Â ≈ α̂2 ⊗ α̂1
    A <- with(approx.kron(A, c(p[2], q[2]), c(p[1], q[1])),
        kronecker(alpha2, alpha1)
    )
    # Least Squares Delta estimate Δ̂ls
    Delta <- tcrossprod(X - crossprod(A, F)) / (ncol(X) - qr(F)$rank)

    # Basis estimate of S_FMSDR ≈ Δ̂ls^-1 span(Γ̂2 ⊗ Γ̂1) ⊆ Δ̂ls^-1 span(α̂2 ⊗ α̂1)
    Gamma1 <- La.svd(alpha1, d[1], 0)$u
    Gamma2 <- La.svd(alpha2, d[2], 0)$u
    solve(Delta, kronecker(Gamma2, Gamma1))
}
```

4.2.2 Maximum Likelihood Kronecker PIR (KPIR (mle))

By further assuming $\boldsymbol{X} \mid Y$ under the vectorized KPIR model (4.2) follows a normal distribution, we can derive a maximum likelihood estimate. Specifically, on the sample level, this means

$$\operatorname{vec}(\boldsymbol{X}_i) \sim \mathcal{N}_{p_1 p_2}(\operatorname{vec}(\boldsymbol{\mu}) + (\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1)\operatorname{vec}(\boldsymbol{F}(Y_i) - \mathbb{E}\,\boldsymbol{F}(Y)), \boldsymbol{\Delta}), \tag{4.6}$$

The log-likelihood (dropping the constant factor and scaling by 2) for n i.i.d. observations, collected in $\mathbb{X}$ and $\mathbb{F}$, is

$$l(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\Delta}) = -n \log |\boldsymbol{\Delta}| - (\mathbb{X} - (\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1)\mathbb{F})\boldsymbol{\Delta}^{-1}(\mathbb{X} - (\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1)\mathbb{F})^T. \tag{4.7}$$

Remark 16. The full likelihood including the marginal mean $\boldsymbol{\mu} = \mathbb{E}\,\boldsymbol{X}$ leads to the MLE of $\boldsymbol{\mu}$ as $\widehat{\boldsymbol{\mu}} = n^{-1}\sum_{i=1}^{n} \boldsymbol{X}_i$. This is already included in $\mathbb{X}$, which in turn leads to the identical full maximum likelihood solutions for the remaining parameters $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$ and $\boldsymbol{\Delta}$.

Solving the score equations for $\boldsymbol{\Delta}$ fixing both $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ gives

$$\widehat{\boldsymbol{\Delta}} = \frac{1}{n}\big(\mathbb{X} - (\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1)\mathbb{F}\big)\big(\mathbb{X} - (\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1)\mathbb{F}\big)^T \tag{4.8}$$

while the score equations for $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$ do *not* possess a closed form solution. To solve this we opt for an iterative algorithm for maximizing the log-likelihood (4.7) by alternating between an (arbitrary) numeric optimization algorithm for $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$ fixing $\boldsymbol{\Delta}$ and updating $\boldsymbol{\Delta}$ with fixed $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$. This is repeated until convergence (or a maximum number of iterations is exceeded). We determine convergence in the I'th iteration by the two criteria $\|\widehat{\boldsymbol{\Delta}}_I - \widehat{\boldsymbol{\Delta}}_{I-1}\| \leq \delta\|\widehat{\boldsymbol{\Delta}}_{I-1}\|$ and $\|\widehat{\boldsymbol{A}}_I - \widehat{\boldsymbol{A}}_{I-1}\| \leq \delta\|\widehat{\boldsymbol{A}}_{I-1}\|$ with $\widehat{\boldsymbol{A}}_I = \widehat{\boldsymbol{\alpha}}_{2I} \otimes \widehat{\boldsymbol{\alpha}}_{1I}$ for a small constant $\delta > 0$. The whole procedure is initialized with the least squares estimates from Section 4.2.1. After convergence, the subspace $\mathcal{S}_{\text{FMSDR}}$ is estimated as

$$\widehat{\mathcal{S}}_{\text{KPIR(mle)}} = \widehat{\boldsymbol{\Delta}}^{-1}\,\text{span}(\widehat{\boldsymbol{\Gamma}}_2 \otimes \widehat{\boldsymbol{\Gamma}}_1)$$

where $\widehat{\boldsymbol{\Gamma}}_k$ are the first d_k left singular vectors of $\widehat{\boldsymbol{\alpha}}_k$ for $k = 1, 2$.

4.2.2.1 Implementation in R

For the MLE version of KPIR, we implement an iterative algorithm. We safeguard the method by adding a maximum number of iterations to avoid (in theory never occurring) infinite loops. The additional `delta` is used in the break condition. Using `optim`, a numeric optimization routine in base R, we have an implementation of KPIR(mle).

```
kpir.mle <- function(X, F, p, q, d, max.iter = 10, delta = 1e-2) {
    # Solve least squares problem
    A <- t(solve(crossprod(F), crossprod(F, X)))
    # VLP approximation
    alphas <- approx.kron(A, c(p[2], q[2]), c(p[1], q[1]))
    A <- do.call(kronecker, alphas)
    # MLE Delta estimate given alphas
    Delta <- tcrossprod(X - crossprod(A, F)) / ncol(X)
    Delta.inv <- solve(Delta)

    # negative log likelihood (with Delta fixed)
    log.likelihood <- function(par) {
        alpha2 <- matrix(head(par, p[2] * q[2]), p[2])
        alpha1 <- matrix(tail(par, p[1] * q[1]), p[1])
        error <- X - crossprod(kronecker(alpha2, alpha1), F)
        sum(error * (error %*% Delta.inv))
    }

```

```
    # Iterate till convergence (or max iter reached)
    for (iter in seq_len(max.iter)) {
        # Optimize log-likelihood for alpha1, alpha2 with fixed Delta.
        opt <- optim(par = c(alpha2, alpha1), fn = log.likelihood)
        # Store previous alphas and Delta (for break consition).
        alphas.last <- alphas
        Delta.last <- Delta
        # Unpack combined optimized alphas
        alpha2 <- matrix(head(par, p[2] * q[2]), p[2])
        alpha1 <- matrix(tail(par, p[1] * q[1]), p[1])
        A <- kronecker(alpha2, alpha1)
        # Recompute MLE Delta estimate with fixed alphas
        Delta <- tcrossprod(X - crossprod(A, F)) / ncol(X)
        Delta.inv <- solve(Delta)

        # Check break conditions
        if (norm(Delta - Delta.last, "F") < delta * norm(Delta.last, "F")) {
            if (norm(B - B.last, "F") < delta * norm(B.last, "F")) {
                break
            }
        }
    }

    # FMSDR subspace basis
    Gamma2 <- La.svd(alpha2, d[2], 0)$u
    Gamma1 <- La.svd(alpha1, d[1], 0)$u
    Delta.inv %*% kronecker(Gamma2, Gamma1)
}
```

4.2.3 Kronecker PFC (KPFC)

The KPFC model (4.3) leads to a different log-likelihood with parameters $\boldsymbol{\Gamma}_k$, $\boldsymbol{\gamma}_k$ and $\boldsymbol{\Delta}$, for $k = 1, 2$, with data $\mathbb{X}$ and $\mathbb{F}$ as

$$l(\boldsymbol{\Gamma}_1, \boldsymbol{\Gamma}_2, \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\Delta}) = \\ - n \log |\boldsymbol{\Delta}| - (\mathbb{X} - (\boldsymbol{\Gamma}_2 \otimes \boldsymbol{\Gamma}_1)(\boldsymbol{\gamma}_2 \otimes \boldsymbol{\gamma}_1)\mathbb{F})\boldsymbol{\Delta}^{-1}(\mathbb{X} - (\boldsymbol{\Gamma}_2 \otimes \boldsymbol{\Gamma}_1)(\boldsymbol{\gamma}_2 \otimes \boldsymbol{\gamma}_1)\mathbb{F})^T \quad (4.9)$$

which we scaled by 2 and dropped the constant factor.

Remark 17. As in the KPIR log-likelihood (4.7), the MLE for $\boldsymbol{\mu} = \mathbb{E}\,\boldsymbol{X}$ in (4.9) is the sample mean.

We let $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}_2 \otimes \boldsymbol{\Gamma}_1$ be a semi-orthogonal $p_1 p_2 \times d_1 d_2$ matrix and $\boldsymbol{\gamma} = \boldsymbol{\gamma}_2 \otimes \boldsymbol{\gamma}_1$ be full rank $d_1 d_2$, but otherwise unconstrained of dimensions $d_1 d_2 \times q_1 q_2$. Substituting in (4.9) gives the PFC log-likelihood

$$l(\boldsymbol{\Gamma}, \boldsymbol{\gamma}, \boldsymbol{\Delta}) = -n \log |\boldsymbol{\Delta}| - (\mathbb{X} - \boldsymbol{\Gamma}\boldsymbol{\gamma}\mathbb{F})\boldsymbol{\Delta}^{-1}(\mathbb{X} - \boldsymbol{\Gamma}\boldsymbol{\gamma}\mathbb{F})^T.$$

The maximum likelihood estimates for the PFC log-likelihood were derived in [CF08]. With $\widehat{\boldsymbol{A}} = \mathbb{X}^T\mathbb{F}(\mathbb{F}\mathbb{F}^T)^{-1}$ being the multivariate regression coefficients and $\boldsymbol{P}_{\mathbb{F}} = \mathbb{F}^T(\mathbb{F}\mathbb{F}^T)^{-1}\mathbb{F}$ the hat matrix for the same linear regression, we let $\widehat{\boldsymbol{\Delta}}_{\text{fit}}$ denote the variance of the fitted values

and $\widehat{\boldsymbol{\Delta}}_{\text{res}}$ the estimated residual variance given by

$$\widehat{\boldsymbol{\Delta}}_{\text{fit}} = \frac{1}{n}\mathbb{X}\boldsymbol{P}_{\mathbb{F}}\mathbb{X}^T = \frac{1}{n}\mathbb{X}\mathbb{X}^T - \widehat{\boldsymbol{\Delta}}_{\text{res}}.$$

Now we compute the singular value decomposition $\widehat{\boldsymbol{U}}\widehat{\boldsymbol{D}}\widehat{\boldsymbol{U}}^T = \widehat{\boldsymbol{\Delta}}_{\text{res}}^{-1/2}\widehat{\boldsymbol{\Delta}}_{\text{fit}}\widehat{\boldsymbol{\Delta}}_{\text{res}}^{-1/2}$ used to provide the PFC maximum likelihood estimate of $\boldsymbol{\Delta}$ given by

$$\widehat{\boldsymbol{\Delta}} = \widehat{\boldsymbol{\Delta}}_{\text{res}} + \widehat{\boldsymbol{\Delta}}_{\text{fit}}^{1/2}\widehat{\boldsymbol{U}}\widehat{\boldsymbol{D}}_{>d_1d_2}\widehat{\boldsymbol{U}}^T\widehat{\boldsymbol{\Delta}}_{\text{fit}}^{1/2} \tag{4.10}$$

where $\widehat{\boldsymbol{D}}_{>d_1d_2} = \text{diag}(0,\ldots,0,\widehat{\boldsymbol{\lambda}}_{d_1d_2+1},\ldots,\widehat{\boldsymbol{\lambda}}_{p_1p_2})$. Given $\widehat{\boldsymbol{\Delta}}$,

$$\text{span}(\widehat{\boldsymbol{\Gamma}}) = \widehat{\boldsymbol{\Delta}}^{1/2}\,\text{span}_{d_1d_2}(\widehat{\boldsymbol{\Delta}}^{-1/2}\widehat{\boldsymbol{\Delta}}_{\text{fit}}\widehat{\boldsymbol{\Delta}}^{-1/2}), \tag{4.11}$$

$$\widehat{\boldsymbol{\gamma}} = (\widehat{\boldsymbol{\Gamma}}^T\widehat{\boldsymbol{\Delta}}^{-1}\widehat{\boldsymbol{\Gamma}})^{-1}\widehat{\boldsymbol{\Gamma}}^T\widehat{\boldsymbol{\Delta}}^{-1}\widehat{\boldsymbol{A}} \tag{4.12}$$

where span_d denotes the span of the first d left singular vectors of its argument. The matrix $\widehat{\boldsymbol{\Gamma}}$ is only specified up to its span which is the meaning behind the notation. For computational purposes we simply take any semi-orthogonal matrix $\widehat{\boldsymbol{\Gamma}}$ which fulfils (4.11).

Remark 18. If $d_1d_2 = q_1q_2$ then $\widehat{\boldsymbol{\Delta}} = \widehat{\boldsymbol{\Delta}}_{\text{res}}$ which is due to $\text{rank}(\widehat{\boldsymbol{\Delta}}_{\text{fit}}) = q_1q_2$ resulting in $\widehat{\boldsymbol{D}}_{>d_1d_2} = \boldsymbol{0}$.

The parameters of interest are the components of $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}_2 \otimes \boldsymbol{\Gamma}_1$ which we retrieve from the PFC estimate $\widehat{\boldsymbol{\Gamma}}$ using the VLP approximation in Theorem 4. The resulting components $\widehat{\boldsymbol{\Gamma}}_1, \widehat{\boldsymbol{\Gamma}}_2$ are least squares estimates of the objective $\|\widehat{\boldsymbol{\Gamma}} - \boldsymbol{\Gamma}_2 \otimes \boldsymbol{\Gamma}_1\|_F$ which do *not* guarantee to maximize the log-likelihood (4.9).

There are basically two ways of getting different estimates for the first moment subspace $\mathcal{S}_{\text{FMSDR}}$ in the form of a folded subspace. One is to apply the VLP approximation to $\widehat{\boldsymbol{\Gamma}}\widehat{\boldsymbol{\gamma}}$ to get estimates $\widehat{\boldsymbol{\alpha}}_1$, $\widehat{\boldsymbol{\alpha}}_2$ from which we estimate $\widehat{\boldsymbol{\Gamma}}_1$, $\widehat{\boldsymbol{\Gamma}}_2$ as their d_1, d_2 first left singular vectors, respectively. The alternative is to first compute $\widehat{\boldsymbol{\Gamma}}$ and then apply the VLP approximation to obtain $\widehat{\boldsymbol{\Gamma}}_1$, $\widehat{\boldsymbol{\Gamma}}_2$. In both cases, the estimated subspace is

$$\widehat{\mathcal{S}}_{\text{KPFC}} = \widehat{\boldsymbol{\Delta}}^{-1}\,\text{span}(\widehat{\boldsymbol{\Gamma}}_2 \otimes \widehat{\boldsymbol{\Gamma}}_1).$$

Remark 19. Here we are only interested in estimating $\mathcal{S}_{\text{FMSDR}}$. In [PKB21] there are three variations on how to decompose the estimates. The basic approach is to estimate $\boldsymbol{\Delta}^{-1}\,\text{span}(\boldsymbol{\alpha}_2 \otimes \boldsymbol{\alpha}_1)$ where the span is based on the d_k first eigenvectors of the $\widehat{\boldsymbol{\alpha}}_k$'s. The three different ways of estimating the $\boldsymbol{\alpha}_k$'s are: (a) First, decompose $\widehat{\boldsymbol{\Gamma}}\widehat{\boldsymbol{\gamma}} \approx \widehat{\boldsymbol{\alpha}}_2 \otimes \widehat{\boldsymbol{\alpha}}_1$. The other two first decompose $\widehat{\boldsymbol{\Gamma}} \approx \widehat{\boldsymbol{\Gamma}}_2 \otimes \widehat{\boldsymbol{\Gamma}}_1$ and then compute $\widehat{\boldsymbol{\gamma}} = ((\widehat{\boldsymbol{\Gamma}}_2 \otimes \widehat{\boldsymbol{\Gamma}}_1)^T\widehat{\boldsymbol{\Delta}}^{-1}(\widehat{\boldsymbol{\Gamma}}_2 \otimes \widehat{\boldsymbol{\Gamma}}_1))^{-1}(\widehat{\boldsymbol{\Gamma}}_2 \otimes \widehat{\boldsymbol{\Gamma}}_1)^T\widehat{\boldsymbol{\Delta}}^{-1}\widehat{\boldsymbol{A}}$ instead of using (4.12). Then either (b) decompose $(\widehat{\boldsymbol{\Gamma}}_2 \otimes \widehat{\boldsymbol{\Gamma}}_1)\widehat{\boldsymbol{\gamma}} \approx \widehat{\boldsymbol{\alpha}}_2 \otimes \widehat{\boldsymbol{\alpha}}_1$ or (c) approximate $\widehat{\boldsymbol{\gamma}} \approx \widehat{\boldsymbol{\gamma}}_2 \otimes \widehat{\boldsymbol{\gamma}}_1$ to get $\widehat{\boldsymbol{\alpha}}_k = \widehat{\boldsymbol{\Gamma}}_k\widehat{\boldsymbol{\gamma}}_k$. The latter two versions (b) and (c) give the same estimated subspace because $\text{span}(\widehat{\boldsymbol{\Gamma}}_2 \otimes \widehat{\boldsymbol{\Gamma}}_1) = \text{span}((\widehat{\boldsymbol{\Gamma}}_2 \otimes \widehat{\boldsymbol{\Gamma}}_1)\widehat{\boldsymbol{\gamma}}) = \text{span}((\widehat{\boldsymbol{\Gamma}}_2 \otimes \widehat{\boldsymbol{\Gamma}}_1)(\widehat{\boldsymbol{\gamma}}_2 \otimes \widehat{\boldsymbol{\gamma}}_1))$ based on $\widehat{\boldsymbol{\gamma}}$ and $\widehat{\boldsymbol{\gamma}}_2 \otimes \widehat{\boldsymbol{\gamma}}_1$ having full row rank.

Remark 20. KPIR(ls) is based purely on model (4.1) without any distributional assumptions. All three KPIR(mle) and the two versions of KPFC assume $\boldsymbol{X} \mid Y$ to be normal distributed. KPIR(mle) is based on model (4.1) whereas the two versions of KPFC are based on model (4.3).

4.2.3.1 Implementation in R

Combining both versions in a single method toggled with `version` we get.

```
kpfc <- function(X, F, p, q, d, version = 2) {
    # Solve least squares problem
    A <- t(solve(tcrossprod(F), tcrossprod(F, X)))
    P.F <- crossprod(F, solve(tcrossprod(F), F))
    Delta.fit <- tcrossprod(X, X %*% P.F) / ncol(X)
    Delta.res <- tcrossprod(X) / ncol(X) - Delta.fit
    # Compute Delta
    tmp <- matpow(Delta.res, -1 / 2)
    SVD <- La.svd(D %*% Delta.fit %*% D)
    D <- SVD$d * (seq_along(SVD$d) > prod(d))
    tmp <- matpow(Delta.res, 1 / 2)
    Delta <- Delta.res + tmp %*% (SVD$u * D) %*% SVD$vt %*% tmp
    Delta.inv <- matpow(Delta, -1)

    # MLE estimte of (full) Gamma
    tmp <- matpow(Delta, -1 / 2)
    Gamma <- matpow(Delta, 1 / 2) %*%
        La.svd(tmp %*% Delta.fit %*% tmp, prod(d))$u

    # version 1 computes ‘gamma‘ and VLP approx. ‘Gamma gamma‘
    if (version = 1) {
        tmp <- crossprod(Gamma, Delta.inv)
        gamma <- solve(tmp %*% Gamma, tmp %*% A)

        # VLP approx
        alphas <- approx.kron(Gamma %*% gamma, p, q)
        Gamma1 <- La.svd(alphas[[1]], d[1])$u
        Gamma2 <- La.svd(alphas[[2]], d[2])$u
    # version 2 VLP approx. ‘Gamma‘ directly
    } else {
        Gammas <- approx.kron(Gamma, p, d)
        Gamma1 <- Gammas[[1]]
        Gamma2 <- Gammas[[2]]
    }

    # FMSDR estimated subspace
    Delta.inv %*% kronecker(Gamma2, Gamma1)
}
```

In Figure 4.1 the computational dependence of KPIR(ls), KPIR(mle) and KPFC, in both version KPFC(v1) and KPFC(v2), are visualized similar to [PKB21, Fig. 1].

A real data application is given in conjunction with GMLM in Section 5.6.1.

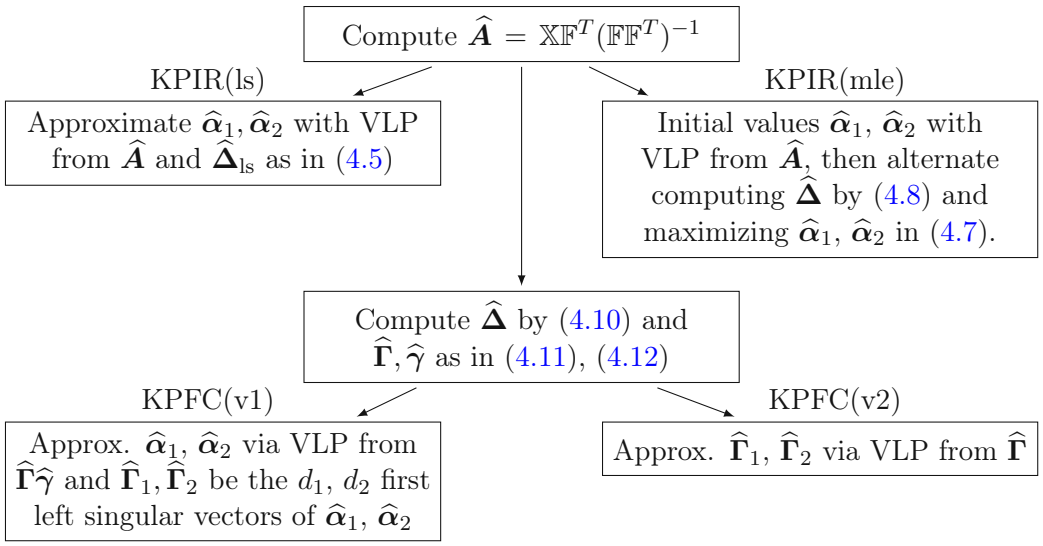

Figure 4.1: Flow Chart of KPIR and KPFC algorithms.

5 Generalized Multilinear Models

5.1 The Generalized Multi-Linear Model (GMLM)

We assume the distribution of $\mathcal{X} \mid Y$ belongs to the *quadratic exponential family*, in order to simplify modeling and keep estimation feasible. We assume that $\mathcal{X} \mid Y$ is a full rank quadratic exponential family with density

$$\begin{aligned} f_{\boldsymbol{\eta}_y}(\mathcal{X} \mid Y = y) &= h(\mathcal{X}) \exp(\boldsymbol{\eta}_y^T \boldsymbol{t}(\mathcal{X}) - b(\boldsymbol{\eta}_y)) \\ &= h(\mathcal{X}) \exp(\langle \boldsymbol{t}_1(\mathcal{X}), \boldsymbol{\eta}_{1y} \rangle + \langle \boldsymbol{t}_2(\mathcal{X}), \boldsymbol{\eta}_{2y} \rangle - b(\boldsymbol{\eta}_y)) \end{aligned} \tag{5.1}$$

where $\boldsymbol{t}_1(\mathcal{X}) = \operatorname{vec} \mathcal{X}$ and $\boldsymbol{t}_2(\mathcal{X})$ is linear in $\mathcal{X} \circ \mathcal{X}$. The dependence of $\mathcal{X}$ on Y is fully captured in the natural parameter $\boldsymbol{\eta}_y$. The function h is non-negative real-valued and b is assumed to be at least twice continuously differentiable and strictly convex. An important feature of the *quadratic exponential family* is that the distribution of its members is fully characterized by their first two moments. Distributions within the quadratic exponential family include the *tensor normal* and *tensor Ising model* (a generalization of the (inverse) Ising model which is multi-variate Bernoulli with up to second order interactions) and mixtures of these two.

In model (5.1), the dependence of $\mathcal{X}$ and Y is absorbed in $\boldsymbol{\eta}_y$, and $\boldsymbol{t}(\mathcal{X})$ is the minimal sufficient statistic for the *pseudo*-parameter $\boldsymbol{\eta}_y = (\boldsymbol{\eta}_{1y}, \boldsymbol{\eta}_{2y})$ with

$$\boldsymbol{t}(\mathcal{X}) = (\boldsymbol{t}_1(\mathcal{X}), \boldsymbol{t}_2(\mathcal{X})) = (\operatorname{vec} \mathcal{X}, \boldsymbol{T}_2 \operatorname{vech}((\operatorname{vec} \mathcal{X})(\operatorname{vec} \mathcal{X})^T)), \tag{5.2}$$

where the $d \times p(p+1)/2$ dimensional matrix $\boldsymbol{T}_2$ with $p = \prod_{i=1}^r p_i$ ensures that $\boldsymbol{\eta}_{2y}$ is of minimal dimension d. The matrix $\boldsymbol{T}_2$ is of full rank d and unique to different members of the quadratic exponential family. We can reexpress the exponent in (5.1) as

$$\begin{aligned} \boldsymbol{\eta}_y^T \boldsymbol{t}(\mathcal{X}) &= \langle \operatorname{vec} \mathcal{X}, \boldsymbol{\eta}_{1y} \rangle + \langle \boldsymbol{T}_2 \operatorname{vech}(\mathcal{X} \circ \mathcal{X}), \boldsymbol{\eta}_{2y} \rangle \\ &= \langle \operatorname{vec} \mathcal{X}, \boldsymbol{\eta}_{1y} \rangle + \langle \operatorname{vec}(\mathcal{X} \circ \mathcal{X}), (\boldsymbol{T}_2 \boldsymbol{D}_p^\dagger)^T \boldsymbol{\eta}_{2y} \rangle \end{aligned}$$

where $\boldsymbol{D}_p$ is the *duplication matrix* from Abadir and Magnus [AM05, Ch. 11], defined so that $\boldsymbol{D}_p \operatorname{vech} \boldsymbol{A} = \operatorname{vec} \boldsymbol{A}$ for every symmetric $p \times p$ matrix $\boldsymbol{A}$, and $\boldsymbol{D}_p^\dagger$ is its Moore-Penrose pseudo inverse. The first natural parameter component, $\boldsymbol{\eta}_{1y}$, captures the first order, and $\boldsymbol{\eta}_{2y}$, the second order relationship of Y and $\mathcal{X}$. The quadratic exponential density of $\mathcal{X} \mid Y$ can then be expressed as

$$f_{\eta_y}(\mathcal{X} \mid Y = y) = h(\mathcal{X}) \exp\Big(\langle \operatorname{vec} \mathcal{X}, \boldsymbol{\eta}_{1y} \rangle + \langle \operatorname{vec}(\mathcal{X} \circ \mathcal{X}), (\boldsymbol{T}_2 \boldsymbol{D}_p^\dagger)^T \boldsymbol{\eta}_{2y} \rangle - b(\boldsymbol{\eta}_y)\Big) \tag{5.3}$$

The exponential family in (5.3) is easily generalizable to any order. This, though, would result in the number of parameters becoming prohibitive to estimate, which is also the reason why we opted for the second order exponential family in our formulation.

By the equivalence of forward SDR (2.2) and inverse SDR (2.3), in order to find the sufficient reduction $\mathcal{R}(\mathcal{X})$ we need to infer $\boldsymbol{\eta}_{1y}$, and $\boldsymbol{\eta}_{2y}$. This is reminiscent of generalized linear modeling, which we extend to a multi-linear formulation next. Suppose $\mathcal{F}_y$ is a known mapping of y with zero expectation $\mathbb{E}_Y \mathcal{F}_Y = 0$. We assume the dependence of $\mathcal{X}$ and Y is reflected only in the first parameter and let

$$\boldsymbol{\eta}_{1y} = \operatorname{vec} \overline{\eta} + \boldsymbol{B} \operatorname{vec} \mathcal{F}_y, \tag{5.4}$$

$$\boldsymbol{\eta}_2 = ((\boldsymbol{T}_2 \boldsymbol{D}_p^\dagger)^\dagger)^T \operatorname{vec}(c\, \boldsymbol{\Omega}), \tag{5.5}$$

where $\overline{\eta} \in \mathbb{R}^{p_1 \times \cdots \times p_r}$, $\boldsymbol{\Omega} \in \mathbb{R}^{p \times p}$ is positive definite with $p = \prod_{j=1}^r p_j$, and $c \in \mathbb{R}$ is a known constant determined by the distribution to ease modeling. That is, we assume that only $\boldsymbol{\eta}_{1y}$ depends on Y through $\boldsymbol{B}$. The second parameter $\boldsymbol{\eta}_2$ captures the second order interaction structure of $\mathcal{X}$, which we assume not to depend on the response Y. In order to relate individual modes of $\mathcal{X}$ to the response, allowing flexibility in modeling, we assume $\mathcal{F}_y$ takes values in $\mathbb{R}^{q_1 \times \cdots \times q_r}$; that is, $\mathcal{F}_y$ is a tensor valued independent variable. This, in turn, leads to imposing corresponding tensor structure to the regression parameter $\boldsymbol{B}$. Thus, (5.4) becomes

$$\boldsymbol{\eta}_{1y} = \operatorname{vec}\left(\overline{\eta} + \mathcal{F}_y \bigtimes_{j=1}^{r} \boldsymbol{\beta}_j \right), \tag{5.6}$$

where $\boldsymbol{B} = \bigotimes_{j=r}^{1} \boldsymbol{\beta}_j$ and the component matrices $\boldsymbol{\beta}_j \in \mathbb{R}^{p_j \times q_j}$ are of known rank for $j = 1, \ldots, r$.

As the bilinear form of the matrix normal requires its covariance be separable, the multilinear structure of $\mathcal{X}$ also induces separability on its covariance structure (see, e.g., Hoff [Hof11]). Therefore, we further assume that

$$(\boldsymbol{T}_2 \boldsymbol{D}_p^\dagger)^T \boldsymbol{\eta}_{2y} = (\boldsymbol{T}_2 \boldsymbol{D}_p^\dagger)^T \boldsymbol{\eta}_2 = \operatorname{vec}\left(c \bigotimes_{j=r}^{1} \boldsymbol{\Omega}_j \right), \tag{5.7}$$

where $\boldsymbol{\Omega}_j \in \mathbb{R}^{p_j \times p_j}$ are symmetric positive definite matrices for $j = 1, \ldots, r$. Requiring $\boldsymbol{\Omega} = \bigotimes_{j=r}^{1} \boldsymbol{\Omega}_j$ substantially reduces the number of parameters to estimate in $\boldsymbol{\Omega}$. The assumption that the $\boldsymbol{\Omega}_j$'s be positive definite is possible due to the constant c.

Equation (5.7) is underdetermined since $(\boldsymbol{T}_2 \boldsymbol{D}_p^\dagger)^T$ has full column rank $d < p^2$ (with a non-strict inequality if $\mathcal{X}$ is univariate) but $\boldsymbol{\eta}_2$ is uniquely determined given any $\boldsymbol{\Omega}$ as $((\boldsymbol{T}_2 \boldsymbol{D}_p^\dagger)^\dagger)^T$ has full row rank. We let $\boldsymbol{\xi} = (\operatorname{vec} \overline{\eta}, \operatorname{vec} \boldsymbol{B}, \operatorname{vech} \boldsymbol{\Omega})$ be the unconstrained $p(p + 2q + 3)/2$-parameter vector and $\boldsymbol{\theta} = (\operatorname{vec} \overline{\eta}, \operatorname{vec} \boldsymbol{B}, \operatorname{vech} \boldsymbol{\Omega})$ be the constrained parameter vector, where $\boldsymbol{B} = \bigotimes_{j=r}^{1} \boldsymbol{\beta}_j$ and $\boldsymbol{\Omega} = \bigotimes_{j=r}^{1} \boldsymbol{\Omega}_j$. We also let Ξ and Θ denote the unconstrained and constrained parameter spaces, with $\boldsymbol{\xi}$ and $\boldsymbol{\theta}$ varying in Ξ and Θ, respectively. The parameter space Ξ is an open subset of $\mathbb{R}^{p(p+2q+3)/2}$ so that (5.3) is a proper density. We relax the assumptions for $\boldsymbol{\beta}_k$ and $\boldsymbol{\Omega}_k$ later as a consequence of Theorem 9 in Section 5.3.2.

In a classical *generalized linear model* (GLM), the link function connecting the natural parameters to the expectation of the sufficient statistic $\boldsymbol{\eta}_y = \boldsymbol{g}(\mathbb{E}[\boldsymbol{t}(\mathcal{X}) \mid Y = y])$ is invertible. Such a link may not exist in our setting, but for our purpose what we call the "inverse" link suffices. The "inverse" link $\widetilde{\boldsymbol{g}}$ exists as the natural parameters fully describe the distribution.

As in the non-minimal formulation (5.3), we define the "inverse" link through its tensor valued components

$$\mathfrak{g}_1(\boldsymbol{\eta}_y) = \mathbb{E}[\mathcal{X} \mid Y = y], \tag{5.8}$$
$$\mathfrak{g}_2(\boldsymbol{\eta}_y) = \mathbb{E}[\mathcal{X} \circ \mathcal{X} \mid Y = y] \tag{5.9}$$

as $\widetilde{\boldsymbol{g}}(\boldsymbol{\eta}_y) = (\operatorname{vec} \mathfrak{g}_1(\boldsymbol{\eta}_y), \operatorname{vec} \mathfrak{g}_2(\boldsymbol{\eta}_y))$. Under the quadratic exponential family model (5.3), a sufficient reduction for the regression of Y on $\mathcal{X}$ is given in Theorem 5.

Theorem 5 (Generalized Multi-Linear SDR)**.** *A sufficient reduction for the regression $Y \mid \mathcal{X}$ under the quadratic exponential family inverse regression model (5.3) with natural parameters (5.6) and (5.7) is given by*

$$\mathcal{R}(\mathcal{X}) = (\mathcal{X} - \mathbb{E}\,\mathcal{X}) \bigtimes_{k=1}^{r} \boldsymbol{\beta}_j^T. \tag{5.10}$$

The reduction (5.10) is minimal if $\boldsymbol{\beta}_j$ are full rank for all $j = 1, \ldots, r$.

Proof. A direct implication of Bura, Duarte, and Forzani [BDF16, Theorem 1] is that, under the exponential family (5.3) with natural statistic (5.2),

$$\boldsymbol{\alpha}^T(\boldsymbol{t}(\mathcal{X}) - \mathbb{E}\,\boldsymbol{t}(\mathcal{X}))$$

is a sufficient reduction, where $\boldsymbol{\alpha} \in \mathbb{R}^{(p+d)\times q}$ with $\operatorname{span}(\boldsymbol{\alpha}) = \operatorname{span}(\{\boldsymbol{\eta}_y - \mathbb{E}_Y\,\boldsymbol{\eta}_Y : y \in \mathcal{S}_Y\})$. Since $\mathbb{E}_Y\,\mathcal{F}_Y = 0$, $\mathbb{E}_Y\,\boldsymbol{\eta}_{1Y} = \mathbb{E}[\operatorname{vec}\overline{\eta} - \boldsymbol{B}\operatorname{vec}\mathcal{F}_Y] = \operatorname{vec}\overline{\eta}$. Thus,

$$\boldsymbol{\eta}_y - \mathbb{E}_Y\,\boldsymbol{\eta}_Y = \begin{pmatrix} \boldsymbol{\eta}_{1y} - \mathbb{E}_Y\,\boldsymbol{\eta}_{1Y} \\ \boldsymbol{\eta}_2 - \mathbb{E}_Y\,\boldsymbol{\eta}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{B}\operatorname{vec}\mathcal{F}_y \\ \boldsymbol{0} \end{pmatrix}.$$

as $\boldsymbol{\eta}_2$ does not depend on y. The set $\{\operatorname{vec}\mathcal{F}_y : y \in \mathcal{S}_y\}$ is a subset of $\mathbb{R}^q$. Therefore,

$$\operatorname{span}\left(\{\boldsymbol{\eta}_y - \mathbb{E}_Y\,\boldsymbol{\eta}_Y : y \in \mathcal{S}_Y\}\right) = \operatorname{span}\left(\left\{\begin{pmatrix} \boldsymbol{B}\operatorname{vec}\mathcal{F}_Y \\ \boldsymbol{0} \end{pmatrix} : y \in \mathcal{S}_Y\right\}\right) \subseteq \operatorname{span}\begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{pmatrix},$$

which obtains that

$$\begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{pmatrix}^T (\boldsymbol{t}(\mathcal{X}) - \mathbb{E}\,\boldsymbol{t}(\mathcal{X})) = \boldsymbol{B}^T \operatorname{vec}(\mathcal{X} - \mathbb{E}\,\mathcal{X}) = \operatorname{vec}\Big(\mathcal{F}_y \bigtimes_{k=1}^{r} \boldsymbol{\beta}_k\Big)$$

is also a sufficient reduction, though not necessarily minimal, using $\boldsymbol{B} = \bigotimes_{k=1}^{r} \boldsymbol{\beta}_k$. When the exponential family is full rank, which in our setting amounts to all $\boldsymbol{\beta}_j$ being full rank matrices, $j = 1, \ldots, r$, then Bura, Duarte, and Forzani [BDF16, Thm 1] also obtains the minimality of the reduction. □

The reduction in vectorized form is $\operatorname{vec}\mathcal{R}(\mathcal{X}) = \boldsymbol{B}^T \operatorname{vec}(\mathcal{X} - \mathbb{E}\,\mathcal{X})$, where $\boldsymbol{B} = \bigotimes_{k=r}^{1} \boldsymbol{\beta}_k$ with $\operatorname{span}(\boldsymbol{B}) = \operatorname{span}(\{\boldsymbol{\eta}_{1y} - \mathbb{E}_Y\,\boldsymbol{\eta}_{1Y} : y \in \mathcal{S}_Y\})$, using $\mathcal{S}_Y$ to denote the set of values of the random variable Y.

Theorem 5 obtains that the *sufficient reduction* $\mathcal{R}(\mathcal{X})$ reduces $\mathcal{X}$ along each dimension linearly. The graph in Figure 5.1 is a visual depiction of the sufficient reduction.

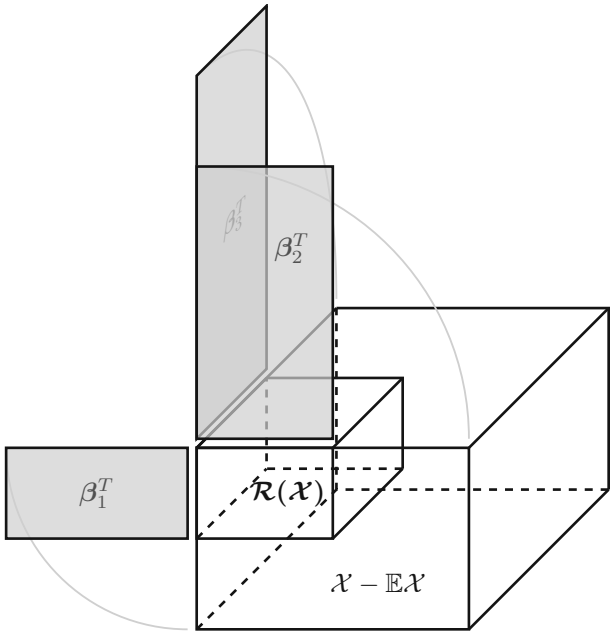

Figure 5.1: Visual depiction of the sufficient reduction in Theorem 5.

Example 3 (Vector valued $\boldsymbol{x}$ $(r = 1)$)**.** *Given vector valued predictor $\boldsymbol{X} \in \mathbb{R}^p$, the tensor order is $r = 1$, then the collection of parameters is $\boldsymbol{\theta} = (\overline{\boldsymbol{\eta}}, \boldsymbol{\beta}, \boldsymbol{\Omega})$ with $\overline{\boldsymbol{\eta}} \in \mathbb{R}^p$, $\boldsymbol{\beta} \in \mathbb{R}_*^{p \times q}$ and $\boldsymbol{\Omega} \in \mathrm{Sym}_{++}^{p \times p}$ where $\boldsymbol{f}_y \in \mathbb{R}^q$ are known functions of the response Y. The conditional density of $\boldsymbol{X} \mid Y = y$ is given by*

$$\begin{aligned} f_\theta(\boldsymbol{x} \mid Y = y) &= h(\boldsymbol{x}) \exp(\langle \boldsymbol{x}, \boldsymbol{\eta}_{1y}(\boldsymbol{\theta}) \rangle + \langle \mathrm{vec}(\boldsymbol{x} \circ \boldsymbol{x}), \boldsymbol{\eta}_2(\boldsymbol{\theta}) \rangle - b(\boldsymbol{\eta}_y(\boldsymbol{\theta}))) \\ &= h(\boldsymbol{x}) \exp((\overline{\boldsymbol{\eta}} + \boldsymbol{\beta} \boldsymbol{f}_y)^T \boldsymbol{x} + c\, \boldsymbol{x}^T \boldsymbol{\Omega}\, \boldsymbol{x} - b(\boldsymbol{\eta}_y(\boldsymbol{\theta}))). \end{aligned}$$

using the relation of $\boldsymbol{\theta}$ to the natural parameters given by $\boldsymbol{\eta}_{1y}(\boldsymbol{\theta}) = \overline{\boldsymbol{\eta}} + \boldsymbol{\beta} \boldsymbol{f}_y$ and $\boldsymbol{\eta}_2(\theta) = c\, \boldsymbol{\Omega}$.

Example 4 (Matrix valued $\boldsymbol{X}$ $(r = 2)$)**.** *Assuming $\boldsymbol{X}$ to be matrix valued, that is $r = 2$, $\boldsymbol{\theta} = (\overline{\boldsymbol{\eta}}, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2)$, where the intercept term $\overline{\boldsymbol{\eta}} \in \mathbb{R}^{p_1 \times p_2}$ is now matrix valued. Similar to Example 3 with $\boldsymbol{F}_y \in \mathbb{R}^{q_1 \times q_2}$ being matrix valued, the conditional density of $\boldsymbol{X} \mid Y = y$ reads*

$$\begin{aligned} f_{\boldsymbol{\theta}}(\boldsymbol{X} \mid Y = y) &= h(\boldsymbol{X}) \exp(\langle \mathrm{vec}\, \boldsymbol{X}, \boldsymbol{\eta}_{1y}(\boldsymbol{\theta}) \rangle + \langle \mathrm{vec}(\boldsymbol{X} \circ \boldsymbol{X}), \boldsymbol{\eta}_2(\boldsymbol{\theta}) \rangle - b(\boldsymbol{\eta}_y(\boldsymbol{\theta}))) \\ &= h(\boldsymbol{X}) \exp(\mathrm{tr}((\overline{\boldsymbol{\eta}} + \boldsymbol{\beta}_1 \boldsymbol{F}_y \boldsymbol{\beta}_2^T) \boldsymbol{X}^T) + c\, \mathrm{tr}(\boldsymbol{\Omega}_1 \boldsymbol{X} \boldsymbol{\Omega}_2 \boldsymbol{X}^T) - b(\boldsymbol{\eta}_y(\boldsymbol{\theta}))). \end{aligned}$$

5.2 Maximum Likelihood Estimation

Suppose $(\mathcal{X}_i, Y_i)$ are independently and identically distributed with joint cdf $F(\mathcal{X}, Y)$, for $i = 1, \ldots, n$. The empirical log-likelihood function of (5.3) under (5.6) and (5.7), dropping terms not depending on the parameters, is

$$l_n(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left(\left\langle \overline{\eta} + \mathcal{F}_{y_i} \bigtimes_{k=1}^{r} \beta_k, \mathcal{X}_i \right\rangle + c \left\langle \mathcal{X}_i \bigtimes_{k=1}^{r} \boldsymbol{\Omega}_k, \mathcal{X}_i \right\rangle - b(\boldsymbol{\eta}_{y_i}) \right). \tag{5.11}$$

The maximum likelihood estimate of $\boldsymbol{\theta}_0$ is the solution to the optimization problem

$$\widehat{\boldsymbol{\theta}}_n = \arg\max_{\boldsymbol{\theta}\in\Theta} l_n(\boldsymbol{\theta}) \tag{5.12}$$

with $\widehat{\boldsymbol{\theta}}_n = (\operatorname{vec}\widehat{\overline{\eta}}, \operatorname{vec}\widehat{\boldsymbol{B}}, \operatorname{vech}\widehat{\boldsymbol{\Omega}})$ where $\widehat{\boldsymbol{B}} = \bigotimes_{k=r}^{1}\widehat{\boldsymbol{\beta}}_k$ and $\widehat{\boldsymbol{\Omega}} = \bigotimes_{k=r}^{1}\widehat{\boldsymbol{\Omega}}_k$.

A straightforward and general method for parameter estimation is *gradient descent*. To apply gradient based optimization, we compute the gradients of l_n in Theorem 6.

Theorem 6. *For n i.i.d. observations $(\mathcal{X}_i, y_i), i = 1, ..., n$ the log-likelihood is of the form in (5.11) with $\boldsymbol{\theta}$ being the collection of all GMLM parameters $\overline{\eta}$, $\boldsymbol{B} = \bigotimes_{k=r}^{1}\boldsymbol{\beta}_k$ and $\boldsymbol{\Omega} = \bigotimes_{k=r}^{1}\boldsymbol{\Omega}_k$ for $k = 1, ..., r$. Let $\mathcal{G}_2(\boldsymbol{\eta}_y)$ be a tensor of dimensions $p_1, \ldots, p_r$ such that*

$$\operatorname{vec}\mathcal{G}_2(\boldsymbol{\eta}_y) = (\boldsymbol{T}_2\boldsymbol{D}_p^{\dagger})^{\dagger}\boldsymbol{T}_2\boldsymbol{D}_p^{\dagger}\operatorname{vec}\mathfrak{g}_2(\boldsymbol{\eta}_y).$$

Then, the partial gradients with respect to $\overline{\eta}, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_r, \boldsymbol{\Omega}_1, \ldots, \boldsymbol{\Omega}_r$ are given by

$$\begin{aligned}
\nabla_{\overline{\eta}} l_n &= \operatorname{vec}\frac{1}{n}\sum_{i=1}^{n}(\mathcal{X}_i - \mathfrak{g}_1(\boldsymbol{\eta}_{y_i})),\\
\nabla_{\boldsymbol{\beta}_j} l_n &= \operatorname{vec}\frac{1}{n}\sum_{i=1}^{n}(\mathcal{X}_i - \mathfrak{g}_1(\boldsymbol{\eta}_{y_i}))_{(j)}\Big(\mathcal{F}_{y_i}\bigtimes_{k\in[r]\setminus j}\boldsymbol{\beta}_k\Big)_{(j)}^{T},\\
\nabla_{\boldsymbol{\Omega}_j} l_n &= \operatorname{vec}\frac{c}{n}\sum_{i=1}^{n}(\mathcal{X}_i\otimes\mathcal{X}_i - \mathcal{K}(\mathcal{G}_2(\boldsymbol{\eta}_{y_i})))\bigtimes_{k\in[r]\setminus j}(\operatorname{vec}\boldsymbol{\Omega}_k)^T
\end{aligned}$$

which obtains $\nabla l_n = (\nabla_{\overline{\eta}} l_n, \nabla_{\boldsymbol{\beta}_1} l_n, \ldots, \nabla_{\boldsymbol{\beta}_r} l_n, \nabla_{\boldsymbol{\Omega}_1} l_n, \ldots, \nabla_{\boldsymbol{\Omega}_r} l_n)$. If $\boldsymbol{T}_2$ is the identity matrix $\boldsymbol{I}_{p(p+1)/2}$, then $\mathcal{G}_2(\boldsymbol{\eta}_y) = \mathfrak{g}_2(\boldsymbol{\eta}_y)$.

Proof. We first note that for any exponential family with density (5.1) the term $b(\boldsymbol{\eta}_{y_i})$ differentiated with respect to the natural parameter $\boldsymbol{\eta}_{y_i}$ is the expectation of the statistic $\boldsymbol{t}(\mathcal{X})$ given $Y = y_i$. In our case we get $\nabla_{\boldsymbol{\eta}_{y_i}} b = (\nabla_{\boldsymbol{\eta}_{1y_i}} b, \nabla_{\boldsymbol{\eta}_2} b)$ with components

$$\nabla_{\boldsymbol{\eta}_{1y_i}} b = \mathbb{E}[\boldsymbol{t}_1(\mathcal{X}) \mid Y = y_i] = \operatorname{vec}\mathbb{E}[\mathcal{X} \mid Y = y_i] = \operatorname{vec}\mathfrak{g}_1(\boldsymbol{\eta}_{y_i})$$

and

$$\begin{aligned}
\nabla_{\boldsymbol{\eta}_2} b &= \mathbb{E}[\boldsymbol{t}_2(\mathcal{X}) \mid Y = y_i] = \mathbb{E}[\boldsymbol{T}_2\operatorname{vech}((\operatorname{vec}\mathcal{X})(\operatorname{vec}\mathcal{X})^T) \mid Y = y_i]\\
&= \mathbb{E}[\boldsymbol{T}_2\boldsymbol{D}_p^{\dagger}\operatorname{vec}(\mathcal{X}\circ\mathcal{X}) \mid Y = y_i] = \boldsymbol{T}_2\boldsymbol{D}_p^{\dagger}\operatorname{vec}\mathfrak{g}_2(\boldsymbol{\eta}_{y_i}).
\end{aligned}$$

The gradients are related to their derivatives by transposition, $\nabla_{\boldsymbol{\eta}_{1y_i}} b = \mathrm{D}b(\boldsymbol{\eta}_{1y_i})^T$ and $\nabla_{\boldsymbol{\eta}_2} b = \mathrm{D}b(\boldsymbol{\eta}_2)^T$. Next we provide the differentials of the natural parameter components from (5.6) and (5.7) in a quite direct form, without any further "simplifications", because

the down-stream computations won't benefit from re-expressing the following

$$\begin{aligned}
\mathrm{d}\boldsymbol{\eta}_{1y_i}(\overline{\eta}) &= \mathrm{d}\operatorname{vec}\overline{\eta},\\
\mathrm{d}\boldsymbol{\eta}_{1y_i}(\boldsymbol{\beta}_j) &= \operatorname{vec}\Big(\mathcal{F}_{y_i}\bigtimes_{\substack{k=1\\k\neq j}}^{r}\boldsymbol{\beta}_k\times_j\mathrm{d}\boldsymbol{\beta}_j\Big),\\
\mathrm{d}\boldsymbol{\eta}_2(\boldsymbol{\Omega}_j) &= ((\boldsymbol{T}_2\boldsymbol{D}_p^{\dagger})^{\dagger})^T\operatorname{vec}(c\,\mathrm{d}\boldsymbol{\Omega})\\
&= c((\boldsymbol{T}_2\boldsymbol{D}_p^{\dagger})^{\dagger})^T\operatorname{vec}\Big(\bigotimes_{k=r}^{j+1}\boldsymbol{\Omega}_k\otimes\mathrm{d}\boldsymbol{\Omega}_j\otimes\bigotimes_{l=j-1}^{1}\boldsymbol{\Omega}_l\Big).
\end{aligned}$$

All other combinations, namely $\mathrm{d}\boldsymbol{\eta}_{1y_i}(\boldsymbol{\Omega}_j)$, $\mathrm{d}\boldsymbol{\eta}_2(\overline{\eta})$ and $\mathrm{d}\boldsymbol{\eta}_2(\boldsymbol{\beta}_j)$, are zero. Continuing with the partial differentials of l_n from (5.11)

$$\begin{aligned}
\mathrm{d}l_n(\overline{\eta}) = \sum_{i=1}^{n}(\langle\mathrm{d}\overline{\eta},\mathcal{X}_i\rangle - \mathrm{D}b(\boldsymbol{\eta}_{1y_i})\mathrm{d}\boldsymbol{\eta}_{1y_i}(\overline{\eta})) &= \sum_{i=1}^{n}(\operatorname{vec}\mathcal{X}_i - \operatorname{vec}\mathfrak{g}_1(\boldsymbol{\eta}_{y_i}))^T\mathrm{d}\operatorname{vec}\overline{\eta}\\
&= (\mathrm{d}\operatorname{vec}\overline{\eta})^T\operatorname{vec}\sum_{i=1}^{n}(\mathcal{X}_i - \mathfrak{g}_1(\boldsymbol{\eta}_{y_i})).
\end{aligned}$$

For every $j = 1, ..., r$ we get the differentials

$$\begin{aligned}
\mathrm{d}l_n(\boldsymbol{\beta}_j) &= \sum_{i=1}^{n}\Big(\Big\langle\mathcal{F}_{y_i}\bigtimes_{\substack{k=1\\k\neq j}}^{r}\boldsymbol{\beta}_k\times_j\mathrm{d}\boldsymbol{\beta}_j,\mathcal{X}_i\Big\rangle - \mathrm{D}b(\boldsymbol{\eta}_{1y_i})\mathrm{d}\boldsymbol{\eta}_{1y_i}(\boldsymbol{\beta}_j)\Big)\\
&= \sum_{i=1}^{n}\Big\langle\mathcal{F}_{y_i}\bigtimes_{\substack{k=1\\k\neq j}}^{r}\boldsymbol{\beta}_k\times_j\mathrm{d}\boldsymbol{\beta}_j,\mathcal{X}_i - \mathfrak{g}_1(\boldsymbol{\eta}_{y_i})\Big\rangle\\
&= \sum_{i=1}^{n}\operatorname{tr}\Big(\mathrm{d}\boldsymbol{\beta}_j\Big(\mathcal{F}_{y_i}\bigtimes_{\substack{k=1\\k\neq j}}^{r}\boldsymbol{\beta}_k\Big)_{(j)}(\mathcal{X}_i - \mathfrak{g}_1(\boldsymbol{\eta}_{y_i}))_{(j)}^T\Big)\\
&= (\mathrm{d}\operatorname{vec}\boldsymbol{\beta}_j)^T\operatorname{vec}\sum_{i=1}^{n}(\mathcal{X}_i - \mathfrak{g}_1(\boldsymbol{\eta}_{y_i}))_{(j)}\Big(\mathcal{F}_{y_i}\bigtimes_{\substack{k=1\\k\neq j}}^{r}\boldsymbol{\beta}_k\Big)_{(j)}^T
\end{aligned}$$

as well as

$$\begin{aligned}
\mathrm{d}l_n(\boldsymbol{\Omega}_j) &= \sum_{i=1}^{n}\Big(c\Big\langle \mathcal{X}_i \bigtimes_{\substack{k=1\\k\neq j}}^{r} \boldsymbol{\Omega}_k \times_j \mathrm{d}\boldsymbol{\Omega}_j, \mathcal{X}_i\Big\rangle - \mathrm{D}b(\boldsymbol{\eta}_2)\mathrm{d}\boldsymbol{\eta}_2(\boldsymbol{\Omega}_j)\Big) \\
&= c\sum_{i=1}^{n}\Big(\Big\langle \mathcal{X}_i \bigtimes_{\substack{k=1\\k\neq j}}^{r} \boldsymbol{\Omega}_k \times_j \mathrm{d}\boldsymbol{\Omega}_j, \mathcal{X}_i\Big\rangle - (\boldsymbol{T}_2\boldsymbol{D}_p^{\dagger}\operatorname{vec}\mathfrak{g}_2(\boldsymbol{\eta}_{y_i}))^T((\boldsymbol{T}_2\boldsymbol{D}_p^{\dagger})^{\dagger})^T\operatorname{vec}\Big(\bigotimes_{k=r}^{j+1}\boldsymbol{\Omega}_k \otimes \mathrm{d}\boldsymbol{\Omega}_j \otimes \bigotimes_{l=j-1}^{1}\boldsymbol{\Omega}_l\Big)\Big) \\
&= c\sum_{i=1}^{n}\Big(\Big\langle \mathcal{X}_i \bigtimes_{\substack{k=1\\k\neq j}}^{r} \boldsymbol{\Omega}_k \times_j \mathrm{d}\boldsymbol{\Omega}_j, \mathcal{X}_i\Big\rangle - (\operatorname{vec}\mathcal{G}_2(\boldsymbol{\eta}_{y_i}))^T\operatorname{vec}\Big(\bigotimes_{k=r}^{j+1}\boldsymbol{\Omega}_k \otimes \mathrm{d}\boldsymbol{\Omega}_j \otimes \bigotimes_{l=j-1}^{1}\boldsymbol{\Omega}_l\Big)\Big) \\
&= c\sum_{i=1}^{n}\Big(\operatorname{vec}(\mathcal{X}_i\circ\mathcal{X}_i - \mathcal{G}_2(\boldsymbol{\eta}_{y_i}))^T\operatorname{vec}\Big(\bigotimes_{k=r}^{j+1}\boldsymbol{\Omega}_k \otimes \mathrm{d}\boldsymbol{\Omega}_j \otimes \bigotimes_{l=j-1}^{1}\boldsymbol{\Omega}_l\Big)\Big) \\
&= c\sum_{i=1}^{n}\mathcal{K}(\mathcal{X}_i\circ\mathcal{X}_i - \mathcal{G}_2(\boldsymbol{\eta}_{y_i}))\bigtimes_{\substack{k=1\\k\neq j}}^{r}(\operatorname{vec}\boldsymbol{\Omega}_k)^T \times_j (\mathrm{d}\operatorname{vec}\boldsymbol{\Omega}_j)^T \\
&= c(\mathrm{d}\operatorname{vec}\boldsymbol{\Omega}_j)^T\sum_{i=1}^{n}\Big((\mathcal{X}_i\otimes\mathcal{X}_i - \mathcal{K}(\mathcal{G}_2(\boldsymbol{\eta}_{y_i})))\bigtimes_{\substack{k=1\\k\neq j}}^{r}(\operatorname{vec}\boldsymbol{\Omega}_k)^T\Big)_{(j)} \\
&= c(\mathrm{d}\operatorname{vec}\boldsymbol{\Omega}_j)^T\operatorname{vec}\sum_{i=1}^{n}(\mathcal{X}_i\otimes\mathcal{X}_i - \mathcal{K}(\mathcal{G}_2(\boldsymbol{\eta}_{y_i})))\bigtimes_{\substack{k=1\\k\neq j}}^{r}(\operatorname{vec}\boldsymbol{\Omega}_k)^T
\end{aligned}$$

Now, applying the identity $\mathrm{d}\mathcal{A}(\mathcal{B}) = (\mathrm{d}\operatorname{vec}\mathcal{B})^T\nabla_{\mathcal{B}}\mathcal{A}$ gives the required partial gradients.

Finally, if $\boldsymbol{T}_2$ is the identify matrix, then

$$\operatorname{vec}\mathcal{G}_2(\boldsymbol{\eta}_y) = (\boldsymbol{T}_2\boldsymbol{D}_p^{\dagger})^{\dagger}\boldsymbol{T}_2\boldsymbol{D}_p^{\dagger}\operatorname{vec}\mathfrak{g}_2(\boldsymbol{\eta}_y) = \boldsymbol{D}_p\boldsymbol{D}_p^{\dagger}\operatorname{vec}\mathfrak{g}_2(\boldsymbol{\eta}_y) = \operatorname{vec}\mathfrak{g}_2(\boldsymbol{\eta}_y)$$

where the last equality holds because $\boldsymbol{N}_p = \boldsymbol{D}_p\boldsymbol{D}_p^{\dagger}$ is the symmetrizer matrix from Abadir and Magnus [AM05, Ch. 11]. For the symmetrizer matrix $\boldsymbol{N}_p$ holds $\boldsymbol{N}_p\operatorname{vec}\boldsymbol{A} = \operatorname{vec}\boldsymbol{A}$ if $\boldsymbol{A} = \boldsymbol{A}^T$, while

$$\operatorname{vec}\mathfrak{g}_2(\boldsymbol{\eta}_y) = \operatorname{vec}\mathbb{E}[\mathcal{X}\circ\mathcal{X}\mid Y=y] = \operatorname{vec}\mathbb{E}[(\operatorname{vec}\mathcal{X})(\operatorname{vec}\mathcal{X})^T\mid Y=y]$$

is the vectorization of a symmetric matrix. □

Although the general case of any GMLM model can be fitted via gradient descent using Theorem 6, this may be very inefficient. In Theorem 6, $\boldsymbol{T}_2$ can be used to introduce flexible second moment structures. For example, it allows modeling effects differently for predictor components, as described in Section 5.2.2 after Eqn. (5.20). In the remainder, we focus on $\boldsymbol{T}_2$'s that are identity matrices. This approach simplifies the estimation algorithm and the speed of the numerical calculation in the case of tensor normal predictors. In the case of the tensor normal distribution, an iterative cyclic updating scheme is derived in Section 5.2.1, which has much faster convergence, is stable and does not require hyper parameters, as will be discussed later. On the other hand, the Ising model does not allow such a scheme. There we need to use a gradient based method, which is the subject of Section 5.2.2.

5.2.1 Tensor Normal

The tensor normal, also known as the *multilinear normal*, is the extension of the matrix normal to tensor-valued random variables and a member of the quadratic exponential family (5.3) under (5.7). Dawid [Daw81] and Arnold [Arn81] introduced the term matrix normal and, in particular, Arnold [Arn81] provided several theoretical results, such as its density, moments and conditional distributions of its components. The matrix normal distribution is a bilinear normal distribution; a distribution of a two-way array, each component representing a vector of observations [OAvR13]. Kollo and von Rosen [KvR05], Hoff [Hof11], and Ohlson, Ahmad, and von Rosen [OAvR13] presented the extension of the bilinear to the multilinear normal distribution, what we call tensor normal, using a parallel extension of bilinear matrices to multilinear tensors [Com09].

The defining feature of the matrix normal distribution, and its tensor extension, is the Kronecker product structure of its covariance. This formulation, where the covariates are multivariate normal with multiway covariance structure modeled as a Kronecker product of matrices of much lower dimension, aims to overcome the significant modeling and computational challenges arising from the high computational complexity of manipulating tensor representations [see, e.g., HL13; WSS+22].

Multilinear tensor normal models have been used in various applications, including medical imaging [BP07; DKZ09], spatio-temporal data analysis [GH14], regression analysis for longitudinal relational data [Hof15]. One of the most important uses of the multilinear normal (MLN) distribution, and hence tensor analysis, is perhaps in magnetic resonance imaging (MRI) [OAvR13]. A recent survey [WSS+22] and references therein contain more information and potential applications of multilinear tensor normal models.

Suppose $\mathcal{X} \mid Y = y$ follows a tensor normal distribution with mean μ_y and covariance $\mathbf{\Sigma} = \bigotimes_{k=r}^{1} \mathbf{\Sigma}_k$. We assume the distribution is non-degenerate which means that the covariances $\mathbf{\Sigma}_k$ are symmetric positive definite matrices. Its density is given by

$$f_{\boldsymbol{\theta}}(\mathcal{X} \mid Y = y) = (2\pi)^{-p/2} \prod_{k=1}^{r} \det(\mathbf{\Sigma}_k)^{-p/2p_k} \exp\left(-\frac{1}{2} \left\langle \mathcal{X} - \mu_y, (\mathcal{X} - \mu_y) \bigtimes_{k=1}^{r} \mathbf{\Sigma}_k^{-1} \right\rangle \right).$$

For the sake of simplicity and w.l.o.g., we assume $\mathcal{X}$ has 0 marginal expectation; i.e., $\mathbb{E}\,\mathcal{X} = 0$. Rewriting this in the quadratic exponential family form (5.3), determines the scaling constant $c = -1/2$. The relation to the GMLM parameters $\overline{\eta}, \boldsymbol{\beta}_k$ and $\mathbf{\Omega}_k$, for $k = 1, \ldots, r$ is

$$\mu_y = \mathcal{F}_y \bigtimes_{k=1}^{r} \mathbf{\Omega}_k^{-1} \boldsymbol{\beta}_k, \qquad \mathbf{\Omega}_k = \mathbf{\Sigma}_k^{-1}, \tag{5.13}$$

where we used that $\overline{\eta} = 0$ due to $0 = \mathbb{E}\,\mathcal{X} = \mathbb{E}\,\mathbb{E}[\mathcal{X} \mid Y] = \mathbb{E}\,\mu_Y$ in combination with $\mathbb{E}\,\mathcal{F}_Y = 0$. Additionally, all the $\mathbf{\Omega}_k$'s are symmetric positive definite, because the $\mathbf{\Sigma}_k$'s are. This lead to another simplification since then $\boldsymbol{T}_2$ in (5.2) equals the identity. This also means that the gradients of the log-likelihood l_n in Theorem 6 are simpler. We obtain

$$\mathfrak{g}_1(\boldsymbol{\eta}_y) = \mathbb{E}[\mathcal{X} \mid Y = y] = \mu_y,$$

$$\mathcal{G}_2(\boldsymbol{\eta}_y) = \mathfrak{g}_2(\boldsymbol{\eta}_y) = \mathbb{E}[\mathcal{X} \circ \mathcal{X} \mid Y = y] \equiv \bigotimes_{k=r}^{1} \mathbf{\Sigma}_k + (\operatorname{vec} \mu_y)(\operatorname{vec} \mu_y)^T.$$

In practice, we assume we have a random sample of n observations $(\mathcal{X}_i, \mathcal{F}_{y_i})$ from the joint distribution. We start the estimation process by demeaning them. Then, only the reduction matrices $\boldsymbol{\beta}_k$ and the scatter matrices $\boldsymbol{\Omega}_k$ need to be estimated. To solve the optimization problem (5.12), with $\overline{\eta} = 0$ we initialize the parameters using a simple heuristic approach. First, we compute moment based mode-wise marginal covariance estimates $\widehat{\boldsymbol{\Sigma}}_k(\mathcal{X})$ and $\widehat{\boldsymbol{\Sigma}}_k(\mathcal{F}_Y)$ as

$$\widehat{\boldsymbol{\Sigma}}_k(\mathcal{X}) = \frac{1}{n}\sum_{i=1}^{n}(\mathcal{X}_i)_{(k)}(\mathcal{X}_i)_{(k)}^T, \qquad \widehat{\boldsymbol{\Sigma}}_k(\mathcal{F}_Y) = \frac{1}{n}\sum_{i=1}^{n}(\mathcal{F}_{y_i})_{(k)}(\mathcal{F}_{y_i})_{(k)}^T.$$

From those we compute for every mode $k = 1, \ldots, r$ the first $j = 1, \ldots, q_k$ eigenvectors $\boldsymbol{v}_j(\widehat{\boldsymbol{\Sigma}}_k(\mathcal{X}))$, $\boldsymbol{v}_j(\widehat{\boldsymbol{\Sigma}}_k(\mathcal{F}_Y))$ and eigenvalues $\lambda_j(\widehat{\boldsymbol{\Sigma}}_k(\mathcal{X}))$, $\lambda_j(\widehat{\boldsymbol{\Sigma}}_k(\mathcal{X}))$ of the marginal covariance estimates. We set

$$\begin{aligned}
\boldsymbol{U}_k &= (\boldsymbol{v}_1(\widehat{\boldsymbol{\Sigma}}_1(\mathcal{X})), \ldots, \boldsymbol{v}_{q_k}(\widehat{\boldsymbol{\Sigma}}_{q_k}(\mathcal{X}))),\\
\boldsymbol{D}_k &= \operatorname{diag}(\boldsymbol{v}_1(\widehat{\boldsymbol{\Sigma}}_1(\mathcal{X}))\boldsymbol{v}_1(\widehat{\boldsymbol{\Sigma}}_1(\mathcal{F}_Y)), \ldots, \boldsymbol{v}_{q_k}(\widehat{\boldsymbol{\Sigma}}_{q_k}(\mathcal{X}))\boldsymbol{v}_{q_k}(\widehat{\boldsymbol{\Sigma}}_k(\mathcal{F}_Y))),\\
\boldsymbol{V}_k &= (\boldsymbol{v}_1(\widehat{\boldsymbol{\Sigma}}_1(\mathcal{F}_Y), \ldots, \boldsymbol{v}_{q_k}(\widehat{\boldsymbol{\Sigma}}_{q_k}(\mathcal{F}_Y)).
\end{aligned}$$

The initial value of $\boldsymbol{\beta}_k$ is

$$\widehat{\boldsymbol{\beta}}_k^{(0)} = \boldsymbol{U}_k\sqrt{\boldsymbol{D}_k}\boldsymbol{V}_k^T, \tag{5.14}$$

and the initial value of $\boldsymbol{\Omega}_k$ is set to the identity $\boldsymbol{\Omega}_k^{(0)} = \boldsymbol{I}_{p_k}$, for $k = 1, \ldots, r$.

Given $\widehat{\boldsymbol{\beta}}_1, \ldots, \widehat{\boldsymbol{\beta}}_r, \widehat{\boldsymbol{\Omega}}_1, \ldots, \widehat{\boldsymbol{\Omega}}_r$, we take the gradient $\nabla_{\boldsymbol{\beta}_j} l_n$ of the tensor normal log likelihood l_n in (5.11) applying Theorem 6 and keep all other parameters except $\boldsymbol{\beta}_j$ fixed. Then, $\nabla_{\boldsymbol{\beta}_j} l_n = 0$ has the closed form solution

$$\boldsymbol{\beta}_j^T = \Big(\sum_{i=1}^{n}\Big(\mathcal{F}_{y_i}\bigtimes_{k\neq j}\widehat{\boldsymbol{\Omega}}_k^{-1}\widehat{\boldsymbol{\beta}}_k\Big)_{(j)}\Big(\mathcal{F}_{y_i}\bigtimes_{k\neq j}\widehat{\boldsymbol{\beta}}_k\Big)_{(j)}^T\Big)^{-1}\Big(\sum_{i=1}^{n}\Big(\mathcal{F}_{y_i}\bigtimes_{k\neq j}\widehat{\boldsymbol{\beta}}_k\Big)_{(j)}(\mathcal{X}_i)_{(j)}^T\Big)\widehat{\boldsymbol{\Omega}}_j. \tag{5.15}$$

Equating the partial gradient of the jth scatter matrix $\boldsymbol{\Omega}_j$ in Theorem 6 to zero ($\nabla_{\boldsymbol{\Omega}_j} l_n = 0$) gives a quadratic matrix equation. This is due to the dependence of μ_y on $\boldsymbol{\Omega}_j$. In practice though, it is faster, more stable, and equally accurate to use mode-wise covariance estimates via the residuals

$$\widehat{\mathcal{R}}_i = \mathcal{X}_i - \widehat{\mu}_{y_i} = \mathcal{X}_i - \mathcal{F}_{y_i}\bigtimes_{k=1}^{r}\widehat{\boldsymbol{\Omega}}_k^{-1}\widehat{\boldsymbol{\beta}}_k. \tag{5.16}$$

The estimates are computed via

$$\tilde{\boldsymbol{\Sigma}}_j = \sum_{i=1}^{n}(\widehat{\mathcal{R}}_i)_{(j)}(\widehat{\mathcal{R}}_i)_{(j)}^T,$$

where $\tilde{s}\tilde{\boldsymbol{\Sigma}}_j = \widehat{\boldsymbol{\Omega}}_j^{-1}$. For scaling we use that the mean squared error has to be equal to the trace of the covariance estimate,

$$\frac{1}{n}\sum_{i=1}^{n}\langle\widehat{\mathcal{R}}_i, \widehat{\mathcal{R}}_i\rangle = \operatorname{tr}\bigotimes_{k=r}^{1}\widehat{\boldsymbol{\Omega}}_k^{-1} = \prod_{k=1}^{r}\operatorname{tr}\widehat{\boldsymbol{\Omega}}_k^{-1} = \tilde{s}^r\prod_{k=1}^{r}\operatorname{tr}\tilde{\boldsymbol{\Sigma}}_k,$$

so that

$$\tilde{s} = \left(\left(\prod_{k=1}^{r} \operatorname{tr} \tilde{\boldsymbol{\Sigma}}_k \right)^{-1} \frac{1}{n} \sum_{i=1}^{n} \langle \widehat{\mathcal{R}}_i, \widehat{\mathcal{R}}_i \rangle \right)^{1/r} \tag{5.17}$$

resulting in the estimates $\widehat{\boldsymbol{\Omega}}_j = (\tilde{s}\tilde{\boldsymbol{\Sigma}}_j)^{-1}$. Estimation is then performed by updating the estimates $\widehat{\boldsymbol{\beta}}_j$ via (5.15) for $j = 1, \ldots, r$, and then recompute the $\widehat{\boldsymbol{\Omega}}_j$ estimates simultaneously keeping the $\widehat{\boldsymbol{\beta}}_j$'s fixed. This procedure is repeated until convergence.

A technical detail for numerical stability is to ensure that the scaled values $\tilde{s}\tilde{\boldsymbol{\Sigma}}_j$, assumed to be symmetric and positive definite, are well conditioned. Thus, we estimate the condition number of $\tilde{s}\tilde{\boldsymbol{\Sigma}}_j$ prior to computing the inverse. In case of ill- conditioning, we use the regularized $\widehat{\boldsymbol{\Omega}}_j = (\tilde{s}\tilde{\boldsymbol{\Sigma}}_j + 0.2\lambda_1(\tilde{s}\tilde{\boldsymbol{\Sigma}}_j)\boldsymbol{I}_{p_j})^{-1}$ instead, where $\lambda_1(\tilde{s}\tilde{\boldsymbol{\Sigma}}_j)$ is the first (maximum) eigenvalue. Experiments showed that this regularization is usually only required in the first few iterations.

Furthermore, if the parameter space follows a more general setting as in Theorem 9, updating may produces estimates outside the parameter space. A simple and efficient method is to project every updated estimate onto the corresponding manifold.

A standard method to compute the MLE of a Kronecker product is block-coordinate descent, also referred to as the "flip-flop algorithm." This algorithm was proposed independently by Mardia and Goodall [MG93] and Dutilleul [Dut99] and was later called "flip-flop" algorithm by Lu and Zimmerman [LZ05] for the computation of the maximum likelihood estimators of the components of a separable covariance matrix. Manceur and Dutilleul [MD13] extended the "flip-flop" algorithm for the computation of the MLE of the separable covariance structure of a 3-way and 4-way normal distribution and obtained a lower bound for the sample size required for its existence. The same issue was also studied by Drton, Kuriki, and Hoff [DKH20] in the case of a two-way array (matrix). Our algorithm uses a similar "flip-flop" approach by iteratively updating the $\boldsymbol{\beta}_k$'s and $\boldsymbol{\Omega}_k$'s, one after the other.

5.2.1.1 Implementation in R

An by itself nice function is to compute an per-mode Kronecker decomposed covariance estimate. It assumes an tensor valued data set `X` with the last axis indexing observations. Based on Equation (5.16) with scaling (5.17) which is applicable in general (not only the residuals as in the reference equations). The `center` argument allows to avoid the recentering of an already centered argument.

```
mcov <- function(X, center = TRUE) {
    dimX <- head(dim(X), -1)

    if (center) {
        X <- X - as.vector(rowMeans(X, dims = length(dimX)))
    }

    tr.Sigma <- prod(dimX) * mean(X^2)
    lapply(seq_along(dimX), function(j) {
        Sigma <- var(t(mat(X, j)))
        (tr.Sigma / sum(diag(Sigma)))^(1 / length(dimX)) * Sigma
    })
}
```

Assuming the predictors $\mathcal{X}_i$ are provided in the form of an $p_1 \times \ldots \times p_r \times n$ dimensional array with the last axis of size n indexing the observations. The predictor functions $\mathcal{F}_{y_i}$ are passed as a $q_1 \times \ldots \times q_r \times n$ dimensional array. The parameter `proj.betas` and `proj.Omegas` are lists of functions which project their argument to the corresponding matrix manifold as described in Sections 5.3.1 and 5.3.2. An implementation of GMLM for tensor normal data is then;

```
gmlm.tensor.normal <- function(X, F, max.iter = 100,
    proj.betas = vector("list", length(dim(X))),
    proj.Omegas = vector("list", length(dim(X))),
    cond.threshold = 25, eps = 1e-6
) {
    dimX <- head(dim(X), -1)
    dimF <- head(dim(F), -1)
    modes <- seq_along(dimX)

    # Centering X ← X − E X and F ← F − E F
    X <- X - as.vector(rowMeans(X, dims = length(dimX)))
    F <- F - as.vector(rowMeans(F, dims = length(dimF)))

    # initialize Σ_k = Ω_k^{-1} = I_{p_k} with log(det(Ω)) = 0 and β_k's as in (5.14)
    Sigmas <- Map(diag, dimX)
    Omegas <- Map(diag, dimX)
    log.det.Omega <- 0
    dirsX <- Map(function(Sigma) {
        with(La.svd(Sigma, nu = 0), sqrt(d) * vt)
    }, mcov(X, sample.axis, center = FALSE))
    dirsF <- Map(function(Sigma) {
        with(La.svd(Sigma, nu = 0), sqrt(d) * vt)
    }, mcov(F, sample.axis, center = FALSE))
    betas <- betas.init <- Map(function(dX, dF) {
        s <- min(ncol(dX), nrow(dF))
        crossprod(dX[1:s, , drop = FALSE], dF[1:s, , drop = FALSE])
    }, dirsX, dirsF)

    # loss (optim. objective, neg. log likelihood)
    R <- X - mlm(F, Map(`%*%`, Sigmas, betas))
    loss <- mean(R * mlm(R, Omegas)) - log.det.Omega

    # Iterative block coordinate descent (flip-flop type)
    for (iter in seq_len(max.iter)) {
        # Update betas (5.15)
        for (j in seq_along(betas)) {
            FxB_j <- mlm(F, betas[-j], modes[-j])
            FxSB_j <- mlm(FxB_j, Sigmas[-j], modes[-j])
            betas[[j]] <- crossprod(Omegas[[j]], solve(
                tcrossprod(mat(FxSB_j, j), mat(FxB_j, j)),
                tcrossprod(mat(FxB_j, j), mat(X, j))
            ))
            # Project betas onto their manifold
            if (is.function(proj_j <- proj.betas[[j]])) {
                betas[[j]] <- proj_j(betas[[j]])
            }
        }
```

```
48
49          # Update Omegas (with regularization if needed)
50          R <- X - mlm(F, Map('%*%', Sigmas, betas))
51          Sigmas <- mcov(R, center = FALSE)
52          for (j in seq_along(Sigmas)) {
53              min_max <- range(eigen(Sigmas[[j]], TRUE, TRUE)$values)
54              if (min_max[2] > cond.threshold * min_max[1]) {
55                  diag(Sigmas[[j]]) <- diag(Sigmas[[j]]) + 0.2 * min_max[2]
56              }
57              Omegas[[j]] <- solve(Sigmas[[j]])
58              # Project Omegas onto their manifold
59              if (is.function(proj_j <- proj.Omegas[[j]])) {
60                  Omegas[[j]] <- proj_j(Omegas[[j]])
61                  Sigmas[[j]] <- solve(Omegas[[j]])
62              }
63          }
64
65          # Numerically more stable version for log(det(Ω))
66          log.det.Omega <- sum(mapply(function(Omega) {
67              sum(log(eigen(Omega, TRUE, TRUE)$values))
68          }, Omegas) / dimX)
69          # Update current loss
70          loss.last <- loss
71          loss <- mean(R * mlm(R, Omegas)) - log.det.Omega
72
73          # check break consition
74          if (abs(loss.last - loss) < eps * abs(loss.last)) {
75              break
76          }
77      }
78
79      betas
80  }
```

5.2.2 Ising Model

The Ising[1] model [Len20; Isi25; Nis05] is a mathematical model originating in statistical physics to study ferromagnetism in a thermodynamic setting. It describes magentic dipoles (atomic "spins") which can take two states (± 1) while allowing two-way interactions between direct neighbours on a lattice, a discrete grid. The model assumes all elementary magnets to be the same, which translates to all having the same coupling strength (two-way interactions) governed by a single parameter relating to the temperature of the system. Nowadays, the Ising model, in its general form, allows for different coupling strength for every (symmetric) interaction as well as an external magnetic field acting on every magnetic dipole separately. A modern review is given by Nguyen, Zecchina, and Berg [NZB17].

In statistics, the Ising model is used to model multivariate binary data. That is, the states are $0, 1$ instead of ± 1. It is related to a multitude of other models; *Graphical Models* and *Markov Random Fields* to describe conditional dependence [Lau96; WJ08; LR02], *Potts*

[1] Also known as the *Lenz-Ising* model as the physical assumptions of the model where developed by both Lenz and Ising [Nis05]. Ising gave a closed form solution for the 1-dimensional lattice, that is, a linear chain [Isi25].

models [Bes74; CKG22] which generalize the Ising model to multiple states, the *multivariate Bernoulli distribution* [Whi90; JKB97; DDW13] considering also interactions (tree-way and higher), to give the most prominent.

The p-dimensional Ising model is a discrete probability distribution on the set of p-dimensional binary vectors $\boldsymbol{x} \in \{0,1\}^p$ with pmf given by

$$P_{\boldsymbol{\gamma}}(\boldsymbol{x}) = p_0(\boldsymbol{\gamma}) \exp(\operatorname{vech}(\boldsymbol{x}\boldsymbol{x}^T)^T \boldsymbol{\gamma}).$$

The scaling factor $p_0(\boldsymbol{\gamma}) \in \mathbb{R}_+$ ensures that $P_{\boldsymbol{\gamma}}$ is a pmf. It is equal to the probability of the zero event $P(X = \mathbf{0}) = p_0(\boldsymbol{\gamma})$. More commonly known as the *partition function*, the reciprocal of p_0, is given by

$$p_0(\boldsymbol{\gamma})^{-1} = \sum_{\boldsymbol{x} \in \{0,1\}^p} \exp(\operatorname{vech}(\boldsymbol{x}\boldsymbol{x}^T)^T \boldsymbol{\gamma}). \tag{5.18}$$

By an abuse of notation, we let $\boldsymbol{\gamma}_{jl}$ denote the element of $\boldsymbol{\gamma}$ corresponding to $\boldsymbol{x}_j \boldsymbol{x}_l$ in $\operatorname{vech}(\boldsymbol{x}\boldsymbol{x}^T)$[2]. The "diagonal" parameter $\boldsymbol{\gamma}_{jj}$ expresses the conditional log odds of $X_j = 1 \mid X_{-j} = \mathbf{0}$, where the negative subscript in X_{-j} describes the $p-1$ dimensional vector X with the jth element removed. The off diagonal entries $\boldsymbol{\gamma}_{jl}$, $j \neq l$, are equal to the conditional log odds of simultaneous occurrence $X_j = 1, X_l = 1 \mid X_{-j,-l} = \mathbf{0}$. More precisely, the conditional probabilities and the natural parameters are related via

$$\begin{aligned}
\boldsymbol{\gamma}_{jj} &= \log \frac{P_{\boldsymbol{\gamma}}(X_j = 1 \mid X_{-j} = \mathbf{0})}{1 - P_{\boldsymbol{\gamma}}(X_j = 1 \mid X_{-j} = \mathbf{0})}, \\
\boldsymbol{\gamma}_{jl} &= \log \frac{1 - P_{\boldsymbol{\gamma}}(X_j = 1 \mid X_{-j} = \mathbf{0}) P_{\boldsymbol{\gamma}}(X_l = 1 \mid X_{-l} = \mathbf{0})}{P_{\boldsymbol{\gamma}}(X_j = 1 \mid X_{-j} = \mathbf{0}) P_{\boldsymbol{\gamma}}(X_l = 1 \mid X_{-l} = \mathbf{0})} \frac{P_{\boldsymbol{\gamma}}(X_j = 1, X_l = 1 \mid X_{-j,-l} = \mathbf{0})}{1 - P_{\boldsymbol{\gamma}}(X_j = 1, X_l = 1 \mid X_{-j,-l} = \mathbf{0})}.
\end{aligned} \tag{5.19}$$

Conditional Ising models, incorporating the information of covariates Y into the model, were considered by Cheng et al. [CLW+14] and Bura et al. [BFG+22]. The direct way is to parameterize $\boldsymbol{\gamma} = \boldsymbol{\gamma}_y$ by the covariate $Y = y$ to model a conditional distribution $P_{\boldsymbol{\gamma}_y}(\boldsymbol{x} \mid Y = y)$.

We extend the conditional pmf by allowing the binary variables to be tensor valued; that is, we set $\boldsymbol{x} = \operatorname{vec} \mathcal{X}$, with dimension $p = \prod_{k=1}^r p_k$ for $\mathcal{X} \in \{0,1\}^{p_1 \times \cdots \times p_r}$. The tensor structure of $\mathcal{X}$ is accommodated by assuming Kronecker product constraints to the parameter vector $\boldsymbol{\gamma}_y$ in a similar fashion as for the tensor normal model. This means that we compare the pmf $P_{\boldsymbol{\gamma}_y}(\operatorname{vec} \mathcal{X} | Y = y)$ with the quadratic exponential family (5.3) with the natural parameters modeled by (5.6) and (5.7). A detail to be considered is that the diagonal of $(\operatorname{vec} \mathcal{X})(\operatorname{vec} \mathcal{X})^T$ is equal to $\operatorname{vec} \mathcal{X}$. This gives the GMLM model as

$$\begin{aligned}
P_{\boldsymbol{\gamma}_y}(\mathcal{X} \mid Y = y) &= p_0(\boldsymbol{\gamma}_y) \exp(\operatorname{vech}((\operatorname{vec} \mathcal{X})(\operatorname{vec} \mathcal{X})^T)^T \boldsymbol{\gamma}_y) \\
&= p_0(\boldsymbol{\gamma}_y) \exp\Big(\Big\langle \mathcal{X}, \mathcal{F}_y \bigtimes_{k=1}^r \boldsymbol{\beta}_k \Big\rangle + \Big\langle \mathcal{X} \bigtimes_{k=1}^r \boldsymbol{\Omega}_k, \mathcal{X} \Big\rangle \Big)
\end{aligned} \tag{5.20}$$

[2] Specifically, the element $\boldsymbol{\gamma}_{jl}$ of $\boldsymbol{\gamma}$ is a short hand for $\boldsymbol{\gamma}_{\iota(j,l)}$ with $\iota(j,l) = (\min(j,l)-1)(2p - \min(j,l))/2 + \max(j,l)$ mapping the matrix row index j and column index l to the corresponding half vectorization indices $\iota(j,l)$.

where we set $\overline{\eta} = 0$ and $\boldsymbol{T}_2$ to the identity. This imposes an additional constraint to the model, the reason is that the diagonal elements of $\boldsymbol{\Omega} = \bigotimes_{k=r}^{1} \boldsymbol{\Omega}_k$ take the role of $\overline{\eta}$, although not fully. Having the diagonal of $\boldsymbol{\Omega}$ and $\overline{\eta}$ handling the self interaction effects might lead to interference in the optimization routine. Another approach would be to use the $\boldsymbol{T}_2$ matrix to set the corresponding diagonal elements of $\boldsymbol{\Omega}$ to zero and let $\overline{\eta}$ handle the self interaction effect. All of those approaches, namely setting $\overline{\eta} = 0$, keeping $\overline{\eta}$ or using $\boldsymbol{T}_2$, are theoretically solid and compatible with Theorems 6, 9 and 10, assuming all axis dimensions p_k are non-degenerate, that is $p_k > 1$ for all $k = 1, \ldots, r$. Regardless, under our modeling choice the relation between the natural parameters $\boldsymbol{\gamma}_y$ of the conditional Ising model and the GMLM parameters $\boldsymbol{\beta}_k$ and $\boldsymbol{\Omega}_k$ is

$$\boldsymbol{\gamma}_y = \boldsymbol{D}_p^T \operatorname{vec}(\boldsymbol{\Omega} + \operatorname{diag}(\boldsymbol{B} \operatorname{vec} \mathcal{F}_y)) = \boldsymbol{D}_p^T \operatorname{vec}\left(\bigotimes_{k=r}^{1} \boldsymbol{\Omega}_k + \operatorname{diag}\left(\operatorname{vec}\left(\mathcal{F}_y \bigtimes_{k=1}^{r} \boldsymbol{\beta}_k \right) \right) \right). \quad (5.21)$$

In contract to the tensor normal GMLM, the matrices $\boldsymbol{\Omega}_k$ are only required to be symmetric. More specifically, we require $\boldsymbol{\Omega}_k$, for $k = 1, \ldots, r$, to be elements of an embedded submanifold of $\operatorname{Sym}^{p_k \times p_k}$ (see: Sections 5.3.1 and 5.3.2). The mode wise reduction matrices $\boldsymbol{\beta}_k$ are elements of an embedded submanifold of $\mathbb{R}^{p_k \times q_k}$. Common choices are listed in Section 5.3.1.

To solve the optimization problem (5.12), given a data set $(\mathcal{X}_i, y_i)$, for $i = 1, \ldots, n$, we use a variation of gradient descent.

5.2.2.1 Initial Values

The first step is to get reasonable starting values. Experiments showed that a good starting value of $\boldsymbol{\beta}_k$, for $k = 1, \ldots, r$, it to use the tensor normal estimates from Section 5.2.1, considering $\mathcal{X}_i$ as continuous. For initial values of $\boldsymbol{\Omega}_k$, a different approach is required. Setting everything to the uninformed initial value, that is $\boldsymbol{\Omega}_k = \boldsymbol{0}$ as this corresponds to the conditional log odds to be $1 : 1$ for every component and pairwaide interaction. This is not possible, since $\boldsymbol{0}$ is a stationary point of the log-likelihood. This is directly observed by taking a look at the partial gradients of the log-likelihood in Theorem 6. Instead, we use a crude heuristic which threads every mode seperately and ignores any relation to the covariates. It is computationaly cheap and better than any of the alternatives we considered. For every $k = 1, \ldots, r$, let the k'th mode second moment estimate be

$$\widehat{\boldsymbol{M}}_{2(k)} = \frac{p_k}{np} \sum_{i=1}^{n} (\mathcal{X}_i)_{(k)} (\mathcal{X}_i)_{(k)}^T \quad (5.22)$$

which contains the k'th mode first moment estimate in its diagonal $\widehat{\boldsymbol{M}}_{1(k)} = \operatorname{diag} \widehat{\boldsymbol{M}}_{2(k)}$. Considering every column of the matricized observation $(\mathcal{X}_i)_{(k)}$ as a p_k dimensional observation itself. The number of those artificially generated observations is $n \prod_{j \neq k} p_j$. Let Z_k denote the random variable those artificial observations are realization of. Then, we can interpret the elements $(\widehat{\boldsymbol{M}}_{1(k)})_j$ as the estimates of the probability $P((Z_k)_j = 1)$, that is the marginal probability of the jth element of Z_k being 1. Similar, for $l \neq j$ we have $(\widehat{\boldsymbol{M}}_{2(k)})_{jl}$ estimating $P((Z_k)_j = 1, (Z_k)_l = 1)$, the marginal probability of two-way

interactions. Now, we set the diagonal elements of $\mathbf{\Omega}_k$ to zero. For the off diagonal elements of $\mathbf{\Omega}_k$, we equate the conditional probabilities $P((Z_k)_j = 1 \mid (Z_k)_{-j} = \mathbf{0})$ and $P((Z_k)_j = 1, (Z_k)_l = 1 \mid (Z_k)_{-j,-l} = \mathbf{0})$ with the marginal probability estimates $(\widehat{\boldsymbol{M}}_{1(k)})_j$ and $(\widehat{\boldsymbol{M}}_{2(k)})_{jl}$, respectively. Use (5.19) then gives the initial estimates $\widehat{\mathbf{\Omega}}_k^{(0)}$, with $j \neq l$ component wise

$$(\widehat{\mathbf{\Omega}}_k^{(0)})_{jj} = 0, \qquad (\widehat{\mathbf{\Omega}}_k^{(0)})_{jl} = \log \frac{1 - (\widehat{\boldsymbol{M}}_{1(k)})_j (\widehat{\boldsymbol{M}}_{1(k)})_l}{(\widehat{\boldsymbol{M}}_{1(k)})_j (\widehat{\boldsymbol{M}}_{1(k)})_l} \frac{(\widehat{\boldsymbol{M}}_{2(k)})_{jl}}{1 - (\widehat{\boldsymbol{M}}_{2(k)})_{jl}}. \tag{5.23}$$

5.2.2.2 Gradient Optimization

Given initial values, the gradients provided by Theorem 6 can be evaluated for the Ising model. The first step therefore is to determine the values of the inverse link components $\mathfrak{g}_1(\boldsymbol{\gamma}_y) = \mathbb{E}[\mathcal{X} \mid Y = y]$ and $\mathcal{G}_2(\boldsymbol{\gamma}_y) = \mathfrak{g}_2(\boldsymbol{\gamma}_y) = \mathbb{E}[\mathcal{X} \circ \mathcal{X} \mid Y = y]$. An immediate simplification is that the first moment is a part of the second moment. Its values are determined via $\operatorname{vec}(\mathbb{E}[\mathcal{X} \mid Y = y]) = \operatorname{diag}(\mathbb{E}[\mathcal{X} \circ \mathcal{X} \mid Y = y]_{(1,\dots,r)})$. This means only the second moment needs to be computed, or estimated (see: Section 5.2.2.4) in the case of slightly bigger p. For the Ising model, the conditional second moment with parameters $\boldsymbol{\gamma}_y$ is given by the matricized relation

$$\mathfrak{g}_2(\gamma_y)_{(1,\dots,r)} = \mathbb{E}[(\operatorname{vec}\mathcal{X})(\operatorname{vec}\mathcal{X})^T \mid Y = y] = p_0(\boldsymbol{\gamma}_y) \sum_{\boldsymbol{x} \in \{0,1\}^p} \boldsymbol{x}\boldsymbol{x}^T \exp(\operatorname{vech}(\boldsymbol{x}\boldsymbol{x}^T)^T \boldsymbol{\gamma}_y). \tag{5.24}$$

The natural parameter $\boldsymbol{\gamma}_y$ is evaluated via (5.21) enabling us to compute the partial gradients of the log-likelihood l_n (5.11) for the Ising model by Theorem 6 for the GMLM parameters $\boldsymbol{\beta}_k$ and $\mathbf{\Omega}_k$, $k = 1, \dots, r$, at the current iterate $\boldsymbol{\theta}^{(I)} = (\boldsymbol{\beta}_1^{(I)}, \dots, \boldsymbol{\beta}_r^{(I)}, \mathbf{\Omega}_1^{(I)}, \dots, \mathbf{\Omega}_r^{(I)})$. Using classic gradient ascent for maximizing the log-likelihood, we have to specify a learning rate $\lambda \in \mathbb{R}_+$, usualy something like 10^{-3}. The update rule is

$$\boldsymbol{\theta}^{(I+1)} = \boldsymbol{\theta}^{(I)} + \lambda \nabla_{\boldsymbol{\theta}} l_n(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(I)}},$$

which is iterated till convergence. In practice, iteration is performed until ether a maximum number of iterations is exhausted and/or some break condition is satisfied. A proper choice of the learning rate is needed as a too large learning rate λ causes instabilities, while a too low learning rate requires an enormous amount of iterations. Generically, there are two approach against the need to determine a proper learning rate. First, *line search methods* determin an appropriate step size for every iteration. This works great if the evaluation of the object function (the log-likelihood) is cheap. This is not the case in our setting, see Section 5.2.2.4. The second approach is an *adaptive learning rate*. The basic idea is to track specific statistics while optimizing and dynamically adapt the leaning rate via well tested heuristics using the gathered knowledge from past iterations. We opted to use an adaptive leaning rate approach, this not only removes the need to determine an appropriate leaning rate, but also accelerates learning.

Our method of choise is RMSprop [Hin12] as outlined in Section 3.1.1 in the context of

neural networks[3]. According to our experiments, RMSprop requires in the range of 50 till 1000 iterations till convergence while gradient ascent with a learning rate of 10^{-3} is in the range of 1000 till 10000.

5.2.2.3 Small Data Sets

In case of a finite number of observations, specifically in data sets with a small number of observations n, the situation where one components is always ether zero or one can occur. Its also possible to observe two exclusive components. This situation of a "degenerate" data set needs to be safeguarded against in practice. Working with parameters on a log-scale, this gives estimates of $\pm\infty$. This is outside of the parameter space and breaks our optimization algorithm.

The first situation where this needs to be addressed is in (5.23), where we set initial estimates for $\boldsymbol{\Omega}_k$. To avoid divition by zero as well as evaluating the log of zero, we addapt (5.22), the mode wise moment estimates $\widehat{\boldsymbol{M}}_{2(k)}$. A simple method is to replace the "degenerate" components, that are entries with value 0 or 1, with the smallest positive estimate of exactly one occurrence p_k/np, or all but one occurrence $1 - p_k/np$, respectively.

The same problem is present in gradient optimization. Therefore, before starting the optimization, we detect degenerate combinations. We compute upper and lower bounds for the "degenerate" element in the Kronecker product $\widehat{\boldsymbol{\Omega}} = \bigotimes_{k=r}^{1} \widehat{\boldsymbol{\Omega}}_k$. After every gradient update, we check if any of the "degenerate" elements fall outside of the bounds. In that case, we adjust all the elements of the Kronecker component estimates $\widehat{\boldsymbol{\Omega}}_k$, corresponding to the "degenerate" element of their Kronecker product, to fall inside the precomputed bounds. While doing so, we try to alter every component as little as possible to ensure that the non-degenerate elements in $\widehat{\boldsymbol{\Omega}}$, effected by this change due to its Kronecker structure, are altered as little as possible. The exact details are technically cumbersome while providing little insight.

5.2.2.4 Slightly Bigger Dimensions

A big challenge for the Ising model is its high computational complexity as it involves summing over all binary vectors of length $p = \prod_{k=1}^{r} p_k$ in the partition function (5.18). Computing the partition function exactly requires to sum all 2^p binary vectors. For small dimensions, say $p \approx 10$, this is easily computed. Increasing the dimension beyond 20 becomes extremely expensive while it is impossible for dimension bigger than 30. Trying to avoid the evaluation of the log-likelihood and only computing its partial gradients via Theorem 6 does not resolve the issue. The gradients require the inverse link, in other words the second moment (5.24), where, if dropping the scaling factor p_0, still involves to sum 2^p summands. Basically, with our model, this means that the optimization of the Ising model using exactly computed gradients is impossible for moderately sized problems.

For estimation of dimensions p bigger than 20, we use a Monte-Carlo method to estimate the second moment (5.24), required to compute the partial gradients of the log-likelihood.

[3]The optimization algorithm RMSprop as described minimizes the objective. Therefore, we need to switch the sign from $-$ to $+$ in the gradient update step for the maximization problem.

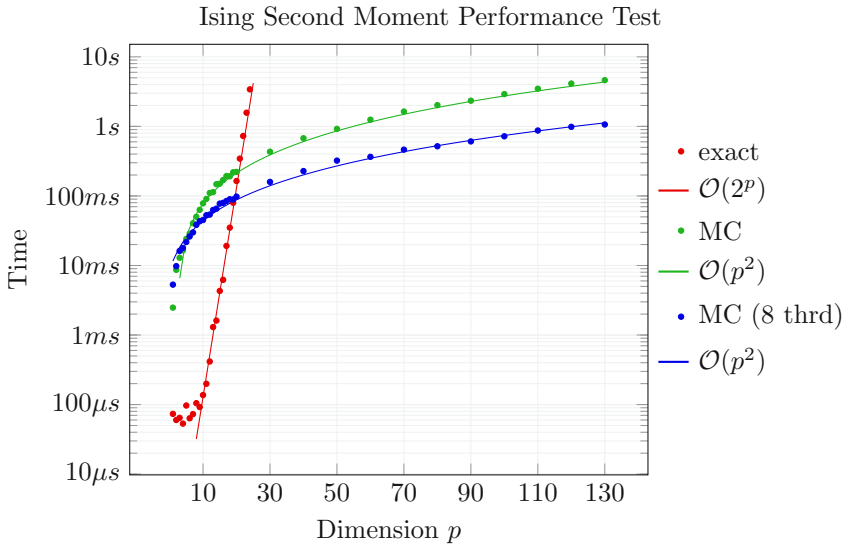

Figure 5.2: Performance test for computing/estimating the second moment of the Ising model of dimension p using ether the exact method or a Monte-Carlo (MC) simulation.

Specifically, we use a Gibbs-Sampler to sample from the conditional distribution and approximate the second moment in an importance sampling framework. This can be implemented quite efficiently while the estimation accuracy for the second moment is evaluated experimentally which seems to be very reliable. Simultaneously, we use the same approach to estimate the partition function. This though, is in comparison inaccurate, and may only be used to get a rough idea of the log-likelihood. Regardless, for our method, we only need the gradient for optimization where appropriate break conditions, not based on the likelihood, lead to a working method for MLE estimation.

5.2.2.5 Implementation in R

Here we provide a significantly simplified version of the Ising GMLM algorithm. A full implementation is beyond the scope of this discussion. The simulations in Section 5.5.2 use a complete implementation as described in Section 5.2.2 while the chess data example from Section 5.6.2 uses an even different implementation which is capable of streaming data, which is necessary for managing the gigantic amount of data involved.

With this in mind, we provide a pure R function to compute the second (uncentered) moment $\boldsymbol{M}_2(\boldsymbol{\Gamma}_y) = \mathfrak{g}_2(\boldsymbol{\gamma}_y)$ from (5.24) of the Ising model. The given prameters are $\boldsymbol{\Gamma}_y = \operatorname{vec}(\boldsymbol{\Omega} + \operatorname{diag}(\boldsymbol{B} \operatorname{vec} \mathcal{F}_y))$ as in (5.21).

```
ising.m2 <- function(Gamma) {
    prob.accum <- m2.accum <- 0
    bitmask <- bitwShiftL(1L, rev(seq_len(nrow(Gamma))) - 1)
    for (i in seq.int(0, 2^nrow(Gamma) - 1)) {
        x <- as.numeric(bitwAnd(i, bitmask) != 0L)
        prob_i <- exp(sum(x * (Gamma %*% x)))
```

```
        prob.accum <- prob.accum + prob_i
        m2.accum <- m2.accum + prob_i * outer(x, x)
    }
    list(m2 = m2.accum / prob.accum, log_p0 = -log(prob.accum))
}
```

The performance comparison in Figure 5.2 for the exact method uses the same base algorithm but is implemented in `C`, which is magnitutes faster.

Next we give the simplified Ising GMLM method. It is only for matrix-valued $\boldsymbol{X} \in \{0,1\}^{p_1 \times p_2}$ with very small $p = p_1 p_2$ (see Section 5.2.2.4). The data is also assumed to be non-degenerate (see Section 5.2.2.3). Finaly, we skip any optimization tricks. With all of those assumptions and simplifications we get the following implementation.

```
gmlm.ising.simple <- function(X, F,
    max.iter = 1000L, step.size = 1e-3, eps = 1e-8,
    proj.betas = list(identity, identity),
    proj.Omegas = list(identity, identity)
) {
    # Get problem dimensions
    dimX <- dim(X)[1:2]
    dimF <- dim(F)[1:2]
    sample.size <- dim(X)[3]

    # initialize β1,β2 as tensor normal estimates
    fit_normal <- gmlm.tensor.normal(X, F, proj.betas = proj.betas)
    beta_1 <- fit_normal$betas[[1]]
    beta_2 <- fit_normal$betas[[2]]

    init.Omega <- function(mode) {
        n <- prod(dim(X)[-mode])
        P2 <- tcrossprod(mat(X, mode)) / n
        P2[P2 == 0] <- 1 / n
        P2[P2 == n] <- (n - 1) / n
        P1 <- diag(P2)
        `P1^2` <- outer(P1, P1)

        `diag<-`(log(((1 - `P1^2`) / `P1^2`) * P2 / (1 - P2)), 0)
    }
    Omega_1 <- init.Omega(1)
    Omega_2 <- init.Omega(2)

    # Initialize mean squared gradients
    G2.beta_1  <- G2.beta_2  <- 0
    G2.Omega_1 <- G2.Omega_2 <- 0

    # Keep track of loss for break condition
    loss <- Inf

    # Iterate till a break condition triggers or till max. nr. of iterations
    for (iter in seq_len(max.iter)) {

        Omega <- do.call(kronecker, rev(Omegas))
        G.beta_1 <- G.beta_2 <- 0
        R2 <- 0             # second order residual accumulator
        loss.last <- loss
```

```
        loss <- 0           # negative log-likelihood

        for (i in seq_len(sample.size)) {
            params_i <- Omega + diag(as.vector(mlm(F[, , i], betas)))
            ising_i <- ising.m2(params_i)

            # accumulate loss
            vecX_i <- as.vector(X[, , i])
            loss <- loss - sum(vecX_i * (params_i %*% vecX_i)) + ising_i$log_p0

            R2_i <- tcrossprod(vecX_i) - ising_i$m2
            R1_i <- array(diag(R2_i), dimX)

            G.beta_1 <- G.beta_1 + tcrossprod(R1_i, F[, , i] %*% t(betas[[2]]))
            G.beta_2 <- G.beta_2 +  crossprod(R1_i, betas[[1]] %*% F[, , i])

            R2 <- R2 + as.vector(R2_i)
        }

        # check break conditions
        if (abs(loss.last - loss) < eps * loss.last) { break }

        K.R2 <- kronperm(R2, dims = c(dimX, dimX))
        G.Omega_1 <- as.vector(K.R2 %*% as.vector(Omega_2))
        G.Omega_2 <- as.vector(as.vector(Omega_1) %*% K.R2)

        # Accumulate root mean squared gradiends
        G2.beta_1  <- 0.9 * G2.beta_1  + 0.1 * G.beta_1 * G.beta_1
        G2.beta_2  <- 0.9 * G2.beta_2  + 0.1 * G.beta_2 * G.beta_2
        G2.Omega_1 <- 0.9 * G2.Omega_1 + 0.1 * G.Omega_1 * G.Omega_1
        G2.Omega_2 <- 0.9 * G2.Omega_2 + 0.1 * G.Omega_2 * G.Omega_2

        beta_1  <- beta_1  + step.size / (sqrt(G2.beta_1) + eps) * G.beta_1
        beta_2  <- beta_2  + step.size / (sqrt(G2.beta_2) + eps) * G.beta_2
        Omega_1 <- Omega_1 + step.size / (sqrt(G2.Omega_1) + eps) * G.Omega_1
        Omega_2 <- Omega_2 + step.size / (sqrt(G2.Omega_2) + eps) * G.Omega_2

        # Project parameters onto manifold
        beta_1  <- proj.betas[[1]](beta_1)
        beta_2  <- proj.betas[[2]](beta_2)
        Omega_1 <- proj.Omegas[[1]](Omega_1)
        Omega_2 <- proj.Omegas[[2]](Omega_2)
    }

    list(beta_1, beta_2)
}
```

5.3 Manifolds

Theorem 5 identifies the sufficient reduction for the regression of Y on $\mathcal{X}$ in the population. Any estimation of the sufficient reduction requires application of some optimality criterion. As we operate within the framework of the exponential family, we opted for maximum likelihood estimation (MLE). For the unconstrained problem, where the parameters are

simply $\boldsymbol{B}$ and $\boldsymbol{\Omega}$ in (5.4), maximizing the likelihood of $\mathcal{X} \mid Y$ is straightforward and yields well-defined MLEs of both parameters. Our setting, though, requires the constrained optimization of the $\mathcal{X} \mid Y$ likelihood subject to $\boldsymbol{B} = \bigotimes_{j=r}^{1} \boldsymbol{\beta}_j$ and $\boldsymbol{\Omega} = \bigotimes_{j=r}^{1} \boldsymbol{\Omega}_j$. Theorems 8 and 9 provide the setting for which the MLE of the constrained parameter $\boldsymbol{\theta}$ is well-defined, which in turn leads to the derivation of its asymptotic normality.

The main problem in obtaining asymptotic results for the MLE of the constrained parameter $\boldsymbol{\theta} = (\overline{\eta}, \operatorname{vec} \boldsymbol{B}, \operatorname{vech} \boldsymbol{\Omega})$ stems from the nature of the constraint. We assumed that $\boldsymbol{B} = \bigotimes_{k=r}^{1} \boldsymbol{\beta}_k$, where the parameter $\boldsymbol{B}$ is identifiable. This means that different values of $\boldsymbol{B}$ lead to different densities $f_{\boldsymbol{\theta}}(\mathcal{X} \mid Y = y)$, a basic property needed to ensure consistency of parameter estimates, which in turn is needed for asymptotic normality. On the other hand, the components $\boldsymbol{\beta}_j$, $j = 1, \ldots, r$, are *not* identifiable, which is a direct consequence of the equality $\boldsymbol{\beta}_2 \otimes \boldsymbol{\beta}_1 = (c\boldsymbol{\beta}_2) \otimes (c^{-1}\boldsymbol{\beta}_1)$ for every $c \neq 0$. This is the reason we formulated Θ as a constrained parameter space instead of parameterizing the densities of $\mathcal{X} \mid Y$ with respect to the components $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_r$. The same is true for $\boldsymbol{\Omega} = \bigotimes_{k=r}^{1} \boldsymbol{\Omega}_k$.

In addition to identifiable parameters, asymptotic normality obtained in Theorem 10 requires differentiation. Therefore, the space itself needs to admit defining differentiation, which is usually a vector space. This is too strong an assumption for our purposes. To weaken the vector space assumption we consider *smooth manifolds*. The latter are spaces which look like Euclidean spaces locally and allow the notion of differentiation. The more general *topological* manifolds are too weak for differentiation. To make matters worse, a smooth manifold only allows first derivatives. Without going into details, the solution is a *Riemannian manifold*. Similar to an abstract *smooth manifold*, Riemannian manifolds are detached from our usual intuition as well as complicated to handle in an already complicated setting. This is where an *embedded (sub)manifold* comes to the rescue. Simply speaking, an embedded manifold is a manifold which is a subset of a manifold from which it inherits its properties. If a manifold is embedded in a Euclidean space, almost all the complication of the abstract manifold theory simplifies drastically. Moreover, since a Euclidean space is itself a Riemannian manifold, we inherit the means for higher derivatives. Finally, smooth embedded submanifold structure for the parameter space maintains consistency with existing approaches and results for parameter sets with linear subspace structure. These reasons justify the constraint that the parameter space Θ be an *smooth embedded submanifold* in an open subset Ξ of a Euclidean space.

Now, we directly define a *smooth manifold* embedded in $\mathbb{R}^p$ without any detours to the more generel theory. See for example Lee [Lee12; Lee18], Absil, Mahony, and Sepulchre [AMS08], and Kaltenbäck [Kal21] among others.

Definition 10 (Manifolds)**.** A set $\mathfrak{A} \subseteq \mathbb{R}^p$ is an *embedded smooth manifold* of dimension d if for every $\boldsymbol{x} \in \mathfrak{A}$ there exists a smooth[4] bi-continuous map $\varphi : U \cap \mathfrak{A} \to V$, called a *chart*, with $\boldsymbol{x} \in U \subseteq \mathbb{R}^p$ open and $V \subseteq \mathbb{R}^d$ open.

We also need the concept of a *tangent space* to formulate asymptotic normality in a way which is independent of a particular coordinate representation. Intuitively, the tangent space at a point $\boldsymbol{x} \in \mathfrak{A}$ of the manifold $\mathfrak{A}$ is the hyperspace of all velocity vectors $\nabla\gamma(0)^T$ of any curve $\gamma : (-1, 1) \to \mathfrak{A}$ passing through $\boldsymbol{x} = \gamma(0)$, see Figure 5.3. Locally, at

[4] Here *smooth* means infinitely differentiable or C^∞.

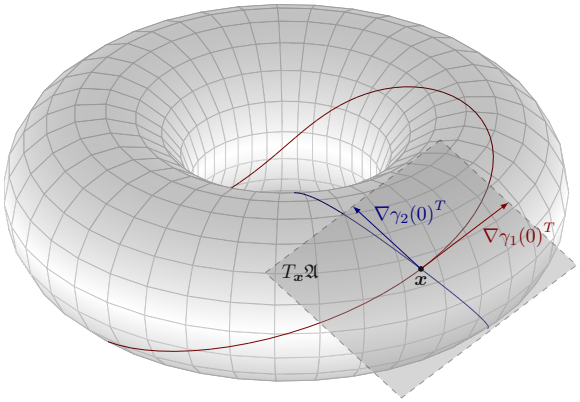

Figure 5.3: Visualization of the tangent space $T_{\boldsymbol{x}}\mathfrak{A}$ at $\boldsymbol{x}$ of the torus $\mathfrak{A}$. The torus $\mathfrak{A}$ is a 2-dimensional embedded manifold in $\mathbb{R}^3$. The tangent space $T_{\boldsymbol{x}}\mathfrak{A} \subset \mathbb{R}^3$ is a 2-dimensional hyperplane visualized with its origin $\mathbf{0}$ shifted to $\boldsymbol{x}$. Moreover, two curves γ_1, γ_2 on the torus are drawn with $\boldsymbol{x} = \gamma_1(0) = \gamma_2(0)$. The curve velocity vectors $\nabla\gamma_1(0)^T$ and $\nabla\gamma_2(0)^T$ are drawn as tangent vectors with root $\boldsymbol{x}$.

$\boldsymbol{x} = \gamma(0)$ with a chart φ we can written $\gamma(t) = \varphi^{-1}(\varphi(\gamma(t)))$ which gives that span $\nabla\gamma(0)^T \subseteq$ span $\nabla\varphi^{-1}(\varphi(\boldsymbol{x}))^T$. Taking the union over all smooth curves through $\boldsymbol{x}$ gives equality. The following definition leverages the simplified setup of smooth manifolds in Euclidean space.

Definition 11 (Tangent Space)**.** Let $\mathfrak{A} \subseteq \mathbb{R}^p$ be an embedded smooth manifold and $\boldsymbol{x} \in \mathfrak{A}$. The *tangent space* at $\boldsymbol{x}$ of $\mathfrak{A}$ is defined as

$$T_{\boldsymbol{x}}\mathfrak{A} = \operatorname{span} \nabla\varphi^{-1}(\varphi(\boldsymbol{x}))^T$$

for any chart φ with $\boldsymbol{x}$ in the pre-image of φ.

Definition 11 is consistent since it can be shown that two different charts at the same point have identical span.

The *Submersion Theorem* is a very powerfull tool to determin if a particular set is a manifold (provided without proof). For our use the most interesting case is if $\mathfrak{A}$ is a Euclidean space, like $\mathbb{R}^p$ or $\mathbb{R}^{p\times q}$.

Theorem 7 (Submersion Theorem, [AMS08, Prop 3.3.3])**.** *Let $f : \mathfrak{A} \to \mathfrak{B}$ be a smooth function between manifolds $\mathfrak{A}, \mathfrak{B}$ with $\dim(\mathcal{M}) > \dim(\mathcal{N})$. If $y \in \mathfrak{B}$ is a point such that for all $x \in f^{-1}(y) \subseteq \mathfrak{A}$ the differential $\mathrm{d}f(x)$ has full rank, then $f^{-1}(y)$ is a closed* embedded submanifold *of dimension* $\dim(\mathfrak{A}) - \dim(\mathfrak{B})$.

Not directly related to manifolds in general but needed in Section 5.3.2 is the concept of a *spherical* set, which is a set $\mathfrak{A}$, on which the Frobenius norm is constant. That is, $\|\,.\,\|_F : \mathfrak{A} \to \mathbb{R}$ is constant. Another set property is to be a *cone*, denoting a scale invariant set $\mathfrak{A}$, that is $\mathfrak{A} = \{c\boldsymbol{A} : \boldsymbol{A} \in \mathfrak{A}\}$ for all $c > 0$.

5.3.1 Matrix Manifolds

A special kind of manifolds are the *matrix manifolds*, which are submanifold of $\mathbb{R}^{p\times q}$. A good reference for this kind of manifolds in [AMS08]. Here we introduce some common matrix manifolds which are usefill as building blocks for the *Kronecker Manifolds* of Section 5.3.2.

The *noncompact Stiefel manifold* is the set of all full column rank matrices

$$\mathbb{R}_*^{p\times q} = \{\boldsymbol{A} \in \mathbb{R}^{p\times q} : \det(\boldsymbol{A}^T\boldsymbol{A}) \neq 0\}$$

which is a open subset of $\mathbb{R}^{p\times q}$. This is a generalization of the *general linear group*, as in the case of square matrices ($p = q$) they are identical $\mathbb{R}_*^{p\times q} \simeq \mathrm{GL}_p(\mathbb{R})$. To see why $\mathbb{R}_*^{p\times q}$ is open, consider the smooth function $f : \boldsymbol{A} \mapsto \det(\boldsymbol{A}^T\boldsymbol{A})$, then $\mathbb{R}_*^{p\times q} = f^{-1}(0)^c$ where $f^{-1}(0)$ is closed as preimage under a continuous function. Its dimension is pq as open subset of a pq dimensional manifold, moreover it is also a cone because scaling a matrix with a non-zero constant does not change its rank.

Another simple matrix manifold is the set of *symmetric matrices*

$$\mathrm{Sym}^{p\times p} = \{\boldsymbol{A} \in \mathbb{R}^{p\times p} : \boldsymbol{A}^T = \boldsymbol{A}\}$$

which is a closed convex cone in $\mathbb{R}^{p\times p}$. A well known and widely used matrix manifold in statistics is the *Stiefel manifold* for $p \geq q$ given by

$$\mathrm{St}^{p\times q} = \{\boldsymbol{A} \in \mathbb{R}^{p\times q} : \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_q\}$$

which is compact and spherical with dimension $pq - \frac{1}{2}q(q+1)$. The Stiefel manifold contains the semiorthogonal matrices. For $q = 1$, the Stiefel manifold is identical to the *hypersphere* $\mathrm{St}^{p\times 1} \simeq \mathrm{S}^{p-1}$ while in the other extreme where $p = q$ the Stiefel manifold is the *special linear group* $\mathrm{St}^{p\times p} \simeq \mathrm{SL}_p(\mathbb{R})$. To see why the Stiefel manifold is actually a submanifold of $\mathbb{R}^{p\times q}$, consider the function $f : \mathbb{R}^{p\times q} \to \mathrm{Sym}^{q\times q} : \boldsymbol{A} \mapsto \boldsymbol{A}^T\boldsymbol{A} - \boldsymbol{I}_q$. The Stiefel manifold is then the preimage $f^{-1}(\boldsymbol{0}_{p\times p})$. In order to apply Theorem 7, we need to show that $\boldsymbol{0}_{q\times q}$ is a regular point of f, that is $\mathrm{D}f(\boldsymbol{A})$ has full rank $\forall \boldsymbol{A} \in f^{-1}(\boldsymbol{0}_{q\times q}) = \mathrm{St}^{p\times q}$. To show that $\mathrm{D}f(\boldsymbol{A})$ has full rank, we show that $\forall \boldsymbol{C} \in \mathrm{Sym}^{q\times q}$ there exists a $\boldsymbol{B} \in T_{\boldsymbol{A}}\mathbb{R}^{p\times q}$ such that the directional derivatives at $\boldsymbol{A}$ in direction $\boldsymbol{B}$ is equal to $\boldsymbol{C}$, that is $\boldsymbol{C} = \mathrm{D}f(\boldsymbol{A})[\boldsymbol{B}] = \boldsymbol{B}^T\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{B}$. Setting $\boldsymbol{B} = \frac{1}{2}\boldsymbol{A}\boldsymbol{C}$ obtains $\mathrm{D}f(\boldsymbol{A})[\boldsymbol{B}] = \frac{1}{2}(\boldsymbol{C}^T\boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}) = \boldsymbol{C}$, which shows that $\boldsymbol{0}_{q\times q}$ is a regular point of f. Now we can apply Theorem 7 which gives that $\mathrm{St}^{p\times q}$ is a closed submanifold of $\mathbb{R}^{p\times q}$ of dimension $\dim(\mathrm{St}^{p\times q}) = \dim(\mathbb{R}^{p\times q}) - \dim(\mathrm{Sym}^{q\times q}) = pq - q(q+1)/2$. The Stiefel manifold is also spherical because for every $\boldsymbol{A} \in \mathrm{St}^{p\times q}$ we get $\|\boldsymbol{A}\|_F = \sqrt{\mathrm{tr}(\boldsymbol{A}^T\boldsymbol{A})} = \sqrt{q}$.

We are also interested in the set of all matrices of fixed rank r,

$$\mathbb{R}_{\mathrm{rank}=r}^{p\times q} = \{\boldsymbol{A} \in \mathbb{R}^{p\times q} : \mathrm{rank}\,\boldsymbol{A} = r\}$$

which can be shown to be a smooth embedded submanifold of dimension $\dim(\mathbb{R}_{\mathrm{rank}=r}^{p\times q}) = r(p+q-r)$. Similarly to [UV20, Sec. 9.2.2], we sketch the derivation. Let $\mathcal{B} \subseteq \mathbb{R}_{\mathrm{rank}=r}^{p\times q}$ be the set of matrices where the left upper $r \times r$ block is invertible. Then, $\boldsymbol{A} \in \mathcal{B}$ can be written as a block matrix

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B} & \boldsymbol{C} \\ \boldsymbol{C} & \boldsymbol{E} \end{pmatrix}$$

$\boldsymbol{A}$ has rank r if and only if the Schur complement $f(\boldsymbol{A}) = \boldsymbol{E} - \boldsymbol{D}\boldsymbol{B}^{-1}\boldsymbol{C}$ of the block $\boldsymbol{D}$ is zero. Then $f^{-1}(0)$ is an embedded submanifold by Theorem 7 of dimension $pq-(p-r)(q-r)$. Now, as every rank r matrix contains a invertible $r \times r$ submatrix we split the space into matrices where the same submatrix is invertible. For every such set we can permute rows and columns mapping the invertible submatrix to the left upper $r \times r$ block. For the complete set we construct an atlas from the atlas of $f^{-1}(0)$ by concatenating the row and column mappings with the charts of $f^{-1}(0)$.

Another important matrix manifold is comprised of the *symmetric positive definite* (SPD) matrices

$$\mathrm{Sym}_{++}^{p\times p} = \{\boldsymbol{A} \in \mathrm{Sym}^{p\times p} : \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} > 0 \,\forall \boldsymbol{x} \in \mathbb{R}_*^p\}.$$

The manifold structure of the SPD matrices is based on the smooth and bijective relation between the SPD matrices and their *Cholesky decomposition* [Lin19]. The Cholesky decomposition of a SPD matrix $\boldsymbol{A}$ is $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$ where $\boldsymbol{L}$ is a lower triangular matrix with positive diagonal entries. The Cholesky decomposition of a SPD matrix is unique, which gives a bijection $\boldsymbol{A} \mapsto \boldsymbol{L}$ which is also smooth. As such the SPD matrices are an embedded submanifold if the set of lower triangular matrices is an embedded manifold. This is certainly the case as the lower triangular matrices are simply a hyperplane. The subset of matrices with positive diagonal elements is an open subset in the hyperplane of lower triangular matrices, which makes it an embedded submanifold.

An overview of the discussed, and some additional embedded matrix manifolds, is given in Table 5.1.

Remark 21. The *Grassmann Manifold* of q dimensional subspaces in $\mathbb{R}^p$ is not listed in Table 5.1 since it is not embedded in $\mathbb{R}^{p\times q}$.

5.3.2 Kronecker Product Manifolds

As a basis to ensure that the constrained parameter space Θ is a manifold, which is a requirement of Theorem 9, we need Theorem 8.

Theorem 8 (Kronecker Product Manifolds)**.** *Let* $\mathfrak{A} \subseteq \mathbb{R}^{p_1\times q_1}\backslash\{\boldsymbol{0}\}, \mathfrak{B} \subseteq \mathbb{R}^{p_2\times q_2}\backslash\{\boldsymbol{0}\}$ *be smooth embedded submanifolds. Assume one of the following conditions holds.*

- *"sphere condition": At least one of* $\mathfrak{A}$ *or* $\mathfrak{B}$ *is* spherical *and let* $d = \dim\mathfrak{A} + \dim\mathfrak{B}$.
- *"cone condition": Both* $\mathfrak{A}$ *and* $\mathfrak{B}$ *are* cones *and let* $d = \dim\mathfrak{A} + \dim\mathfrak{B} - 1$.

Then, $\{\boldsymbol{A}\otimes\boldsymbol{B} : \boldsymbol{A} \in \mathfrak{A}, \boldsymbol{B} \in \mathfrak{B}\} \subset \mathbb{R}^{p_1p_2\times q_1q_2}$ *is a smooth embedded* d*-manifold.*

Proof. We start by considering the first case and assume that $\mathfrak{B}$ is spherical with radius 1 w.l.o.g. We equip $\mathfrak{K} = \{\boldsymbol{A}\otimes\boldsymbol{B} : \boldsymbol{A} \in \mathfrak{A}, \boldsymbol{B} \in \mathfrak{B}\} \subset \mathbb{R}^{p_1p_2\times q_1q_2}$ with the subspace topology (see [Lee12; Kal21]). Define the hemispheres $H_i^+ = \{\boldsymbol{B} \in \mathfrak{B} : (\mathrm{vec}\,\boldsymbol{B})_i > 0\}$ and $H_i^- = \{\boldsymbol{B} \in \mathfrak{B} : (\mathrm{vec}\,\boldsymbol{B})_i < 0\}$ for $i = 1, ..., p_2q_2$. The hemispheres are an open cover of $\mathfrak{B}$ with respect to the subspace topology. Define for every $H_i^\pm$, where $\pm$ is a placeholder for ether $+$ or $-$, the function

$$f_{H_i^\pm} : \mathfrak{A}\times H_i^\pm \to \mathbb{R}^{p_1p_2\times q_1q_2} : (\boldsymbol{A},\boldsymbol{B}) \mapsto \boldsymbol{A}\otimes\boldsymbol{B}$$

Symbol	Description	C	S
$\mathbb{R}^{p\times q}$	All matrices of dimension $p \times q$ Dim: pq	✓	✗
$\mathbb{R}_*^{p\times q}$	Full rank $p \times q$ matrices Dim: pq	✓	✗
$\mathrm{St}^{p\times q}$	*Stiefel Manifold*, $\{\boldsymbol{U} \in \mathbb{R}^{p\times q} : \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_q\}$ for $q \leq p$ Dim: $pq - q(q+1)/2$	✗	✓
S^{p-1}	Unit sphere in $\mathbb{R}^p$, special case $\mathrm{St}^{p\times 1}$ of the Stiefel Manifold Dim: $p-1$	✗	✓
$\mathrm{O}(p)$	*Orthogonal Group*, special case $\mathrm{St}^{p\times p}$ of the Stiefel Manifold Dim: $p(p-1)/2$	✗	✓
$\mathrm{SO}(p)$	*Special Orthogonal Group* $\{\boldsymbol{U} \in \mathrm{O}(p) : \det \boldsymbol{U} = 1\}$ Dim: $p(p-1)/2$	✗	✓
$\mathbb{R}_r^{p\times q}$	Matrices of known rank $r > 0$, generalizes $\mathbb{R}_*^{p\times q}$ Dim: $r(p+q-r)$	✓	✗
	Symmetric matrice Dim: $p(p+1)/2$	✓	✗
$\mathrm{Sym}_{++}^{p\times p}$	Symmetric Positive Definite matrices Dim: $p(p+1)/2$	✓	✗
	Scaled Identity $\{a\boldsymbol{I}_p : a \in \mathbb{R}_+\}$ Dim: 1	✓	✗
	Symmetric r-band matrices (includes diagonal) $\{\boldsymbol{A} \in \mathbb{R}^{p\times p} : \boldsymbol{A} = \boldsymbol{A}^T \wedge \boldsymbol{A}_{ij} = 0 \ \forall \lvert i-j\rvert > r\}$ Dim: $(2p-r)(r+1)/2$	✓	✗
	Auto correlation alike $\{\boldsymbol{A} \in \mathbb{R}^{p\times p} : \boldsymbol{A}_{ij} = \rho^{\lvert i-j\rvert}, \rho \in (0,1)\}$ Dim: 1	✗	✗

Table 5.1: Examples of embedded matrix manifolds. "Symbol" a (more or less) common notation for the matrix manifold, if at all. "C" stands for *cone*, meaning it is scale invariant. "S" means *spherical*, that is, constant Frobenius norm.

which is smooth. With the spherical property of $\mathfrak{B}$ the relation $\|\boldsymbol{A}\otimes\boldsymbol{B}\|_F = \|\boldsymbol{A}\|_F$ for all $\boldsymbol{A}\otimes\boldsymbol{B}\in\mathfrak{K}$ ensures that the function $f_{H_i^\pm}$, constrained to its image, is bijective with inverse function (identifying $\mathbb{R}^{p\times q}$ with $\mathbb{R}^{pq}$) given by

$$f^{-1}_{H_i^\pm} : \begin{cases} f_{H_i^\pm}(\mathfrak{A}\times H_i^\pm) \to \mathfrak{A}\times H_i^\pm \\ \boldsymbol{C} \mapsto \left(\pm\frac{\|\boldsymbol{C}\|_F}{\|\boldsymbol{R}(\boldsymbol{C})\boldsymbol{e}_i\|_2}\boldsymbol{R}(\boldsymbol{C})\boldsymbol{e}_i, \pm\frac{1}{\|\boldsymbol{C}\|_F\|\boldsymbol{R}(\boldsymbol{C})\boldsymbol{e}_i\|_2}\boldsymbol{R}(\boldsymbol{C})\boldsymbol{R}(\boldsymbol{C})^T\boldsymbol{e}_i\right) \end{cases}$$

where $\pm$ is $+$ for a "positive" hemisphere H_i^+ and $-$ otherwise, $\boldsymbol{e}_i \in \mathbb{R}^{p_2q_2}$ is the ith unit vector and $\boldsymbol{R}(\boldsymbol{C})$ is a "reshaping" permutation[5] which acts on Kronecker products as $\boldsymbol{R}(\boldsymbol{A}\otimes\boldsymbol{B}) = (\operatorname{vec}\boldsymbol{A})(\operatorname{vec}\boldsymbol{B})^T$. This makes $f^{-1}_{H_i^\pm}$ a combination of smooth functions ($\boldsymbol{0}$ is excluded from $\mathfrak{A},\mathfrak{B}$ guarding against division by zero) and as such it is also smooth. This ensures that $f_{H_i^\pm} : \mathfrak{A}\times H_i^\pm \to f_{H_i^\pm}(\mathfrak{A}\times H_i^\pm)$ is a diffeomorphism.

Next, we construct an atlas[6] for $\mathfrak{K}$ which is equipped with the subspace topology. Let $(\varphi_j, U_j)_{j\in J}$ be a atlas of $\mathfrak{A}\times\mathfrak{B}$. Such an atlas exists and admits a unique smooth structure as both $\mathfrak{A},\mathfrak{B}$ are embedded manifolds from which we take the product manifold. The images of the coordinate domains $f_H(U_j)$ are open in $\mathfrak{K}$, since f_H is a diffeomorphism, with the corresponding coordinate maps

$$\phi_{H_i^\pm,j} : f_{H_i^\pm}(U_j) \to \varphi_j(U_j) : \boldsymbol{C} \mapsto \varphi_j(f^{-1}_{H_i^\pm}(\boldsymbol{C})).$$

By construction the set $\{\phi_{H_i^\pm,j} : i = 1,\dots,p_2q_2, \pm\in\{+,-\}, j\in J\}$ is an atlas if the charts are compatible. This means we need to check if the transition maps are diffeomorphisms, let $(\phi_{H,j}, V_j), (\phi_{\widetilde{H},k}, V_k)$ be two arbitrary charts from our atlas, then the transition map $\phi_{\widetilde{H},k}\circ\phi^{-1}_{H,j} : \phi_{H,j}(V_j\cap V_k) \to \phi^{-1}_{\widetilde{H},k}(V_j\cap V_k)$ has the form

$$\phi_{\widetilde{H},k}\circ\phi^{-1}_{H,j} = \varphi_k\circ f^{-1}_{\widetilde{H}}\circ f_H\circ\varphi_j^{-1} = \varphi_k\circ(\pm\mathrm{id})\circ\varphi_j^{-1}$$

where $\pm$ depends on $H,\widetilde{H}$ being of the same "sign" and id is the identity. We conclude that the charts are compatible, which makes the constructed set of charts an atlas. With that we have shown the topological manifold $\mathfrak{K}$ with the subspace topology admit a smooth atlas that makes it an embedded smooth manifold with dimension equal to the dimension of the product topology $\mathfrak{A}\times\mathfrak{B}$; that is, $d = \dim\mathfrak{A} + \dim\mathfrak{B}$.

It remains to show that the cone condition also admits a smooth manifold. $\mathfrak{K} = \{\boldsymbol{A}\otimes\boldsymbol{B} : \boldsymbol{A}\in\mathfrak{A}, \boldsymbol{B}\in\widetilde{B}\}$, where $\widetilde{B} = \{\boldsymbol{B}\in\mathfrak{B} : \|\boldsymbol{B}\|_F = 1\}$, holds if both $\mathfrak{A},\mathfrak{B}$ are cones. Since $g : \mathfrak{B}\to\mathbb{R} : \boldsymbol{B}\mapsto\|\boldsymbol{B}\|_F$ is continuous on $\mathfrak{B}$ with full rank 1 everywhere, $\widetilde{\mathfrak{B}} = g^{-1}(1)$ is a $\dim\mathfrak{B}-1$ dimensional embedded submanifold of $\mathfrak{B}$. An application of the spherical case proves the cone case. □

[5] Relating to $\mathcal{K}$ the operation $\boldsymbol{R}$ is basically its inverse as $\mathcal{K}(\boldsymbol{A}\circ\boldsymbol{B}) = \boldsymbol{A}\otimes\boldsymbol{B}$ with a mismatch in the shapes only.

[6] A collection of charts $\{\varphi_i : i\in I\}$ with index set I of a manifold $\mathfrak{A}$ is called an *atlas* if the pre-images of the charts φ_i cover the entire manifold $\mathfrak{A}$.

Theorem 9 (Parameter Manifold)**.** *Let*

$$\mathfrak{K}_{\boldsymbol{B}} = \left\{ \bigotimes_{k=r}^{1} \boldsymbol{\beta}_k : \boldsymbol{\beta}_k \in \mathfrak{B}_k \right\} \quad \textit{and} \quad \mathfrak{K}_{\boldsymbol{\Omega}} = \left\{ \bigotimes_{k=r}^{1} \boldsymbol{\Omega}_k : \boldsymbol{\Omega}_k \in \mathfrak{O}_k \right\}$$

where $\mathfrak{B}_k \subseteq \mathbb{R}^{p_k \times q_k} \backslash \{\mathbf{0}\}$ *and* $\mathfrak{O}_k \subseteq \mathbb{R}^{p_k \times p_k} \backslash \{\mathbf{0}\}$ *are smooth embedded manifolds which are either spheres or cones, for* $k = 1, ..., r$*. Furthermore, let*

$$\mathfrak{CK}_{\boldsymbol{\Omega}} = \{\operatorname{vech} \boldsymbol{\Omega} : \boldsymbol{\Omega} \in \mathfrak{K}_{\boldsymbol{\Omega}} \wedge (\boldsymbol{T}_2 \boldsymbol{D}_p^\dagger)^\dagger \boldsymbol{T}_2 \boldsymbol{D}_p^\dagger \operatorname{vec} \boldsymbol{\Omega} = \operatorname{vec} \boldsymbol{\Omega}\}$$

then the constrained parameter space $\Theta = \mathbb{R}^p \times \mathfrak{K}_{\boldsymbol{B}} \times \mathfrak{CK}_{\boldsymbol{\Omega}} \subset \mathbb{R}^{p(p+2q+3)/2}$ *is a smooth embedded manifold.*

Proof. An application of Lemma 2 ensures that $\mathfrak{K}_{\boldsymbol{B}}$ and $\mathfrak{K}_{\boldsymbol{\Omega}}$ are embedded submanifolds.

With $\boldsymbol{T}_2$ being a $d \times p(p+1)/2$ full rank matrix and the duplication matrix $\boldsymbol{D}_p$ is full rank of dimension $p(p+1)/2 \times p^2$ we have $\boldsymbol{T}_2 \boldsymbol{D}_p^\dagger$ to be $d \times p^2$ of full rank. This means that $\boldsymbol{P} = (\boldsymbol{T}_2 \boldsymbol{D}_p^\dagger)^\dagger \boldsymbol{T}_2 \boldsymbol{D}_p^\dagger$ is a $p^2 \times p^2$ projection of rank d and $\boldsymbol{I}_{p^2} - \boldsymbol{P}$ is then a projection of rank $p^2 - d$. This leads to

$$\mathfrak{CK}_{\boldsymbol{\Omega}} = \{\boldsymbol{\Omega} \in \mathfrak{K}_{\boldsymbol{\Omega}} : (\boldsymbol{I}_{p^2} - \boldsymbol{P}) \operatorname{vec} \boldsymbol{\Omega} = \mathbf{0}\}.$$

To show that $\mathfrak{CK}_{\boldsymbol{\Omega}}$ is an embedded submanifold of $\mathfrak{K}_{\boldsymbol{\Omega}}$ we apply the "Constant-Rank Level Set Theorem" Lee [Lee12, Thm 5.12] which states (slightly adapted) the following; Let $\mathfrak{A}$, $\mathfrak{B}$ be smooth manifolds and $F : \mathfrak{A} \to \mathfrak{B}$ a smooth map such that $\nabla_{\boldsymbol{a}} F$ has constant matrix rank for all $\boldsymbol{a} \in \mathfrak{A}$. Then, for every $\boldsymbol{b} \in F(\boldsymbol{A}) \subseteq \mathfrak{B}$, the preimage $F^{-1}(\{\boldsymbol{b}\})$ is a smooth embedded submanifold of $\mathfrak{A}$.

In our setting, we have $F : \mathfrak{K}_{\boldsymbol{\Omega}} \to \mathbb{R}^{p^2}$ defined as $F(\boldsymbol{\Omega}) = (\boldsymbol{I}_{p^2} - \boldsymbol{P}) \operatorname{vec} \boldsymbol{\Omega}$ with gradient $\nabla_{\boldsymbol{\Omega}} F = \boldsymbol{I}_{p^2} - \boldsymbol{P}$ of constant rank. Therefore, $F^{-1}(\{\mathbf{0}\}) = \mathfrak{CK}_{\boldsymbol{\Omega}}$ is an embedded submanifold of $\mathfrak{K}_{\boldsymbol{\Omega}}$.

Finally, the finite product manifold of embedded submanifolds is embedded in the finite product space of their ambient spaces, that is $\Theta = \mathbb{R}^p \times \mathfrak{K}_{\boldsymbol{B}} \times \mathfrak{CK}_{\boldsymbol{\Omega}} \subset \mathbb{R}^p \times \mathbb{R}^{p \times q} \times \mathbb{R}^{p \times p}$ is embedded. □

A powerful side effect of Theorem 9 is the modeling flexibinity it provides. For example, we can perform low rank regression. Or, we may constrain two-way interactions between direct axis neighbors by using band matrices for the $\boldsymbol{\Omega}_k$'s, among others.

This flexibility derives from many different matrix manifolds that can be used as building blocks $\mathfrak{B}_k$ and $\mathfrak{O}_k$ of the parameter space Θ in Theorem 9. A list of possible choices, among others, is given in Table 5.1. As long as parameters in Θ are valid paramererization of a density (or PMF) of (5.3) subject to (5.4) and (5.5), one may choose any of the manifolds listed in Table 5.1 which are either cones or spherical. We also included an example which is neither a sphere nor a cone. They may also be valid building blocks, but require more work as they are not directly leading to a parameter manifold by Theorem 9. In case one can show the resulting parameter space Θ is an embedded manifold, the asymptotic theory of Section 5.4.1 is applicable.

5.4 Statistical Properties

5.4.1 Asymptotic Consistency

Let Z be a random variable distributed according to a parameterized probability distribution with density $f_{\boldsymbol{\theta}_0} \in \{f_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}$ where Θ is a subset of a Euclidean space. We want to estimate the parameter $\boldsymbol{\theta}_0$ using n i.i.d. (independent and identically distributed) copies of Z. We assume a known, real-valued and measurable function $z \mapsto m_{\boldsymbol{\theta}}(z)$ for every $\boldsymbol{\theta} \in \Theta$ and that $\boldsymbol{\theta}_0$ is the unique maximizer of the map $\boldsymbol{\theta} \mapsto M(\boldsymbol{\theta}) = \mathbb{E}\, m_{\boldsymbol{\theta}}(Z)$. For the estimation we maximize the empirical version

$$M_n(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} m_{\boldsymbol{\theta}}(Z_i). \tag{5.25}$$

An *M-estimator* $\widehat{\boldsymbol{\theta}}_n = \widehat{\boldsymbol{\theta}}_n(Z_1, ..., Z_n)$ is a maximizer for the objective function M_n over the parameter space Θ defined as

$$\widehat{\boldsymbol{\theta}}_n = \arg\max_{\boldsymbol{\theta} \in \Theta} M_n(\boldsymbol{\theta}).$$

It is not necessary to have a perfect maximizer, as long as the objective has finite supremum, it is sufficient to take an *almost maximizer* $\widehat{\boldsymbol{\theta}}_n$ as defined in the following;

Definition 12 (weak and strong M-estimators)**.** An estimator $\widehat{\boldsymbol{\theta}}_n$ for the objective function M_n in (5.25) with $\sup_{\boldsymbol{\theta} \in \Theta} M_n(\boldsymbol{\theta}) < \infty$ such that

$$M_n(\widehat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta} \in \Theta} M_n(\boldsymbol{\theta}) - o_P(n^{-1})$$

is called a *strong M-estimator* over Θ. Replacing $o_P(n^{-1})$ by $o_P(1)$ gives a *weak M-estimator*.

Theorem 10 (Asymptotic Normality)**.** *Assume $Z = (\mathcal{X}, Y)$ satisfies model (5.3) subject to (5.4) and (5.5) with true constrained parameter $\boldsymbol{\theta}_0 = (\overline{\eta}_0, \boldsymbol{B}_0, \boldsymbol{\Omega}_0) \in \Theta$, where Θ is defined in Theorem 9. Under the regularity Conditions 3-5 in the appendix, there exists a strong M-estimator sequence $\widehat{\boldsymbol{\theta}}_n$ deriving from l_n in (5.11) over Θ. Furthermore, any strong M-estimator $\widehat{\boldsymbol{\theta}}_n$ converges in probability to the true parameter $\boldsymbol{\theta}_0$ over Θ. That is, $\widehat{\boldsymbol{\theta}}_n \xrightarrow{p} \boldsymbol{\theta}_0$. Moreover, every strong M-estimator $\widehat{\boldsymbol{\theta}}_n$ is asymptotically normal,*

$$\sqrt{n}(\widehat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0) \xrightarrow{d} \mathcal{N}_{p(p+2q+3)/2}(0, \boldsymbol{\Sigma}_{\boldsymbol{\theta}_0})$$

with asymptotic variance-covariance structure $\boldsymbol{\Sigma}_{\boldsymbol{\theta}_0}$ given in (5.26).

Proof. The proof consists of three parts. First, we show the existence of a consistent strong M-estimator by applying Theorem 12. Next, we apply Theorem 13 to obtain its asymptotic normality. We conclude by computing the missing parts of the asymtotic covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}_0}$ provided by Theorem 13.

We check whether the conditions of Theorem 12 are satisfied. On Ξ, the mapping $\boldsymbol{\xi} \mapsto m_{\boldsymbol{\xi}}(z) = m_{\boldsymbol{\xi}}(\mathcal{X}, y) = \langle \boldsymbol{F}(y)\boldsymbol{\xi}, \boldsymbol{t}(\mathcal{X}) \rangle - b(\boldsymbol{F}(y)\boldsymbol{\xi})$ is strictly concave for every z because $\boldsymbol{\xi} \mapsto \boldsymbol{F}(y)\boldsymbol{\xi}$ is linear and b is strictly convex by Condition 3. Since $\mathcal{X} \mid Y$ is distributed

according to (5.3), the function $M(\boldsymbol{\xi}) = \mathbb{E}\, m_{\boldsymbol{\xi}}(Z)$ is well defined by Condition 4. Let $\boldsymbol{\xi}_k = (\operatorname{vec} \overline{\eta}_k, \operatorname{vec} \boldsymbol{B}_k, \operatorname{vech} \boldsymbol{\Omega}_k)$, and $f_{\boldsymbol{\xi}_k}$ be the pdf of $\mathcal{X} \mid Y$ indexed by $\boldsymbol{\xi}_k$, for $k = 1, 2$. If $\boldsymbol{\xi}_1 \neq \boldsymbol{\xi}_2$, then $f_{\boldsymbol{\xi}_1} \neq f_{\boldsymbol{\xi}_2}$, which obtains that the true $\boldsymbol{\theta}_0$ is a unique maximizer of $\boldsymbol{\theta}_0 \in \Theta \subseteq \Xi$ by applying van der Vaart [van98, Lemma 5.35]. Finally, under Condition 5, all assumptions of Theorem 12 are fulfilled yielding the existence of an consistent strong M-estimator over $\Theta \subseteq \Xi$.

Next, let $\widehat{\boldsymbol{\theta}}_n$ be a strong M-estimator on $\Theta \subseteq \Xi$, whose existence and consistency was shown in the previous step. Since $z \mapsto m_{\boldsymbol{\xi}}(z)$ is measurable for all $\boldsymbol{\xi} \in \Xi$, it is also measurable in a neighborhood of $\boldsymbol{\theta}_0$. The differentiability of $\boldsymbol{\theta} \mapsto m_{\boldsymbol{\theta}}(z)$ is stated in Condition 3. For the Lipschitz condition, let $K \subseteq \Xi$ be a compact neighborhood of $\boldsymbol{\theta}_0$, which exists since Ξ is open. Then,

$$\begin{aligned} |m_{\boldsymbol{\theta}_1}(z) - m_{\boldsymbol{\theta}_2}(z)| &= |\langle \boldsymbol{t}(\mathcal{X}), \boldsymbol{F}(y)(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2)\rangle - b(\boldsymbol{F}(z)\boldsymbol{\theta}_1) + b(\boldsymbol{F}(z)\boldsymbol{\theta}_2)| \\ &\leq (\|\boldsymbol{F}(y)^T \boldsymbol{t}(\mathcal{X})\|_2 + \sup_{\boldsymbol{\theta} \in K} \|\nabla b(\boldsymbol{F}(y)\boldsymbol{\theta})\boldsymbol{F}(y)\|)\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2 =: u(z)\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2 \end{aligned}$$

with $u(z)$ being measurable and square integrable derives from Condition 5. The existence of a second-order Taylor expansion of $\boldsymbol{\theta} \mapsto M(\boldsymbol{\theta}) = \mathbb{E}\, m_{\boldsymbol{\theta}}(Z)$ in a neighborhood of $\boldsymbol{\theta}_0$ holds by Condition 5. Moreover, the Hessian $\boldsymbol{H}_{\boldsymbol{\theta}_0}$ is non-singular by the strict convexity of b stated in Condition 3. Now, we can apply Theorem 13 to obtain the asymptotic normality of $\sqrt{n}(\widehat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0)$ with variance-covariance structure

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}_0} = \boldsymbol{\Pi}_{\boldsymbol{\theta}_0}\, \mathbb{E}[\nabla m_{\boldsymbol{\theta}_0}(Z)(\nabla m_{\boldsymbol{\theta}_0}(Z))^T]\boldsymbol{\Pi}_{\boldsymbol{\theta}_0} \tag{5.26}$$

where $\boldsymbol{\Pi}_{\boldsymbol{\theta}_0} = \boldsymbol{P}_{\boldsymbol{\theta}_0}(\boldsymbol{P}_{\boldsymbol{\theta}_0}^T \boldsymbol{H}_{\boldsymbol{\theta}_0} \boldsymbol{P}_{\boldsymbol{\theta}_0})^{-1} \boldsymbol{P}_{\boldsymbol{\theta}_0}^T$ and $\boldsymbol{P}_{\boldsymbol{\theta}_0}$ is any $p \times \dim(\Theta)$ matrix such that it spans the tangent space of Θ at $\boldsymbol{\theta}_0$. That is, $\operatorname{span} \boldsymbol{P}_{\boldsymbol{\theta}_0} = T_{\boldsymbol{\theta}_0}\Theta$.

Finally, we compute a matrix $\boldsymbol{P}_{\boldsymbol{\theta}_0}$ such that $\operatorname{span} \boldsymbol{P}_{\boldsymbol{\theta}_0} = T_{\boldsymbol{\theta}_0}\Theta$ for $\Theta = \mathbb{R}^p \times \mathfrak{K}_{\boldsymbol{B}} \times \mathfrak{CK}_{\boldsymbol{\Omega}}$ as in Theorem 9. Since the manifold Θ is a product manifold we get a block diagonal structure for $\boldsymbol{P}_{\boldsymbol{\theta}_0}$ as

$$\boldsymbol{P}_{\boldsymbol{\theta}_0} = \begin{pmatrix} \boldsymbol{I}_p & 0 & 0 \\ 0 & \boldsymbol{P}_{\boldsymbol{B}_0} & 0 \\ 0 & 0 & \boldsymbol{P}_{\boldsymbol{\Omega}_0} \end{pmatrix}$$

where $\boldsymbol{I}_p$ is the identity matrix spanning the tangent space of $\mathbb{R}^p$, which is identified with $\mathbb{R}^p$ itself. The blocks $\boldsymbol{P}_{\boldsymbol{B}_0}$ and $\boldsymbol{P}_{\boldsymbol{\Omega}_0}$ need to span the tangent spaces of $\mathfrak{K}_{\boldsymbol{B}}$ and $\mathfrak{CK}_{\boldsymbol{\Omega}}$, respectively. Both $\mathfrak{K}_{\boldsymbol{B}}$ and $\mathfrak{CK}_{\boldsymbol{\Omega}}$ are manifolds according to Theorem 8 under the cone condition. The constraint manifold $\mathfrak{CK}_{\boldsymbol{\Omega}}$ is the intersection of $\mathfrak{K}_{\boldsymbol{\Omega}}$ with the span of the projection $(\boldsymbol{T}_2\boldsymbol{D}_p^\dagger)^\dagger \boldsymbol{T}_2 \boldsymbol{D}_p^\dagger$ meaning that the differential $\operatorname{vec} \mathrm{d}\boldsymbol{\Omega}$ on $\mathfrak{CK}_{\boldsymbol{\Omega}}$ fulfills $\operatorname{vec} \mathrm{d}\boldsymbol{\Omega} = (\boldsymbol{T}_2\boldsymbol{D}_p^\dagger)^\dagger \boldsymbol{T}_2 \boldsymbol{D}_p^\dagger \operatorname{vec} \mathrm{d}\boldsymbol{\Omega}$. Now, we can apply Lemma 2 for $\mathfrak{K}_{\boldsymbol{B}}$ and $\mathfrak{K}_{\boldsymbol{\Omega}}$ which give

$$\begin{aligned} \boldsymbol{P}_{\boldsymbol{B}_0} &= \boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}[\boldsymbol{\Gamma}_{\boldsymbol{\beta}_1}\boldsymbol{\gamma}_{\boldsymbol{\beta}_1}, \ldots, \boldsymbol{\Gamma}_{\boldsymbol{\beta}_r}\boldsymbol{\gamma}_{\boldsymbol{\beta}_r}], \\ \boldsymbol{P}_{\boldsymbol{\Omega}_0} &= (\boldsymbol{T}_2\boldsymbol{D}_p^\dagger)^\dagger \boldsymbol{T}_2 \boldsymbol{D}_p^\dagger \boldsymbol{S}_{\boldsymbol{p},\boldsymbol{p}}[\boldsymbol{\Gamma}_{\boldsymbol{\Omega}_1}\boldsymbol{\gamma}_{\boldsymbol{\Omega}_1}, \ldots, \boldsymbol{\Gamma}_{\boldsymbol{\Omega}_r}\boldsymbol{\gamma}_{\boldsymbol{\Omega}_r}] \end{aligned}$$

where the matrices $\boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}$, $\boldsymbol{\Gamma}_{\boldsymbol{\beta}_j}$, $\boldsymbol{\gamma}_{\boldsymbol{\beta}_j}$, $\boldsymbol{\Gamma}_{\boldsymbol{\Omega}_j}$ and $\boldsymbol{\gamma}_{\boldsymbol{\Omega}_j}$ are described in Lemma 2 for the Kronecker

manifolds $\mathfrak{K}_{\boldsymbol{B}}$ and $\mathfrak{K}_{\boldsymbol{\Omega}}$. Leading to

$$\boldsymbol{P}_{\boldsymbol{\theta}_0} = \begin{pmatrix} \boldsymbol{I}_p & 0 & 0 \\ 0 & \boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}[\boldsymbol{\Gamma}_{\boldsymbol{\beta}_1}\boldsymbol{\gamma}_{\boldsymbol{\beta}_1}, \ldots, \boldsymbol{\Gamma}_{\boldsymbol{\beta}_r}\boldsymbol{\gamma}_{\boldsymbol{\beta}_r}] & 0 \\ 0 & 0 & (\boldsymbol{T}_2\boldsymbol{D}_p^{\dagger})^{\dagger}\boldsymbol{T}_2\boldsymbol{D}_p^{\dagger}\boldsymbol{S}_{\boldsymbol{p},\boldsymbol{p}}[\boldsymbol{\Gamma}_{\boldsymbol{\Omega}_1}\boldsymbol{\gamma}_{\boldsymbol{\Omega}_1}, \ldots, \boldsymbol{\Gamma}_{\boldsymbol{\Omega}_r}\boldsymbol{\gamma}_{\boldsymbol{\Omega}_r}] \end{pmatrix}. \tag{5.27}$$

□

5.4.2 Asymptotic Normality

The following is a reformulation of Bura et al. [BDF+18, Lemma 2.3] which assumes Condition 2.2 to hold. The existence of a mapping in Condition 2.2 is not needed for Lemma 2.3. It suffices that the restricted parameter space Θ is a subset of the unrestricted parameter space Ξ, which is trivially satisfied in our setting. Under this, Theorem 11 follows directly from Bura et al. [BDF+18, Lemma 2.3].

Theorem 11 (Existence of strong M-estimators on Subsets)**.** *Assume there exists a (weak or strong) M-estimator $\widehat{\boldsymbol{\xi}}_n$ for M_n over Ξ, then there exists a strong M-estimator $\widehat{\boldsymbol{\theta}}_n$ for M_n over any non-empty $\Theta \subseteq \Xi$.*

Proof. Let $\widehat{\boldsymbol{\xi}}_n$ be a (weak/strong) M-estimator for the unconstrained problem. This gives by definition, in any case, that

$$\sup_{\boldsymbol{\xi}\in\Xi} M_n(\boldsymbol{\xi}) \leq M_n(\widehat{\boldsymbol{\xi}}_n) + o_P(1).$$

Cause $\emptyset \neq \Theta \subseteq \Xi$ we have $\sup_{\boldsymbol{\theta}\in\Theta} M_n(\boldsymbol{\theta}) \leq \sup_{\boldsymbol{\xi}\in\Xi} M_n(\boldsymbol{\xi})$ and with $M_n(\boldsymbol{\xi}) < \infty$ for any $\boldsymbol{\xi} \in \Xi$

$$P\Big(\sup_{\boldsymbol{\theta}\in\Theta} M_n(\boldsymbol{\theta}) < \infty\Big) \geq P\Big(\sup_{\boldsymbol{\xi}\in\Xi} M_n(\boldsymbol{\xi}) < \infty\Big) \xrightarrow{n\to\infty} 1.$$

If $\sup_{\boldsymbol{\theta}\in\Theta} M_n(\boldsymbol{\theta}) < \infty$, then, for any $0 < \epsilon_n$ exists $\widehat{\boldsymbol{\theta}}_n \in \Theta$ such that $\sup_{\boldsymbol{\theta}\in\Theta} M_n(\boldsymbol{\theta}) - \epsilon_n \leq M_n(\widehat{\boldsymbol{\theta}}_n)$. Therefore, we can choose $\epsilon_n \in o(n^{-1})$, which yields

$$P\Big(M_n(\widehat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta}\in\Theta} M_n(\boldsymbol{\theta}) - o(n^{-1})\Big) \geq P\Big(\sup_{\boldsymbol{\theta}\in\Theta} M_n(\boldsymbol{\theta}) < \infty\Big) \xrightarrow{n\to\infty} 1.$$

The last statement states

$$M_n(\widehat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta}\in\Theta} M_n(\boldsymbol{\theta}) - o_P(n^{-1})$$

which is the definition of $\widehat{\boldsymbol{\theta}}_n$ being a strong M-estimator over Θ. □

Theorem 12 (Existence and Consistency of M-estimators on Subsets)**.** *Let Ξ be a convex open subset of a Euclidean space and $\Theta \subseteq \Xi$ non-empty. Assume $\boldsymbol{\xi} \mapsto m_{\boldsymbol{\xi}}(z)$ is a strictly concave function on Ξ for almost all z and $z \mapsto m_{\boldsymbol{\xi}}(z)$ is measurable for all $\boldsymbol{\xi} \in \Xi$. Let $M(\boldsymbol{\xi}) = \mathbb{E}\, m_{\boldsymbol{\xi}}(Z)$ be a well defined function with a unique maximizer $\boldsymbol{\theta}_0 \in \Theta \subseteq \Xi$; that is, $M(\boldsymbol{\theta}_0) > M(\boldsymbol{\xi})$ for all $\boldsymbol{\xi} \neq \boldsymbol{\theta}_0$. Also, assume*

$$\mathbb{E} \sup_{\boldsymbol{\xi}\in K} |m_{\boldsymbol{\xi}}(Z)| < \infty,$$

for every non-empty compact $K \subset \Xi$. Then, there exists a strong M-estimator $\widehat{\boldsymbol{\theta}}_n$ of $M_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n m_{\boldsymbol{\theta}}(Z_i)$ over the subset Θ. Moreover, any strong M-estimator $\widehat{\boldsymbol{\theta}}_n$ of M_n over Θ converges in probability to $\boldsymbol{\theta}_0$, that is $\widehat{\boldsymbol{\theta}}_n \xrightarrow{p} \boldsymbol{\theta}_0$.

Proof. It follows the proof of Bura et al. [BDF+18, Proposition 2.4] with the same assumptions. The only exception is we only require Θ to be a subset of Ξ. This is accounted for by replacing Lemma 2.3 in Bura et al. [BDF+18] with Theorem 11 to obtain the existence of a strong M-estimator on Θ. □

In the following we rewrite the log-likelihood (5.11) in a simpler form. This simplifies the proof of Theorem 10 as well as provides the notation to express the regularity conditions for Theorem 10 in a compact form.

Rewriting the first natural parameter component $\boldsymbol{\eta}_{1y}$ defined in (5.4) gives

$$\boldsymbol{\eta}_{1y} = \operatorname{vec}\overline{\eta} + \boldsymbol{B}\operatorname{vec}\mathcal{F}_y = \boldsymbol{I}_p \operatorname{vec}\overline{\eta} + ((\operatorname{vec}\mathcal{F}_y)^T \otimes \boldsymbol{I}_p)\operatorname{vec}\boldsymbol{B} = \begin{pmatrix}\boldsymbol{I}_p & (\operatorname{vec}\mathcal{F}_y)^T \otimes \boldsymbol{I}_p\end{pmatrix}\begin{pmatrix}\operatorname{vec}\overline{\eta} \\ \operatorname{vec}\boldsymbol{B}\end{pmatrix}.$$

For the second natural parameter component $\boldsymbol{\eta}_2$, modeled in (5.5), we have

$$\langle \boldsymbol{\eta}_2, \boldsymbol{T}_2 \operatorname{vech}((\operatorname{vec}\mathcal{X})(\operatorname{vec}\mathcal{X})^T)\rangle = \langle (\boldsymbol{T}_2\boldsymbol{D}_p^\dagger)^T\boldsymbol{\eta}_2, \operatorname{vec}(\mathcal{X}\circ\mathcal{X})\rangle = \langle c\,\boldsymbol{\Omega}, \mathcal{X}\circ\mathcal{X}\rangle$$

which means that

$$c\operatorname{vec}\boldsymbol{\Omega} = (\boldsymbol{T}_2\boldsymbol{D}_p^\dagger)^T\boldsymbol{\eta}_2.$$

The inverse relation is

$$\boldsymbol{\eta}_2 = c((\boldsymbol{T}_2\boldsymbol{D}_p^\dagger)^\dagger)^T\operatorname{vec}\boldsymbol{\Omega} = c((\boldsymbol{T}_2\boldsymbol{D}_p^\dagger)^\dagger)^T\boldsymbol{D}_p\operatorname{vech}\boldsymbol{\Omega},$$

describing the linear relation between $\boldsymbol{\eta}_2$ and $\operatorname{vech}\boldsymbol{\Omega}$. This gives the following relation between $\boldsymbol{\eta}_y = (\boldsymbol{\eta}_{1y}, \boldsymbol{\eta}_2)$ and $\boldsymbol{\xi} = (\operatorname{vec}\overline{\eta}, \operatorname{vec}\boldsymbol{B}, \operatorname{vech}\boldsymbol{\Omega}) \in \Xi$ as

$$\boldsymbol{\eta}_y = \begin{pmatrix}\boldsymbol{I}_p & (\operatorname{vec}\mathcal{F}_y)^T \otimes \boldsymbol{I}_p & 0 \\ 0 & 0 & c((\boldsymbol{T}_2\boldsymbol{D}_p^\dagger)^\dagger)^T\boldsymbol{D}_p\end{pmatrix}\begin{pmatrix}\operatorname{vec}\overline{\eta} \\ \operatorname{vec}\boldsymbol{B} \\ \operatorname{vech}\boldsymbol{\Omega}\end{pmatrix} =: \boldsymbol{F}(y)\boldsymbol{\xi} \tag{5.28}$$

where $\boldsymbol{F}(y)$ is a $(p+d)\times p(p+2q+3)/2$ dimensional matrix valued function in y. Moreover, for every y the matrix $\boldsymbol{F}(y)$ is of full rank $p+d$.

The log-likelihood of model (5.1) for the unconstrained parameters $\xi \in \Xi$ is

$$l_n(\boldsymbol{\xi}) = \frac{1}{n}\sum_{i=1}^n(\langle \boldsymbol{t}(\mathcal{X}), \boldsymbol{\eta}_y\rangle - b(\boldsymbol{\eta}_y)) =: \frac{1}{n}\sum_{i=1}^n m_{\boldsymbol{\xi}}(Z_i)$$

where $Z_i = (\mathcal{X}_i, Y_i)$. Using (5.28) we can write

$$m_{\boldsymbol{\xi}}(z) = \langle \boldsymbol{t}(\mathcal{X}), \boldsymbol{F}(y)\boldsymbol{\xi}\rangle - b(\boldsymbol{F}(y)\boldsymbol{\xi}).$$

The following are the regularity conditions for the log-likelihood required by Theorem 10.

Condition 3. The mapping $\boldsymbol{\xi} \mapsto m_{\boldsymbol{\xi}}(z)$ is twice continuously differentiable for almost every z and $z \mapsto m_{\boldsymbol{\xi}}(z)$ is measurable. Moreover, $\boldsymbol{\eta} \mapsto b(\boldsymbol{\eta})$ is strictly convex. Furthermore, for every $\widetilde{\boldsymbol{\eta}}$ holds $P(\boldsymbol{F}(Y)\boldsymbol{\xi} = \widetilde{\boldsymbol{\eta}}) < 1$.

Condition 4. It holds $\mathbb{E}\,\|\boldsymbol{t}(\mathcal{X})^T \boldsymbol{F}(Y)\| < \infty$ and $\mathbb{E}\,\|\boldsymbol{t}(\mathcal{X})^T \boldsymbol{F}(Y)\|^2 < \infty$.

Condition 5. The mapping $\boldsymbol{\eta} \mapsto b(\boldsymbol{\eta})$ is twice continuously differentiable and for every non-empty compact $K \subseteq \Xi$ holds

$$\mathbb{E} \sup_{\boldsymbol{\xi} \in K} \|b(\boldsymbol{F}(Y)\boldsymbol{\xi})\| < \infty, \qquad \mathbb{E} \sup_{\boldsymbol{\xi} \in K} \|\nabla b(\boldsymbol{F}(Y)\boldsymbol{\xi})^T \boldsymbol{F}(Y)\|^2 < \infty,$$
$$\mathbb{E} \sup_{\boldsymbol{\xi} \in K} \|\boldsymbol{F}(Y)^T \nabla^2 b(\boldsymbol{F}(Y)\boldsymbol{\xi}) \boldsymbol{F}(Y)\| < \infty.$$

The following is a technical Lemma used in the proof of Theorem 10.

Lemma 2. *Let $\mathfrak{A}_k \subseteq \mathbb{R}^{p_k \times q_k} \backslash \{\mathbf{0}\}$ for $k = 1, \dots, r$ be smooth embedded submanifolds as well as ether a sphere or a cone. Then*

$$\mathfrak{K} = \left\{ \bigotimes_{k=r}^{1} \boldsymbol{A}_k : \boldsymbol{A}_k \in \mathfrak{A}_k \right\}$$

is an embedded manifold in $\mathbb{R}^{p \times q}$ for $p = \prod_{k=1}^r p_k$ and $q = \prod_{k=1}^r q_k$. Furthermore, define for $j = 1, \dots, r$ the matrices

$$\boldsymbol{\Gamma}_j = \bigotimes_{k=r}^{1} (\boldsymbol{I}_{p_k q_k} \text{ if } j = k \text{ else } \operatorname{vec} \boldsymbol{A}_k) = \bigotimes_{k=r}^{j+1} (\operatorname{vec} \boldsymbol{A}_k) \otimes \boldsymbol{I}_{p_j q_j} \otimes \bigotimes_{k=j-1}^{1} (\operatorname{vec} \boldsymbol{A}_k) \tag{5.29}$$

and let γ_j be $p_j q_j \times d_j$ matrices with $d_j \geq \dim \mathfrak{A}_j$ which span the tangent space $T_{\boldsymbol{A}_j} \mathfrak{A}_j$ of $\mathfrak{A}$ at $\boldsymbol{A}_j \in \mathfrak{A}_j$, that is $\operatorname{span} \gamma_j = T_{\boldsymbol{A}_j} \mathfrak{A}_j$. Then, with the permutation matrix $\boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}$ defined in (2.17), the $pq \times \sum_{k=1}^r d_j$ dimensional matrix

$$\boldsymbol{P}_{\boldsymbol{A}} = \boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}} \left[\boldsymbol{\Gamma}_1 \gamma_1, \boldsymbol{\Gamma}_2 \gamma_2, \dots, \boldsymbol{\Gamma}_r \gamma_r \right]$$

spans the tangent space $T_{\boldsymbol{A}} \mathfrak{K}$ of $\mathfrak{K}$ at $\boldsymbol{A} = \bigotimes_{k=r}^{1} \boldsymbol{A}_k \in \mathfrak{K}$, in formula $\operatorname{span} \boldsymbol{P}_{\boldsymbol{A}} = T_{\boldsymbol{A}} \mathfrak{K}$.

Proof. The statement that $\mathfrak{K}$ is an embedded manifold follows via induction using Theorem 8.

We compute the differential of the vectorized Kronecker product using Lemma 1 where

$\boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}$ is the permutation (2.17) defined therein.

$$\begin{aligned}
&\mathrm{d}\operatorname{vec}\bigotimes_{k=r}^{1}\boldsymbol{A}_k = \operatorname{vec}\sum_{j=1}^{r}\bigotimes_{k=r}^{1}(\mathrm{d}\boldsymbol{A}_j \text{ if } k=j \text{ else } \boldsymbol{A}_k)\\
&= \boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}\operatorname{vec}\sum_{j=1}^{r}\Big(\bigcirc_{k=1}^{r}(\mathrm{d}\boldsymbol{A}_j \text{ if } k=j \text{ else } \boldsymbol{A}_k)\Big) = \boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}\sum_{j=1}^{r}\bigotimes_{k=r}^{1}(\operatorname{vec}\mathrm{d}\boldsymbol{A}_j \text{ if } k=j \text{ else } \operatorname{vec}\boldsymbol{A}_k)\\
&= \boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}\sum_{j=1}^{r}\Big(\bigotimes_{k=r}^{1}(\boldsymbol{I}_{p_jq_j} \text{ if } k=j \text{ else } \operatorname{vec}\boldsymbol{A}_k)\Big)\operatorname{vec}\mathrm{d}\boldsymbol{A}_j = \boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}\sum_{j=1}^{r}\boldsymbol{\Gamma}_j\operatorname{vec}\mathrm{d}\boldsymbol{A}_j\\
&= \boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}[\boldsymbol{\Gamma}_1,\dots,\boldsymbol{\Gamma}_r]\begin{pmatrix}\operatorname{vec}\mathrm{d}\boldsymbol{A}_1\\ \vdots\\ \operatorname{vec}\mathrm{d}\boldsymbol{A}_r\end{pmatrix}
\end{aligned}$$

Due to the definition of the manifold this differential provides the gradient of a surjective map into the manifold. The span of the gradient then spans the tangent space.

Now, we take a closer look at the differentials $\operatorname{vec}\mathrm{d}\boldsymbol{A}_j$ for $j=1,\dots,r$. Let φ_j be a chart of $\mathfrak{A}_j$ in a neighborhood of $\boldsymbol{A}_j$. Then, $\boldsymbol{A}_j = \varphi_j^{-1}(\varphi_j(\boldsymbol{A}_j))$ which gives

$$\operatorname{vec}\mathrm{d}\boldsymbol{A}_j = \nabla\varphi_j^{-1}(\varphi_j(\boldsymbol{A}_j))^T\operatorname{vec}\mathrm{d}\varphi_j(\boldsymbol{A}_j).$$

Therefore, for every matrix $\boldsymbol{\gamma}_j$ such that $\operatorname{span}\boldsymbol{\gamma}_j = T_{\boldsymbol{A}_j}\mathfrak{A}_j$ holds $\operatorname{span}\nabla\varphi_j^{-1}(\varphi_j(\boldsymbol{A}_j))^T = \operatorname{span}\boldsymbol{\gamma}_j$ by Definition 11 of the tangent space. We get

$$\operatorname{span}\boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}[\boldsymbol{\Gamma}_1,\dots,\boldsymbol{\Gamma}_r]\begin{pmatrix}\operatorname{vec}\mathrm{d}\boldsymbol{A}_1\\ \vdots\\ \operatorname{vec}\mathrm{d}\boldsymbol{A}_r\end{pmatrix} = \operatorname{span}\boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}[\boldsymbol{\Gamma}_1\boldsymbol{\gamma}_1,\dots,\boldsymbol{\Gamma}_r\boldsymbol{\gamma}_r] = \operatorname{span}\boldsymbol{P}_{\boldsymbol{A}}$$

which concludes the proof. □

Now, we are prepared to present the main result of this section. In combination with Theorem 9, this Theorem establishes the asymptotic normality of the GMLM estimate $\widehat{\boldsymbol{B}} = \bigotimes_{k=r}^{1}\widehat{\boldsymbol{\beta}}_k$ of the vectorized reduction matrix $\boldsymbol{B}$ in Theorem 5, our primary inference objective.

Theorem 13 (Asymptotic Normality for M-estimators on Manifolds)**.** *Let $\Theta \subseteq \mathbb{R}^p$ be a smooth embedded manifold. For each $\boldsymbol{\theta}$ in a neighborhood in $\mathbb{R}^p$ of the true parameter $\boldsymbol{\theta}_0 \in \Theta$ let $z \mapsto m_{\boldsymbol{\theta}}(z)$ be measurable and $\boldsymbol{\theta} \mapsto m_{\boldsymbol{\theta}}(z)$ be differentiable at $\boldsymbol{\theta}_0$ for almost all z. Assume also that there exists a measurable function u such that $\mathbb{E}[u(Z)^2] < \infty$, and for almost all z as well as all $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ in a neighborhood of $\boldsymbol{\theta}_0$ such that*

$$|m_{\boldsymbol{\theta}_1}(z) - m_{\boldsymbol{\theta}_2}(z)| \le u(z)\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2.$$

Moreover, assume that $\boldsymbol{\theta} \mapsto \mathbb{E}[m_{\boldsymbol{\theta}}(Z)]$ admits a second-order Taylor expansion at $\boldsymbol{\theta}_0$ in a neighborhood of $\boldsymbol{\theta}_0$ in $\mathbb{R}^p$ with a non-singular Hessian $\boldsymbol{H}_{\boldsymbol{\theta}_0} = \nabla^2_{\boldsymbol{\theta}}\mathbb{E}[m_{\boldsymbol{\theta}}(Z)]|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} \in \mathbb{R}^{p\times p}$.

If $\widehat{\boldsymbol{\theta}}_n$ is a strong M-estimator of $\boldsymbol{\theta}_0$ in Θ, then $\widehat{\boldsymbol{\theta}}_n$ is asymptotically normal

$$\sqrt{n}(\widehat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0) \xrightarrow{d} \mathcal{N}_p(\mathbf{0}, \boldsymbol{\Pi}_{\boldsymbol{\theta}_0} \mathbb{E}[\nabla_{\boldsymbol{\theta}} m_{\boldsymbol{\theta}_0}(Z)(\nabla_{\boldsymbol{\theta}} m_{\boldsymbol{\theta}_0}(Z))^T] \boldsymbol{\Pi}_{\boldsymbol{\theta}_0})$$

where $\boldsymbol{\Pi}_{\boldsymbol{\theta}_0} = \boldsymbol{P}_{\boldsymbol{\theta}_0}(\boldsymbol{P}_{\boldsymbol{\theta}_0}^T \boldsymbol{H}_{\boldsymbol{\theta}_0} \boldsymbol{P}_{\boldsymbol{\theta}_0})^\dagger \boldsymbol{P}_{\boldsymbol{\theta}_0}^T$ and $\boldsymbol{P}_{\boldsymbol{\theta}_0}$ is any matrix whose span is the tangent space $T_{\boldsymbol{\theta}_0}\Theta$ of Θ at $\boldsymbol{\theta}_0$.

Proof. Let $\varphi : U \to \varphi(U)$ be a coordinate chart[7] with $\boldsymbol{\theta}_0 \in U \subseteq \Theta$. As φ is continuous we get with the continuous mapping theorem on metric spaces van der Vaart [van98, Thm 18.11] that $\varphi(\widehat{\boldsymbol{\theta}}_n) \xrightarrow{p} \varphi(\boldsymbol{\theta}_0)$ which implies $P(\varphi(\widehat{\boldsymbol{\theta}}_n) \in \varphi(U)) \xrightarrow{n\to\infty} 1$.

The next step is to apply van der Vaart [van98, Thm 5.23] to $\widehat{\boldsymbol{s}}_n = \varphi(\widehat{\boldsymbol{\theta}}_n)$. Therefore, assume that $\widehat{\boldsymbol{s}}_n \in \varphi(U)$. Denote with $\boldsymbol{s} = \varphi(\boldsymbol{\theta}) \in \varphi(U) \subseteq \mathbb{R}^d$ the coordinates of the parameter $\boldsymbol{\theta} \in U \subseteq \Theta$ of the $d = \dim(\Theta)$ dimensional manifold $\Theta \subseteq \mathbb{R}^p$. With $\varphi : U \to \varphi(U)$ being bijective, we can express $m_{\boldsymbol{\theta}}$ in terms of $\boldsymbol{s} = \varphi(\boldsymbol{\theta})$ for every $\boldsymbol{\theta} \in U$ as $m_{\boldsymbol{\theta}} = m_{\varphi^{-1}(\boldsymbol{s})}$. Furthermore, denote

$$M(\boldsymbol{\theta}) = \mathbb{E}[m_{\boldsymbol{\theta}}(Z)] \qquad \text{and} \qquad M_\varphi(\boldsymbol{s}) = \mathbb{E}[m_{\varphi^{-1}(\boldsymbol{s})}(Z)] = M(\varphi^{-1}(\boldsymbol{s})).$$

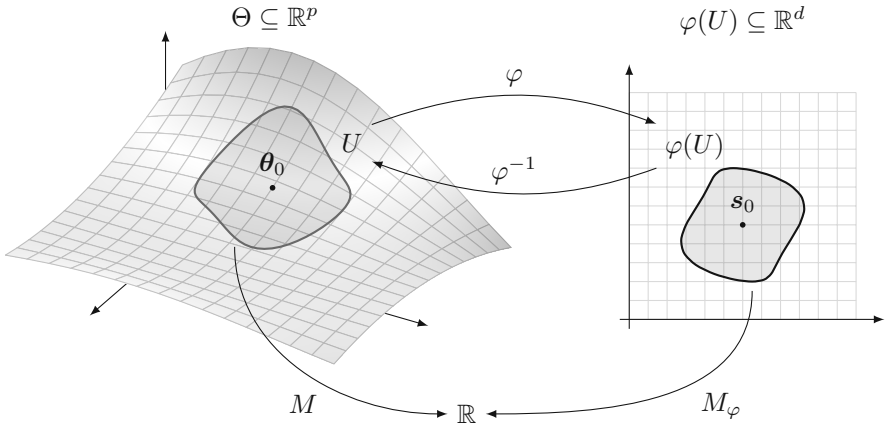

Figure 5.4: Depiction of the notation used in the proof of Theorem 13. Example with $p = 3$ and $d = \dim(\Theta) = 2$.

By assumption, the function $M(\boldsymbol{\theta})$ is twice continuously differentiable in an neighborhood[8] of $\boldsymbol{\theta}_0$. W.l.o.g. we can assume that U is contained in that neighborhood. Then, using the chain rule, we get the gradient of M_φ at $\boldsymbol{s}_0$ to be $\mathbf{0}$ by

$$\nabla M_\varphi(\boldsymbol{s}_0) = \nabla\varphi^{-1}(\boldsymbol{s}_0)\nabla M(\varphi^{-1}(\boldsymbol{s}_0)) = \nabla\varphi^{-1}(\boldsymbol{s}_0)\nabla M(\boldsymbol{\theta}_0) = \nabla\varphi^{-1}(\boldsymbol{s}_0)\mathbf{0} = \mathbf{0}$$

because $\boldsymbol{\theta}_0 = \varphi^{-1}(\boldsymbol{s}_0)$ is a maximizer of M. For the second-derivative, evaluated at $\boldsymbol{s}_0 = \varphi(\boldsymbol{\theta}_0)$, we have

$$\nabla^2 M_\varphi(\boldsymbol{s}_0) = \nabla\varphi^{-1}(\boldsymbol{s}_0)\nabla^2 M(\varphi^{-1}(\boldsymbol{s}_0))\nabla\varphi^{-1}(\boldsymbol{s}_0)^T = \nabla\varphi^{-1}(\boldsymbol{s}_0)\boldsymbol{H}_{\boldsymbol{\theta}_0}\nabla\varphi^{-1}(\boldsymbol{s}_0)^T$$

[7] By Definition 10, the chart $\varphi : U \to \varphi(U)$ is bi-continuous, is infinitely often continuously differentiable, and has a continuously differentiable inverse $\varphi^{-1} : \varphi(U) \to U$. Furthermore, the domain U is open according to the trace topology on Θ, that means that their exists an open set $O \subseteq \mathbb{R}^p$ such that $U = \Theta \cap O$.

[8] A set N is called a neighborhood of u if there exists an open set O such that $u \in O \subseteq N$.

using $\nabla M_\varphi(\boldsymbol{s}_0) = \mathbf{0}$. This gives the second-order Taylor expansion of M_φ at $\boldsymbol{s}_0$ as

$$M_\varphi(\boldsymbol{s}) = M_\varphi(\boldsymbol{s}_0) + \frac{1}{2}(\boldsymbol{s}-\boldsymbol{s}_0)^T \nabla^2 M_\varphi(\boldsymbol{s}_0)(\boldsymbol{s}-\boldsymbol{s}_0) + \mathcal{O}(\|\boldsymbol{s}-\boldsymbol{s}_0\|^3)$$

We also need to check the local Lipschitz condition of $m_{\varphi^{-1}(\boldsymbol{s})}$. Therefore, let $V_\epsilon(\boldsymbol{s}_0) = \{\boldsymbol{s} \in \mathbb{R}^d : \|\boldsymbol{s}-\boldsymbol{s}_0\| < \epsilon\}$ be the open ϵ-ball with center $\boldsymbol{s}_0$. Since $\varphi(U)$ contains $\boldsymbol{s}_0$, and is open in $\mathbb{R}^d$, there exists an $\epsilon > 0$ such that $V_\epsilon(\boldsymbol{s}_0) \subseteq \varphi(U)$. Then, the closed $\epsilon/2$ ball $\overline{V}_{\epsilon/2}(\boldsymbol{s}_0)$ is a neighborhood of $\boldsymbol{s}_0$ and the supremum $\sup_{\boldsymbol{s}\in\overline{V}_{\epsilon/2}(\boldsymbol{s}_0)} \|\nabla\varphi^{-1}(\boldsymbol{s})\| < \infty$ due to the continuouty of $\nabla\varphi^{-1}$ on $\varphi(U)$ with $\overline{V}_{\epsilon/2}(\boldsymbol{s}_0) \subset V_\epsilon(\boldsymbol{s}_0) \subseteq \varphi(U)$. Then, for almost every z and every $\boldsymbol{s}_1 = \varphi(\boldsymbol{\theta}_1), \boldsymbol{s}_2 = \varphi(\boldsymbol{\theta}_2) \in \overline{V}_{\epsilon/2}(\boldsymbol{s}_0)$ holds

$$\begin{aligned}|m_{\varphi^{-1}(\boldsymbol{s}_1)}(z) - m_{\varphi^{-1}(\boldsymbol{s}_2)}(z)| = |m_{\boldsymbol{\theta}_1}(z) - m_{\boldsymbol{\theta}_2}(z)| &\overset{(a)}{\leq} u(z)\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\| = u(z)\|\varphi^{-1}(\boldsymbol{s}_1) - \varphi^{-1}(\boldsymbol{s}_2)\| \\ &\overset{(b)}{\leq} u(z) \sup_{\boldsymbol{s}\in\overline{V}_{\epsilon/2}(\boldsymbol{s}_0)} \|\nabla\varphi^{-1}(\boldsymbol{s})\|\|\boldsymbol{s}_1 - \boldsymbol{s}_2\| =: v(z)\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|.\end{aligned}$$

Here, (a) holds by assumption and (b) is a result of the mean value theorem. Now, $v(z)$ is measurable and square integrable as a scaled version of $u(z)$. Finally, with φ being one-to-one, we get that $\widehat{\boldsymbol{s}}_n = \varphi(\widehat{\boldsymbol{\theta}}_n)$ is a strong M-estimator for $\boldsymbol{s}_0 = \varphi(\boldsymbol{\theta}_0)$ of the objective M_φ. Now, we apply van der Vaart [van98, Thm 5.23] to get the asymptotic normality of $\widehat{\boldsymbol{s}}_n$ as

$$\sqrt{n}(\widehat{\boldsymbol{s}}_n - \boldsymbol{s}_0) \xrightarrow{d} \mathcal{N}_d(0, \boldsymbol{\Sigma}_{\boldsymbol{s}_0})$$

where the $d \times d$ variance-covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{s}_0}$ is given by

$$\boldsymbol{\Sigma}_{\boldsymbol{s}_0} = (\nabla^2 M_\varphi(\boldsymbol{s}_0))^{-1}\, \mathbb{E}[\nabla_{\boldsymbol{s}} m_{\varphi^{-1}(\boldsymbol{s}_0)}(Z)(\nabla_{\boldsymbol{s}} m_{\varphi^{-1}(\boldsymbol{s}_0)}(Z))^T](\nabla^2 M_\varphi(\boldsymbol{s}_0))^{-1}.$$

An application of the delta method yields

$$\sqrt{n}(\widehat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0) = \sqrt{n}(\varphi^{-1}(\widehat{\boldsymbol{s}}_n) - \varphi^{-1}(\boldsymbol{s}_0)) \xrightarrow{d} \mathcal{N}_p(0, \nabla\varphi^{-1}(\boldsymbol{s}_0)^T \boldsymbol{\Sigma}_{\boldsymbol{s}_0} \nabla\varphi^{-1}(\boldsymbol{s}_0)).$$

We continue by reexpressing the $p \times p$ asymtotic variance-covariance matrix of $\widehat{\boldsymbol{\theta}}_n$ in terms of $\boldsymbol{\theta}_0$ instead of $\boldsymbol{s}_0 = \varphi(\boldsymbol{\theta}_0)$. Therefore, let $\boldsymbol{\Phi}_{\boldsymbol{\theta}_0} = \nabla\varphi^{-1}(\varphi(\boldsymbol{\theta}_0))^T = \nabla\varphi^{-1}(\boldsymbol{s}_0)^T$ and observe that for all $\boldsymbol{s} \in \varphi(U)$, the gradient of $\boldsymbol{s} \mapsto m_{\varphi^{-1}(\boldsymbol{s})}(z)$ evaluated at $\boldsymbol{s}_0 = \varphi(\boldsymbol{\theta}_0)$ has the form

$$\nabla_{\boldsymbol{s}} m_{\varphi^{-1}(\boldsymbol{s}_0)}(z) = \nabla\varphi^{-1}(\boldsymbol{s}_0)\nabla_{\boldsymbol{\theta}} m_{\boldsymbol{\theta}_0}(z) = \boldsymbol{\Phi}_{\boldsymbol{\theta}_0}^T \nabla_{\boldsymbol{\theta}} m_{\boldsymbol{\theta}_0}(z).$$

Then

$$\begin{aligned}&\nabla\varphi^{-1}(\boldsymbol{s}_0)^T \boldsymbol{\Sigma}_{\boldsymbol{s}_0} \nabla\varphi^{-1}(\boldsymbol{s}_0) \\ &= \nabla\varphi^{-1}(\boldsymbol{s}_0)^T (\nabla^2 M_\varphi(\boldsymbol{s}_0))^{-1}\, \mathbb{E}[\nabla m_{\varphi^{-1}(\boldsymbol{s}_0)}(Z)(\nabla m_{\varphi^{-1}(\boldsymbol{s}_0)}(Z))^T](\nabla^2 M_\varphi(\boldsymbol{s}_0))^{-1} \nabla\varphi^{-1}(\boldsymbol{s}_0) \\ &\quad = \boldsymbol{\Phi}_{\boldsymbol{\theta}_0}(\boldsymbol{\Phi}_{\boldsymbol{\theta}_0}^T \boldsymbol{H}_{\boldsymbol{\theta}_0} \boldsymbol{\Phi}_{\boldsymbol{\theta}_0})^{-1} \boldsymbol{\Phi}_{\boldsymbol{\theta}_0}^T\, \mathbb{E}[\nabla_{\boldsymbol{\theta}} m_{\boldsymbol{\theta}_0}(Z)(\nabla_{\boldsymbol{\theta}} m_{\boldsymbol{\theta}_0}(Z))^T] \boldsymbol{\Phi}_{\boldsymbol{\theta}_0}(\boldsymbol{\Phi}_{\boldsymbol{\theta}_0}^T \boldsymbol{H}_{\boldsymbol{\theta}_0} \boldsymbol{\Phi}_{\boldsymbol{\theta}_0})^{-1} \boldsymbol{\Phi}_{\boldsymbol{\theta}_0}^T \\ &\quad = \boldsymbol{\Pi}_{\boldsymbol{\theta}_0}\, \mathbb{E}[\nabla_{\boldsymbol{\theta}} m_{\boldsymbol{\theta}_0}(Z)(\nabla_{\boldsymbol{\theta}} m_{\boldsymbol{\theta}_0}(Z))^T] \boldsymbol{\Pi}_{\boldsymbol{\theta}_0}\end{aligned}$$

where the last equality holds by $\operatorname{span} \boldsymbol{\Phi}_{\boldsymbol{\theta}_0} = T_{\boldsymbol{\theta}_0}\Theta$ by Definition 11 of the tangent space $T_{\boldsymbol{\theta}_0}\Theta$.

It remains to show that $\boldsymbol{\Pi}_{\boldsymbol{\theta}_0} = \boldsymbol{P}_{\boldsymbol{\theta}_0}(\boldsymbol{P}_{\boldsymbol{\theta}_0}^T \boldsymbol{H}_{\boldsymbol{\theta}_0} \boldsymbol{P}_{\boldsymbol{\theta}_0})^\dagger \boldsymbol{P}_{\boldsymbol{\theta}_0}^T$ for any $p \times k$ matrix $\boldsymbol{P}_{\boldsymbol{\theta}_0}$ such that $k \geq d$ and $\operatorname{span} \boldsymbol{P}_{\boldsymbol{\theta}_0} = T_{\boldsymbol{\theta}_0}\Theta$. This also ensures that the final result is independent of the chosen chart φ, since the tangent space does not depend on a specific chart. Therefore, let $\boldsymbol{\Phi}_{\boldsymbol{\theta}_0} = \boldsymbol{Q}\boldsymbol{R}$ and $\boldsymbol{P}_{\boldsymbol{\theta}_0} = \widetilde{\boldsymbol{Q}}\widetilde{\boldsymbol{R}}$ be their thin QR decompositions, respectively. Both $\boldsymbol{Q}, \widetilde{\boldsymbol{Q}}$ have dimension $p \times d$ With $\boldsymbol{Q}$ being semi-orthogonal, $\boldsymbol{R}$ is invertible of dimension $d \times d$ while $\widetilde{\boldsymbol{R}}$ is a $d \times k$ full row-rank matrix. With $\boldsymbol{Q}$ being semi-orthogonal the $p \times p$ matrix $\boldsymbol{Q}\boldsymbol{Q}^T$ is an orthogonal projection onto $\operatorname{span} \boldsymbol{Q} = \operatorname{span} \boldsymbol{P}_{\boldsymbol{\theta}_0} = T_{\boldsymbol{\theta}_0}\Theta$. This allows to express $\boldsymbol{P}_{\boldsymbol{\theta}_0}$ in terms of $\boldsymbol{Q}$ as

$$\boldsymbol{P}_{\boldsymbol{\theta}_0} = \boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{P}_{\boldsymbol{\theta}_0} = \boldsymbol{Q}\boldsymbol{Q}^T\widetilde{\boldsymbol{Q}}\widetilde{\boldsymbol{R}} =: \boldsymbol{Q}\boldsymbol{M}.$$

From $\operatorname{span} \boldsymbol{Q} = \operatorname{span} \boldsymbol{P}_{\boldsymbol{\theta}_0}$ follows that the $d \times k$ matrix $\boldsymbol{M}$ is also of full row-rank. We get $\boldsymbol{M}\boldsymbol{M}^\dagger = \boldsymbol{I}_d = \boldsymbol{R}\boldsymbol{R}^{-1}$ as a property of the Moore-Penrose pseudo inverse with $\boldsymbol{M}$ being of full row-rank. Another property of the pseudo inverse is that for matrices $\boldsymbol{A}, \boldsymbol{B}$, where $\boldsymbol{A}$ has full column-rank and $\boldsymbol{B}$ has full row-rank, holds $(\boldsymbol{A}\boldsymbol{B})^\dagger = \boldsymbol{B}^\dagger\boldsymbol{A}^\dagger$. This enables the computation

$$\begin{aligned}\boldsymbol{P}_{\boldsymbol{\theta}_0}(\boldsymbol{P}_{\boldsymbol{\theta}_0}^T \boldsymbol{H}_{\boldsymbol{\theta}_0} \boldsymbol{P}_{\boldsymbol{\theta}_0})^\dagger \boldsymbol{P}_{\boldsymbol{\theta}_0}^T &= \boldsymbol{Q}\boldsymbol{M}\boldsymbol{M}^\dagger(\boldsymbol{Q}^T\boldsymbol{H}_{\boldsymbol{\theta}_0}\boldsymbol{Q})^{-1}(\boldsymbol{M}\boldsymbol{M}^\dagger)^T\boldsymbol{Q}^T \\ &= \boldsymbol{Q}\boldsymbol{R}\boldsymbol{R}^{-1}(\boldsymbol{Q}^T\boldsymbol{H}_{\boldsymbol{\theta}_0}\boldsymbol{Q})^{-1}(\boldsymbol{R}\boldsymbol{R}^{-1})^T\boldsymbol{Q}^T = \boldsymbol{\Phi}_{\boldsymbol{\theta}_0}(\boldsymbol{\Phi}_{\boldsymbol{\theta}_0}^T \boldsymbol{H}_{\boldsymbol{\theta}_0} \boldsymbol{\Phi}_{\boldsymbol{\theta}_0})^{-1}\boldsymbol{\Phi}_{\boldsymbol{\theta}_0}^T = \boldsymbol{\Pi}_{\boldsymbol{\theta}_0}.\end{aligned}$$

□

Remark 22. Theorem 13 has as special case Theorem 5.23 in van der Vaart [van98], when Θ is an open subset of a Euclidean space, which is the simplest form of an embedded manifold.

5.5 Simulations

In this section we report simulation results for the tensor normal and the Ising model where different aspects of the GMLM model are compared against other methods. The comparison methods are Tensor Sliced Inverse Regression (TSIR) [DC15], MGCCA [CKT21; GGL+24] and the Tucker decomposition that is a higher-order form of principal component analysis (HOPCA) Kolda and Bader [KB09], for both continuous and binary data. For the latter, the binary values are treated as continuous. As a base line we also include classic PCA on vectorized observations. In case of the Ising model, we also compare with LPCA (Logistic PCA) and CLPCA (Convex Logistic PCA), both introduced in Landgraf and Lee [LL20]. All experiments are performed at sample size $n = 100, 200, 300, 500$ and 750. Every experiment is repeated 100 times.

We are interested in the quality of the estimate of the true sufficient reduction $\mathcal{R}(\mathcal{X})$ from Theorem 5. Therefore, we compare with the true vectorized reduction matrix $\boldsymbol{B} = \bigotimes_{k=r}^{1} \boldsymbol{\beta}_k$, as it is compatible with any linear reduction method. The distance $d(\boldsymbol{B}, \widehat{\boldsymbol{B}})$ between $\boldsymbol{B} \in \mathbb{R}^{p \times q}$ and an estimate $\widehat{\boldsymbol{B}} \in \mathbb{R}^{p \times \tilde{q}}$ is the *subspace distance* which is proportional to

$$d(\boldsymbol{B}, \widehat{\boldsymbol{B}}) \propto \|\boldsymbol{B}(\boldsymbol{B}^T\boldsymbol{B})^\dagger\boldsymbol{B}^T - \widehat{\boldsymbol{B}}(\widehat{\boldsymbol{B}}^T\widehat{\boldsymbol{B}})^\dagger\widehat{\boldsymbol{B}}^T\|_F,$$

the Frobenius norm of the difference between the projections onto the span of $\boldsymbol{B}$ and $\widehat{\boldsymbol{B}}$. The proportionality constant[9] of $d(\boldsymbol{B}, \widehat{\boldsymbol{B}})$ ensures that the subspace distance is in the interval $[0, 1]$. A distance of zero implies space overlap, a distance of one means that the subspaces are orthogonal.

5.5.1 Tensor Normal

We generate a random sample y_i, $i = 1, \ldots, n$, from the standard normal distribution. We then draw i.i.d. samples $\mathcal{X}_i$ for $i = 1, ..., n$ from the conditional tensor normal distribution of $\mathcal{X} \mid Y = y_i$. The conditional distribution $\mathcal{X} \mid Y = y_i$ depends on the choice of the GMLM parameters $\overline{\eta}$, $\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_r$, $\boldsymbol{\Omega}_1, ..., \boldsymbol{\Omega}_r$, and the function $\mathcal{F}_y$ of y. In all experiments we set $\overline{\eta} = \mathbf{0}$. The other parameters and $\mathcal{F}_y$ are described per experiment. With the true GMLM parameters and $\mathcal{F}_y$ given, we compute the conditional tensor normal mean $\mu_y = \mathcal{F}_y \times_{k=1}^{r} \boldsymbol{\Omega}_k^{-1} \boldsymbol{\beta}_k$ and covariances $\boldsymbol{\Sigma}_k = \boldsymbol{\Omega}_k^{-1}$ as in (5.13).

We consider the following settings:

1a) $\mathcal{X}$ is a three-way ($r = 3$) array of dimension $2 \times 3 \times 5$, and $\mathcal{F}_y \equiv y$ is a $1 \times 1 \times 1$ tensor. The true $\boldsymbol{\beta}_k$'s are all equal to $\boldsymbol{e}_1 \in \mathbb{R}^{p_k}$, the first unit vector, for $k \in \{1, 2, 3\}$. The matrices $\boldsymbol{\Omega}_k = \mathrm{AR}(0.5)$ follow an auto-regression like structure. That is, the elements are given by $(\boldsymbol{\Omega}_k)_{ij} = 0.5^{|i-j|}$.

1b) $\mathcal{X}$ is a three-way ($r = 3$) array of dimension $2 \times 3 \times 5$, and relates to the response y via a qubic polynomial. This is modeled via $\mathcal{F}_y$ of dimension $2 \times 2 \times 2$ by the twice iterated outer product of the vector $(1, y)$. Element wise this reads $(\mathcal{F}_y)_{ijk} = y^{i+j+k-3}$. All $\boldsymbol{\beta}_k$'s are set to $(\boldsymbol{e}_1, \boldsymbol{e}_2) \in \mathbb{R}^{p_k \times 2}$ with $\boldsymbol{e}_i$ the ith unit vector and the $\boldsymbol{\Omega}_k$'s are AR(0.5).

1c) Same as 1b), except that the GMLM parameters $\boldsymbol{\beta}_k$ are rank 1 given by

$$\boldsymbol{\beta}_1 = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \quad \boldsymbol{\beta}_2 = \begin{pmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \boldsymbol{\beta}_3 = \begin{pmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \end{pmatrix}.$$

1d) Same as 1b), but the true $\boldsymbol{\Omega}_k$ is tri-diagonal, for $k = 1, 2, 3$. Their elements are given by $(\boldsymbol{\Omega}_k)_{ij} = \delta_{0,|i-j|} + 0.5\delta_{1,|i-j|}$ with $\delta_{i,j}$ being the Kronecker delta.

1e) For the misspecification model we let $\mathcal{X} \mid Y$ be multivariate but *not* tensor normal. Let $\mathcal{X}$ be a 5×5 random matrix with normal entries, Y univariate standard normal and $\boldsymbol{f}_y$ a 4 dimensional vector given by $\boldsymbol{f}_y = (1, \sin(y), \cos(y), \sin(y)\cos(y))$. The true vectorized reduction matrix $\boldsymbol{B}$ is 25×4 consisting of the first 4 columns of the identity; i.e., $\boldsymbol{B}_{ij} = \delta_{ij}$. The variance-covariance matrix $\boldsymbol{\Sigma}$ has elements $\boldsymbol{\Sigma}_{ij} = 0.5^{|i-j|}$. Both, $\boldsymbol{B}$ and $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ violate the Kronecker product assumptions (5.6) and (5.7) of the GMLM model. Then, we set

$$\operatorname{vec} \mathcal{X} \mid (Y = y) = \boldsymbol{B}\boldsymbol{f}_y + \mathcal{N}_{25}(\mathbf{0}, \boldsymbol{\Sigma}).$$

[9] Depends on row dimension p and the ranks of $\boldsymbol{B}$ and $\widehat{\boldsymbol{B}}$ given by $(\min(\operatorname{rank} \boldsymbol{B} + \operatorname{rank} \widehat{\boldsymbol{B}}, 2p - (\operatorname{rank} \boldsymbol{B} + \operatorname{rank} \widehat{\boldsymbol{B}})))^{-1/2}$.

Furthermore, we fit the model with the wrong "known" function $\mathcal{F}_y$. We set $\mathcal{F}_y$ to be a 2×2 matrix with $(\mathcal{F}_y)_{ij} = y^{i+j-2}$, $i, j = 1, 2$.

The final tensor normal experiment 1e) is a misspecified model to explore the robustness of our approach. The true model does *not* have a Kronecker structure and the "known" function $\mathcal{F}_y$ of y is misspecified as well.

The results are visualized in Figure 5.5. Simulation 1a), given a 1D linear relation between the response Y and $\mathcal{X}$, TSIR and GMLM are equivalent. This is expected as Ding and Cook [DC15] already established that TSIR gives the MLE estimate under a tensor (matrix) normal distributed setting. For the other methods, MGCCA is only a bit better than PCA which, unexpectedly, beats HOPCA. But none of them are close to the performance of TSIR or GMLM. Continuing with 1b), where we introduced a qubic relation between Y and $\mathcal{X}$, we observe a bigger deviation in the performance of GMLM and TSIR. This is caused mainly because we are estimating an 8 dimensional subspace now, which amplifies the small performance boost, in the subspace distance, we gain by avoiding slicing. The results of 1c) are surprising. The GMLM model behaves as expected, clearly being the best. The first surprise is that PCA, HOPCA and MGCCA are visually indistinguishable. This is explained by a high signal to noise ration in this particular example. But the biggest surprise is the failure of TSIR. Even more surprising is that the conditional distribution $\mathcal{X} \mid Y$ is tensor normal distributed which in conjunction with $\operatorname{Cov}(\operatorname{vec} \mathcal{X})$ having a Kronecker structure, should give the MLE estimate. The low-rank assumption is also not an issue, this simply relates to TSIR estimating a 1D linear reduction which fulfills all the requirements. Finally, a common known issue of slicing, used in TSIR, is that conditional multi-modal distributions can cause estimation problems due to the different distribution modes leading to vanishing slice means. Again, this is not the case in simulation 1c). An investigation into this behaviour revealed the problem in the estimation of the mode covariance matrices $\boldsymbol{O}_k = \mathbb{E}[(\mathcal{X} - \mathbb{E}\,\mathcal{X})_{(k)}(\mathcal{X} - \mathbb{E}\,\mathcal{X})_{(k)}^T]$. The mode wise reductions provided by TSIR are computed as $\widehat{\boldsymbol{O}}_k^{-1}\widehat{\boldsymbol{\Gamma}}_k$ where the poor estimation of $\widehat{\boldsymbol{O}}_k$ causes the failure of TSIR. The poor estimate of $\boldsymbol{O}_k$ is rooted in the high signal to noise ratio in this particular simulation. GMLM does not have degenerate behaviour for high signal to noise ratios but it is less robust in low signal to noise ratio setting where TSIR performs better in this specific example. Simulation 1d), incorporating information about the covariance structure behaves similar to 1b), except that GMLM gains a statistically significant lead in estimation performance. The last simulation, 1e), where the model was misspecified for GMLM. GMLM, TSIR as well as MGCCA are on par where GMLM has a sligh lead in the small sample size setting and MGCCA overtakes in higher sample scenarios. The PCA and HOPCA methods both still outperformed. A wrong assumption about the relation to the response is still better than no relation at all.

5.5.2 Ising Model

Assuming for $\mathcal{X}$ being a 2×3 dimensional binary matrix with conditional matrix (tensor) Ising distribution $\mathcal{X} \mid Y$ as in Section 5.2.2. We let for $i = 1, \ldots, n$ the response being i.i.d. uniformly distributed in $[-1, 1]$ establishing the conditional value in the i.i.d. samples from

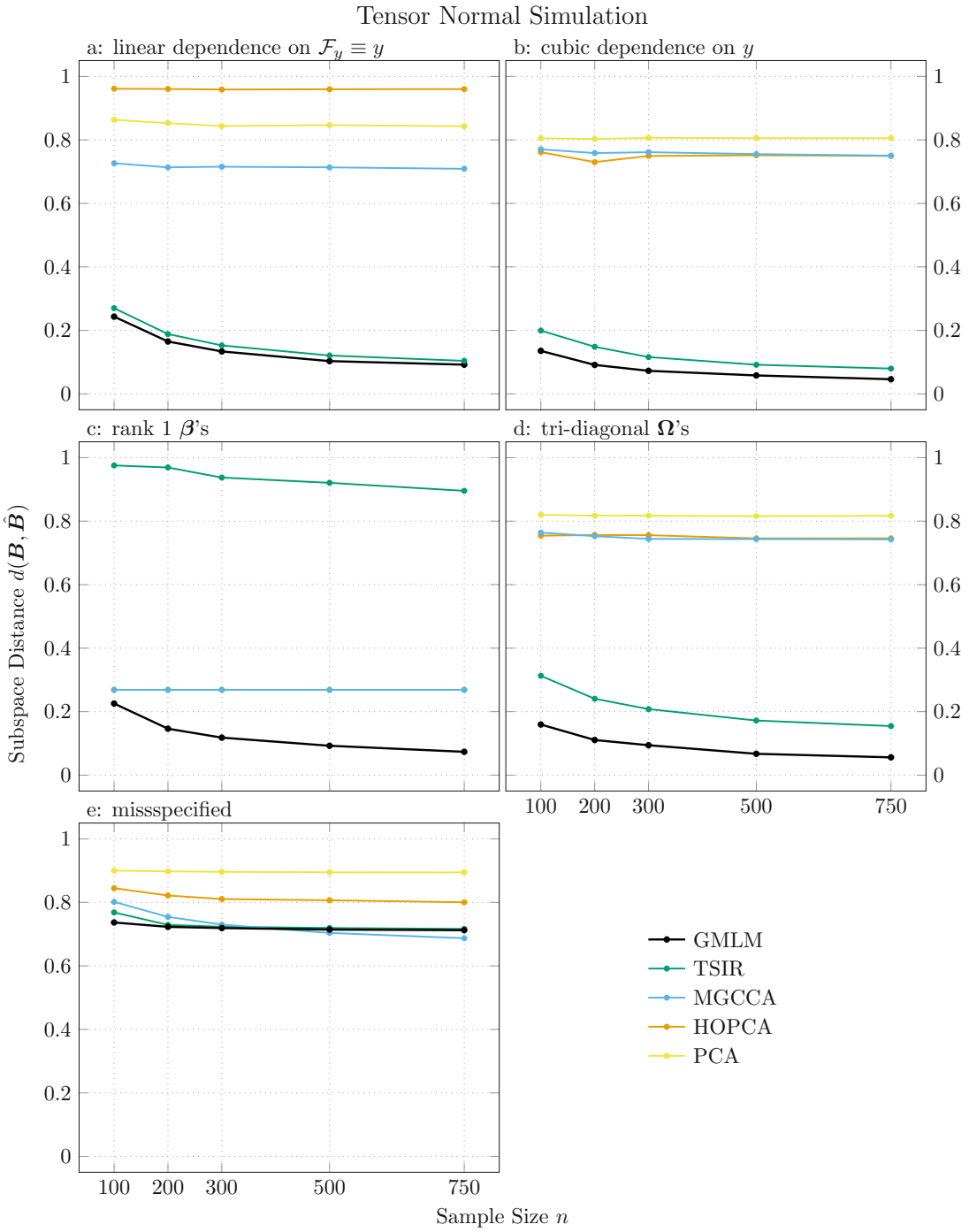

Figure 5.5: Visualization of the simulation results for the tensor normal GMLM. Sample size on the x-axis and the mean of subspace distance $d(\boldsymbol{B}, \widehat{\boldsymbol{B}})$ over 100 replications on the y-axis. Described in Section 5.5.1.

$\mathcal{X}_i \mid Y = y_i$ with GMLM parameters $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2$, $\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2$. We let

$$\boldsymbol{\beta}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \boldsymbol{\beta}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

and

$$\boldsymbol{\Omega}_1 = \begin{pmatrix} 0 & -2 \\ -2 & 0 \end{pmatrix}, \qquad \boldsymbol{\Omega}_2 = \begin{pmatrix} 1 & 0.5 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{pmatrix}$$

as well as

$$\mathcal{F}_y = \begin{pmatrix} \sin(\pi y) & -\cos(\pi y) \\ \cos(\pi y) & \sin(\pi y) \end{pmatrix}$$

if not mentioned otherwise in a specific simulation setup given next.

2a) A fully linear relation to the response set to be $\mathcal{F}_y \equiv y$ being a 1×1 matrix with $\boldsymbol{\beta}_1^T = (1, 0)$ and $\boldsymbol{\beta}_2^T = (1, 0, 0)$.

2b) The "base" simulation with all parameters as described above.

2c) Low rank regression with both $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ of rank 1 chosen as

$$\boldsymbol{\beta}_1 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \qquad \boldsymbol{\beta}_2 = \begin{pmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 0 \end{pmatrix}.$$

2d) We conclude with a simulation relating to the original design of the Ising model. It is a mathematical model to study the behaviour of Ferromagnetism Ising [Isi25] in a thermodynamic setting modeling the interaction effects of elementary magnets (spin up/down relating to 0 and 1). The model assumes all elementary magnets to be the same, which translates to all having the same coupling strength (two-way interactions) governed by a single parameter relating to the temperature of the system. Assuming the magnets to be arranged in a 2D grid (matrix valued $\mathcal{X}$), their interactions are constraint to direct neighbours. We can model this by choosing the true $\boldsymbol{\Omega}_k$'s to be tri-diagonal matrices with zero diagonal entries and all non-zero entries identical. Since this is a 1D matrix manifold, we can enforce the constraint. Setting the true interaction parameters to be

$$\boldsymbol{\Omega}_1 = \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \boldsymbol{\Omega}_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

where $1/2$ relates to an arbitrary temperature. The mean effect depending on $\mathcal{F}_y$ can be interpreted as an external magnetic field.

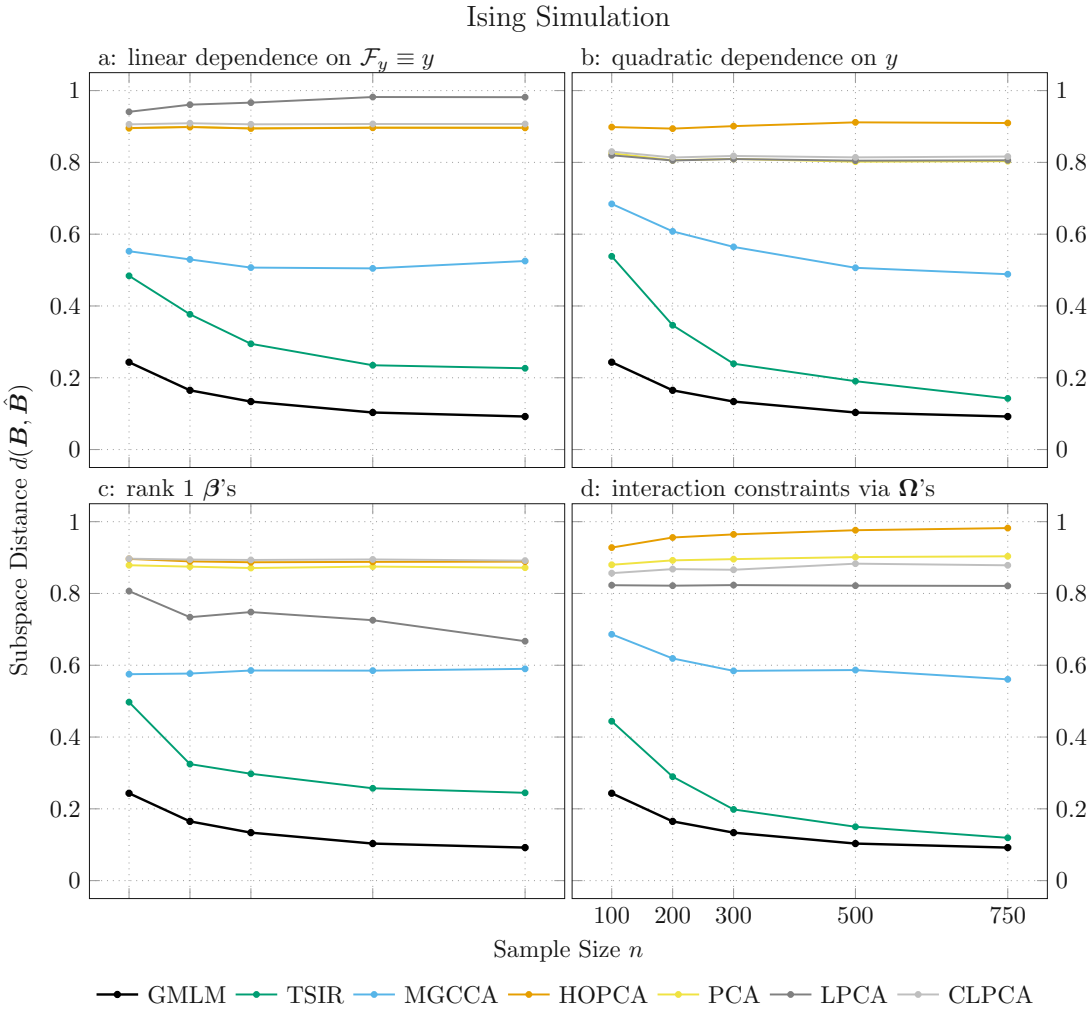

Figure 5.6: Visualization of the simulation results for Ising GMLM. Sample size on the x-axis and the mean of subspace distance $d(\boldsymbol{B}, \widehat{\boldsymbol{B}})$ over 100 replications on the y-axis. Described in Section 5.5.2.

The simulation results are visualized in Figure 5.6. Regardless of the simulation setting 2a-d), the comparative results are similar. We observe that PCA and HOPCA, both treating the response $\mathcal{X}$ as continuous, perform poorly. Not much better are LPCA and CLPCA. Similar to PCA and HOPCA they do not consider the relation to the response, but they are specifically created for binary predictors. Next we have MGCCA which is the first method considering the relation to the response y, clearly out-performing all the PCA variants. Even better is TSIR, regardless of the treatment of the predictors $\mathcal{X}$ as continuous, achieving very god results. Finally, the Ising GMLM model is the best in all the simulations although TSIR gets very close in some settings.

5.6 Data Analysis

In this section we perform two applications of the GMLM model on real data. First example is the tensor normal model applied to EEG data. Next, we perform a prove of concept data analysis example for chess.

5.6.1 EEG

The EEG data[10] is a small study of 77 alcoholic and 45 control subjects. Each data point corresponding to a subject consists of a $p_1 \times p_2 = 256 \times 64$ matrix, with each row representing a time point and each column a channel. The measurements were obtained by exposing each individual to visual stimuli and measuring voltage values from 64 electrodes placed on the subjects' scalps sampled at 256 time points over 1 second (256 Hz). Different stimulus conditions were used, and for each condition, 120 trials were measured. We used only a single stimulus condition (S1), and for each subject, we took the average of all the trials under that condition. That is, we used $(\mathcal{X}_i, y_i)$, $i = 1, \dots, 122$, where $\mathcal{X}_i$ is a 256×64 matrix, with each entry representing the mean voltage value of subject i at a combination of a time point and a channel, averaged over all trials under the S1 stimulus condition, and Y is a binary outcome variable with $Y_i = 1$ for an alcoholic and $Y_i = 0$ for a control subject.

For a comparison we reproduced the leave-one-out cross-validation EEG data analysis [PKB21, Sec. 7] for the classification task. In this data set, $p = p_1 p_2 = 16384$ is much larger than $n = 122$. To deal with this issue, [PKB21] used two approaches. In the first, pre-screening via $(2D)^2$PCA [ZZ05] reduced the dimensions to $(p_1, p_2) = (3, 4)$, $(15, 15)$ and $(20, 30)$. In the second, simultaneous dimension reductions and variable selection was carried out using the fast POI-C algorithm of [JAJ19] (due to high computational high burden, only a 10-fold cross-validation was performed for fast POI-C).

In contrast to [PKB21], our GMLM model can be applied directly to the raw data of dimension $(256, 64)$ without pre-screening or variable selection. This was not possible for KPIR as the time axis alone was in the large p small n regime with the $p_1 = 256 > n = 122$ leading to a singular time axis covariance. The same issue is present in the GMLM model, but the regularization trick used for numerical stability, as described in Section 5.2.1, resolves this without any change to the estimation procedure. In general, the sample size does not need to be large for maximum likelihood estimation in the tensor normal model. In matrix normal models in particular, [DKH20] proved that very small sample sizes, as little as 3,[11] are sufficient to obtain unique MLEs for Kronecker covariance structures.

We use leave-one-out cross-validation to obtain unbiased AUC estimates. Then, we compare the GMLM model to the best performing methods from [PKB21], namely KPIR (ls) and LSIR from [PFB12] for $(p_1, p_2) = (3, 4)$, $(15, 15)$ and $(20, 30)$.

In Table 5.2 we provide the AUC and its standard deviation. For all applied pre-screening dimensions, KPIR (ls) has an AUC of 78%. LSIR performs better at the price of some instability; it peaked at 85% at $(3, 4)$, then dropped down to 81% at $(15, 15)$ and then increased to 83%. In contract, our GMLM method peaked at $(3, 4)$ with 85% and stayed

[10] http://kdd.ics.uci.edu/databases/eeg/eeg.data.html

[11] The required minimum sample size depends on a non-trivial algebraic relations between the mode dimensions, while the magnitude of the dimensions has no specific role.

stable at 84%, even when no pre-processing was applied. In contrast, fast POI-C that carries out simultaneous feature extraction and feature selection resulted in an AUC of 63%, clearly outperformed by all other methods.

	AUC (St. Dev.)			
$(p_1, p_2) =$	$(3, 4)$	$(15, 15)$	$(20, 30)$	$(256, 64)$
FastPOI-C				0.63*(0.22)
$(2D)^2$PCR	0.83 (0.04)	0.50 (0.05)	0.53 (0.05)	
LSIR	0.85 (0.04)	0.81 (0.04)	0.83 (0.04)	
KPIR(ls)	0.78 (0.04)	0.78 (0.04)	0.78 (0.04)	
KPIR(mle)	0.75 (0.05)	0.78 (0.04)	0.77 (0.04)	
KPFC1	0.78 (0.04)	0.78 (0.04)	0.78 (0.04)	
KPFC2	0.78 (0.04)	0.78 (0.04)	0.78 (0.04)	
GMLM	0.85 (0.04)	0.84 (0.04)	0.84 (0.04)	0.84 (0.04)

*Mean AUC based on 10-fold cross-validation.

Table 5.2: Mean AUC values and their standard deviation based on leave-one-out cross-validation for the EEG imaging data (77 alcoholic and 45 control subjects)

5.6.2 Chess

The data set is provided by the *lichess.org open database*[12]. We downloaded November of 2023 consisting of more than 92 million games. We removed all games without position evaluations. The evaluations, also denoted scores, are from Stockfish[13], a free and strong chess engine. The scores take the role of the response Y and correspond to a winning probability from whites point of few. Positive scores are good for white and negative scores indicate an advantage for black. We ignore all highly unbalanced positions, which we set to be positions with absolute score above 5. We also remove all positions with a mate score (one side can force check mate). Furthermore, we only consider positions after 10 half-moves to avoid oversampling the beginning of the most common openings including the start position which is in every game. Finally, we only consider positions with white to move. This leads to a final data set of roughly 64 million positions, including duplicates.

A chess position is encoded as a set of 12 binary matrices $\mathcal{X}_{\text{piece}}$ of dimensions 8×8. Every binary matrix encodes the positioning of a particular piece by containing a 1 if the piece is present at the corresponding board position. The 12 pieces derive from the 6 types of pieces, namely pawns (♙), knights (♘), bishops (♗), queens (♕) and kings (♔) of two colors, black and white. See Figure 5.7 for a visualization.

We assume that $\mathcal{X}_{\text{piece}} \mid Y = y$ follows an Ising GMLM model Section 5.2.2 with different conditional piece predictors being independent. The independence assumption is for the sake of simplicity even though this is clearly not the case in the underlying true distribution. By

[12] T. Duplessis. Lichess.org open database, 2013. visited on December 8, 2023

[13] The Stockfish developers (see AUTHORS file). Stockfish, since 2008. Stockfish is a free and strong UCI chess engine. URL: https://stockfishchess.org

this simplifying assumption we get a mixture model with the log-likelihood

$$l_n(\boldsymbol{\theta}) = \frac{1}{12} \sum_{\text{piece}} l_n(\boldsymbol{\theta}_{\text{piece}})$$

where $l_n(\boldsymbol{\theta}_{\text{piece}})$ is the Ising GMLM log-likelihood as in Section 5.2.2 for $\mathcal{X}_{\text{piece}} \mid Y = y$. For every component the same relation to the scores y is modeled via a 2×2 dimensional matrix valued function $\mathcal{F}_y$ consisting of the monomials $1, y, y^2$, specifically $(\mathcal{F}_y)_{ij} = y^{i+j-2}$.

By the raw scale of the data, millions of observations, it is computationally infeasible to compute the gradients on the entire data set. Simply using a computationally manageable subset is not an option. Due to the high dimension on binary data, which is 12 times a 8×8 for every observation giving a total dimension of 768. The main issue is that a manageable subset, say one million observations, still leads to a degenerate data set. In our simplified mixture model, the pawns are a specific issue as there are multiple millions of different combinations of the 8 pawns per color on the 6×8 sub grid the pawns can be positioned. This alone does not allow to take a reasonable sized subset for estimation. The solution is to switch from a classic gradient based optimization to a stochastic version. This means that every gradient update uses a new random subset of the entire data set. Therefore, we draw independent random samples form the data consisting of 64 million positions. The independence of samples derived from the independence of games, and every sample is drawn from a different game.

Validation: Given the non-linear nature of the reduction, due to the quadratic matrix valued function $\mathcal{F}_y$ of the score y, we use a *generalized additive model*[14] (GAM) to predict position scores from reduced positions. The reduced positions are 48 dimensional continuous values by combining the 12 mixture components from the 2×2 matrix valued reductions per piece. The per piece reduction is

$$\mathcal{R}(\mathcal{X}_{\text{piece}}) = \boldsymbol{\beta}_{1,\text{piece}}(\mathcal{X}_{\text{piece}} - \mathbb{E}\,\mathcal{X}_{\text{piece}})\boldsymbol{\beta}_{2,\text{piece}}^T$$

which gives the complete 48 dimensional vectorized reduction by stacking the piece wise reductions

$$\operatorname{vec}\mathcal{R}(\mathcal{X}) = (\operatorname{vec}\mathcal{R}(\mathcal{X}_{\text{white pawn}}), \ldots, \operatorname{vec}\mathcal{R}(\mathcal{X}_{\text{black king}})) = \boldsymbol{B}^T \operatorname{vec}(\mathcal{X} - \mathbb{E}\,\mathcal{X}).$$

The second line encodes all the piece wise reductions in a block diagonal full reduction matrix $\boldsymbol{B}$ of dimension 768×48 which is applied to the vectorized 3D tensor $\mathcal{X}$ combining all the piece components $\mathcal{X}_{\text{piece}}$ into a single tensor of dimension $8 \times 8 \times 12$. This is a reduction to 6.25% of the original dimension. The R^2 statistic of the GAM fitted on 10^5 new reduced samples is $R^2_{\text{gam}} \approx 46\%$. A linear model on the reduced data achieves $R^2_{\text{lm}} \approx 26\%$ which clearly shows the non-linear relation. On the other hand, the static evaluation of the *Schach Hörnchen*[15] engine, given the full position (*not* reduced), achieves an $R^2_{\text{hce}} \approx 52\%$. The 42% are reasonably well compared to 51% of the engine static evaluation which gets the original position and uses chess specific expert knowledge. Features the static evaluation

[14] using the function `gam()` from the `R` package `mgcv`.
[15] My own UCI chess engine.

includes, which are expected to be learned by the GMLM mixture model, are; *material* (piece values) and *piece square tables* (PSQT, preferred piece type positions). In addition, the static evaluation includes chess specific features like *king safety*, *pawn structure* or *rooks on open files*. This lets us conclude that the reduction captures most of the relevant features possible, given the oversimplified modeling we performed.

Interpretation: For a compact interpretation of the estimated reduction we construct PSQTs. To do so we use the linear model from the validation section. Then, we rewrite the combined linear reduction and linear model in terms of PSQTs. Let $\boldsymbol{B}$ be the 768×48 full vectorized linear reduction. This is the block diagonal matrix with the 64×4 dimensional per piece reductions $\boldsymbol{B}_{\text{piece}} = \boldsymbol{\beta}_2^{\text{piece}} \otimes \boldsymbol{\beta}_1^{\text{piece}}$. Then, the linear model with coefficients $\boldsymbol{b}$ and intercept a on the reduced data is given by

$$y = a + \boldsymbol{b}^T \boldsymbol{B}^T \operatorname{vec}(\mathcal{X} - \mathbb{E}\,\mathcal{X}) + \epsilon \tag{5.30}$$

with an unknown mean zero error term ϵ and treating the binary tensor $\mathcal{X}$ as continuous. Decomposing the linear model coefficients into blocks of 4 gives per piece coefficients $\boldsymbol{b}_{\text{piece}}$ which combine with the diagonal blocks $\boldsymbol{B}_{\text{piece}}$ of $\boldsymbol{B}$ only. Rewriting (5.30) gives

$$\begin{aligned} y &= a + \sum_{\text{piece}} (\boldsymbol{B}_{\text{piece}} \boldsymbol{b}_{\text{piece}})^T \operatorname{vec}(\mathcal{X}_{\text{piece}} - \mathbb{E}\,\mathcal{X}_{\text{piece}}) + \epsilon \\ &= \tilde{a} + \sum_{\text{piece}} \langle \boldsymbol{B}_{\text{piece}} \boldsymbol{b}_{\text{piece}}, \operatorname{vec}(\mathcal{X}_{\text{piece}}) \rangle + \epsilon \end{aligned}$$

with a new intercept term $\tilde{a}$, which is of no interest to us. Finally, we enforce a color symmetry, using known mechanism from chess engines. Specifically, mirroring the position changes the sign of the score y. Here, mirroring reverses the rank (row) order, this is the image in a mirror behind a chess board. Let for every $\boldsymbol{C}_{\text{piece}}$ be a 8×8 matrix with elements $(\boldsymbol{C}_{\text{piece}})_{ij} = (\boldsymbol{B}_{\text{piece}} \boldsymbol{b}_{\text{piece}})_{i+8(j-1)}$. And denote with $\boldsymbol{M}(\boldsymbol{A})$ the matrix mirror operation which reverses the row order of a matrix. Using this new notation allows to enforcing this symmetry leading to the new approximate linear relation

$$\begin{aligned} y &= \tilde{a} + \sum_{\text{piece}} \langle \boldsymbol{C}_{\text{piece}}, \mathcal{X}_{\text{piece}} \rangle + \epsilon \\ &\approx \tilde{a} + \sum_{\text{piece type}} \frac{1}{2} \langle \boldsymbol{C}_{\text{white piece}} - \boldsymbol{M}(\boldsymbol{C}_{\text{black piece}}), \mathcal{X}_{\text{white piece}} - \boldsymbol{M}(\mathcal{X}_{\text{white piece}}) \rangle + \epsilon \end{aligned}$$

If for every piece type (6 types, *not* distinguishing between color) holds $\boldsymbol{C}_{\text{white piece}} = -\boldsymbol{M}(\boldsymbol{C}_{\text{black piece}})$, then we have equality. In our case this is valid given that the estimates $\widehat{\boldsymbol{C}}_{\text{piece}}$ fulfill this property with a small error. The 6 matrices $(\boldsymbol{C}_{\text{white piece}} - \boldsymbol{M}(\boldsymbol{C}_{\text{black piece}}))/2$ are called *piece square tables* (PSQT) which are visualized in Figure 5.8. The interpretation of those tables is straight forward. A high positive values (blue) means that it is usually good to have a piece of the corresponding type on that square while a high negative value (red) means the opposite. It needs to be considered that the PSQTs are for quiet positions only, that means all pieces are save in the sense that there is no legal capturing moves nore is the king in check.

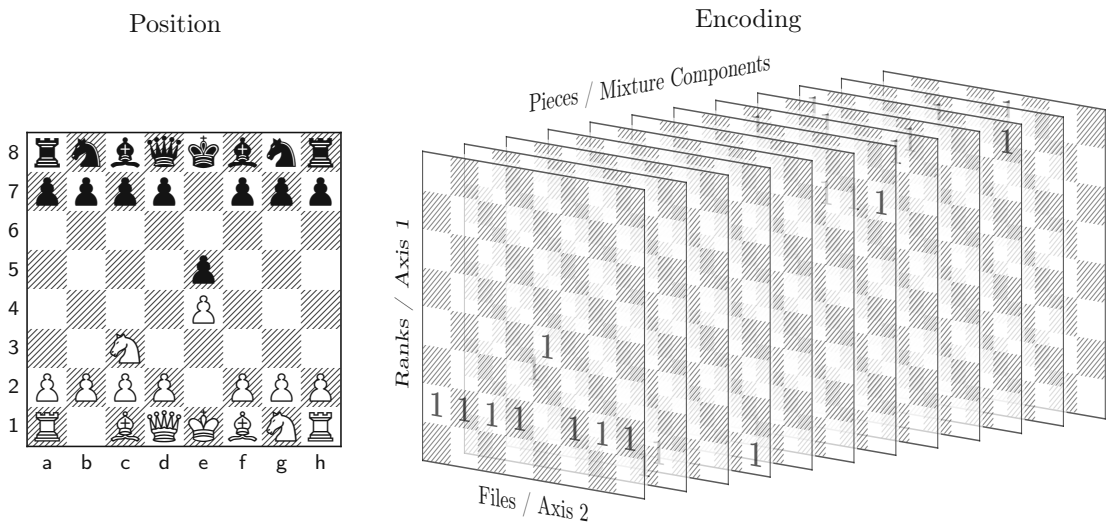

Figure 5.7: The chess start position and its 3D binary tensor representation, empty entries are 0.

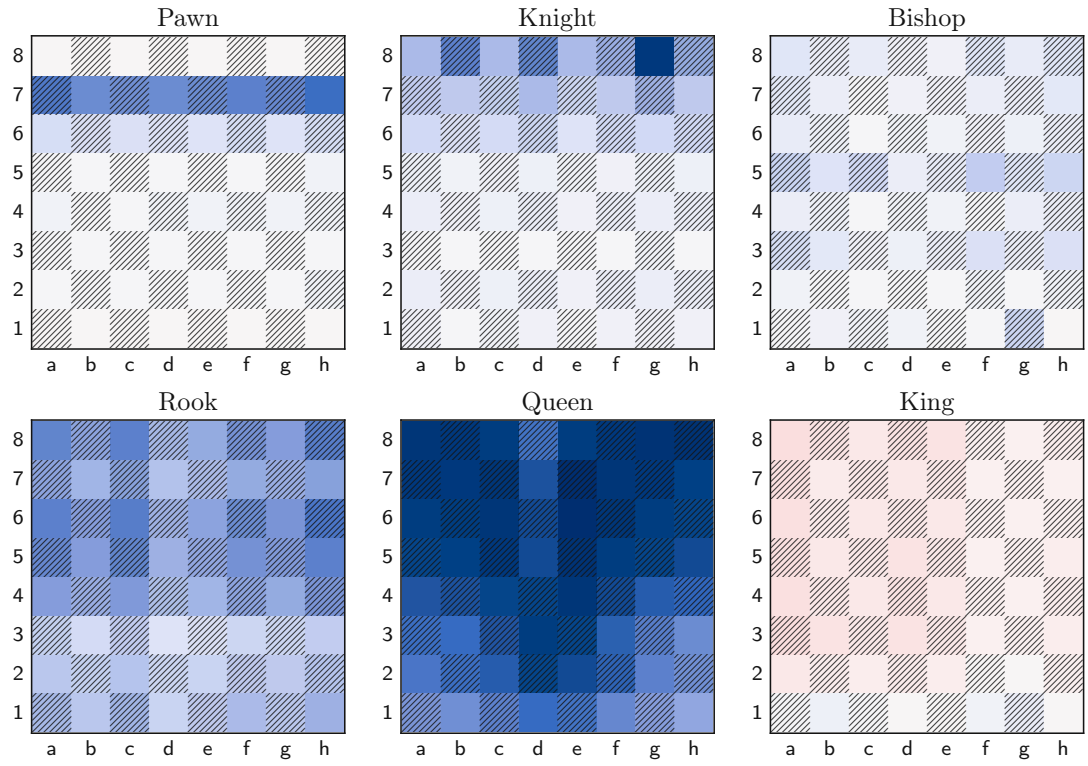

Figure 5.8: Extracted PSQTs (piece square tables) from the chess example GMLM reduction.

The first visual effect in Figure 5.8 is the dark blue PSQT of the Queen followed by a not so dark Rook PSQT. This indicated that the Queen, followed by the Rook, are the most value pieces (after the king, but a king piece value makes no sense). The next two are the Knight and Bishop which have higher value than the Pawns, ignoring the 6th and 7th rank as this makes the pawns a potential queen. This is the classic piece value order known in chess.

Next, goint one by one through the PSQTs, a few words about the prefered positions for every piece type. The pawn positions are specifically good on the 6th and especially on the 7th rank as this threatens a promotion to a Queen (or Knight, Bishop, Rook). The Knight PSQT is a bit surprising, the most likely explanation for the knight being good in the enemy territory is that it got there by capturing an enemy piece for free. A common occurency in low rated games which is a big chunk of the training data, ranging over all levels. The Bishops sem to have no specific prefered placement, only a slight higher overall value than pawns, excluding pawns iminent of a promotion. Continuing with the rooks, we see that the rook is a good attacking piece, indicated by a save rook infiltration. The Queen is powerfull almost everywhere, only the outer back rank squares (lower left and right) tend to reduce her value. This is rooted in the queen being there is a sign for being pushed by enemy pieces. Leading to a lot of squares being controled by the enemy hindering one own movement. Finally, the king, given the goal of the game is to checkmate the king, a save position for the king is very valuable. This is seen by the back rank (rank 1) being the only non-penalized squares. Furthermore, the most save squares are the castling target squares ($g1$ and $c1$) as well as the $b1$ square. Shifting the king over to $b1$ is quite common protecting the $a2$ pawn providing a complete protected pawn shield infront of the king.

6 Summary and Future Directions

In Chapter 3, we presented the *Neural Network Sufficient Dimension Reduction* (NNSDR) method and its sibling method *Neural Network Outer Product of Gradients* (NNOPG) [KFB22]. These methods target the mean subspace $\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$. We also presented a simplified yet comprehensive reference implementation in `R`, showing the step-by-step implementation. We provided a data application highlighting its effectiveness with large datasets in Section 3.5.2, a scenario where competing computationally expensive forward regression-based SDR methods, MAVE and CVE, struggle or even fail. The need for such methods is increasingly relevant in modern times, where massive datasets are commonplace, requiring SDR methods capable of processing such huge datasets without the need for supercomputing resources.

While theoretical guarantees are not provide, because of yet unresolved theoretical challenges inherent to neural networks, justification based on the *Universal Approximator Theorem* [Hor91], along with additional simulations presented in [KFB22], points to that establishing theoretical consistency is possible.

Another direction for exploration involves bi- or multi-linear reductions, extending both NNOPG using an approach based on the VLP decomposition [VP93], as seen in KPIR and KPFC from Chapter 4, or employing a chain rule-based outer product of gradients variant based on a Kronecker constraint to directly estimate mode-wise reductions. For NNSDR, the immediate next step in the context of tensor-valued data is to replace the first linear reduction layer with a multi-linear reduction layer, in analogy to GMLM from Chapter 5.

Shifting focus to matrix-valued predictors in Chapter 4, we introduced *Kronecker Parametric Inverse Regression* (KPIR) and *Kronecker Principal Fitted Components* (KPFC). These methods target the estimation of the first moment SDR subspace $\mathcal{S}_{\text{FMSDR}}$, a subset of the central subspace $\mathcal{S}_{Y|\boldsymbol{X}}$. They leverage additional information inherent to the matrix structure of the data, thereby enhancing estimation accuracy while retaining structural information through bi-linear reduction for which theoretical guarantees are provided. We also provided implementations in base `R`, displaying all the inner workings.

Generalizing these methods from matrix- to tensor-valued predictors is the immediate next step. Alternatively, another perspective is to explore the least squares versions of KPIR, which has weak assumptions, to avoid the need for estimating the full linear reduction, and subsequently applying VLP decomposition. Instead, direct estimation of the mode-wise reduction components could be investigated. It is also worth noting the previously mentioned integration with neural networks.

In Chapter 5, we introduced the novel *Generalized Multi-Linear Model* (GMLM) SDR method. It targets the central subspace $\mathcal{S}_{Y|\mathcal{X}}$ for tensor-valued $\mathcal{X}$ under the assumption that the inverse conditional distribution $\mathcal{X} \mid Y$ is a member of the quadratic exponential family. Moreover, it is assumed that the second moment is Kronecker separable. Under the GMLM, a multi-linear sufficient reduction is provided in Theorem 5. However, maximum likelihood

estimation does not apply to the Kronecker components in the multi-linear reduction due to lack of identifiability of Kronecker product components. This led to the development of the parameter space as an embedded Kronecker product manifold in Section 5.3.2.

Using manifold theory, asymptotic consistency and normality for the MLE as a member of an embedded parameter manifold is proven in Section 5.4, providing theoretical guarantees for the proposed method. A fast and memory-efficient algorithm is developed for maximum likelihood estimation under the tensor normal model in Section 5.2.1, including a simplified implementation in `R`. A less efficient but generally applicable algorithm is used for the Ising model in Section 5.2.2.

From provided simulations in Sections 5.5.1 and 5.5.2, we observe state-of-the-art performance where the modeling versatility of the GMLM method allows the incorporation of additional domain knowledge via modeling of the parameter space using domain-specific matrix manifolds as discussed in Section 5.3.1 as the "building blocks" of the parameter Kronecker manifold. We applied GMLM to EEG data in Section 5.6.1, showing excellent results even without pre-screening.

The excellent performance without pre-screening demonstrates the applicability of GMLM to tensor-valued $\mathcal{X}$ of dimension $p_1 \times \ldots \times p_r$ with sample size $n < p_k$. We mentioned that the number of needed observations can be very small, but no specifics were provided. Future work could investigate the minimum number of observations, which in its full generality depends on the problem size as well as on the dimension of the parameter manifold, and probably even its structure.

We finished with a proof-of-concept data example in Section 5.6.2 using a mixture of 64-bit Ising models covered by GMLM, showcasing its versatility. The implementation for the Ising model with dimensions of $p > 20$ uses Monte Carlo-based methods to be computationally feasible. Future directions could investigate pseudo-likelihood-based algorithms, potentially allowing much bigger problems. In general, interesting future directions could weaken the assumptions, specifically to allow general members of the exponential family, removing the restriction to the quadratic exponential family. The theoretical part is straightforward and basically provided by the general formulation of the asymptotic theory. The major challenge therein is in its practical applicability by providing algorithmic approaches capable of computing the MLE.

Another direction is to assume that the vectorized reduction matrix is a small sum of Kronecker products $\boldsymbol{B} = \sum_{j=1}^{s} \bigotimes_{k=r}^{1} \boldsymbol{\beta}_{s,k}$ for small s. This effectively removes an implicit rank-1 constraint, which arises from the relationship between the Kronecker product and the outer product.

Notation Index

$\bigotimes_{k=1}^{r} \boldsymbol{A}_k$ iterated Kronecker product, 16
$\bigcirc_{k=1}^{r} \boldsymbol{A}_k$ iterated outer product, 16
$\mathcal{A} \times_{k \in S} \boldsymbol{B}_k$ multi-linear multiplication, 15
$\mathcal{A} \circ \mathcal{B}$ outer product, 16
$\mathcal{A} \equiv \mathcal{B}$ vectorized equality, 15
$\mathcal{A} \odot \mathcal{B}$ Hadamard product, 33
$\mathcal{A} \otimes \mathcal{B}$ Kronecker product, 16
$\mathcal{A} \times_k \boldsymbol{B}$ k-mode product, 15
$\mathcal{A}_{(k)}$ k-mode matricization, 17
$T_{\boldsymbol{x}}\mathfrak{A}$ tangent space at $\boldsymbol{x} \in \mathfrak{A}$ of $\mathfrak{A}$, 73
$\operatorname{span}(\boldsymbol{A})$ subspace spanned by $\boldsymbol{A}$, 2
$\boldsymbol{A}^T$ transpose, 2
$\boldsymbol{A}^{-1}$ inverse, 2
$\boldsymbol{A}^{\dagger}$ Moore-Penrose inverse, 2
$\boldsymbol{D}_p$ duplication matrix, 53
$\boldsymbol{I}_p$ identity matrix, 2
$\boldsymbol{K}_{p,q}$ commutation matrix, 16
$\boldsymbol{P}_{\boldsymbol{A}}$ Projection onto span of $\boldsymbol{A}$, 7
$\boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{\Sigma})$ Projection onto span of $\boldsymbol{A}$ w.r.t. inner product induced by $\boldsymbol{\Sigma}$, 7
$\boldsymbol{S}_{\boldsymbol{p},\boldsymbol{q}}$ generalized outer to Kronecker permutation, 16
$\boldsymbol{e}_j$ standard unit vector, 2
$\det(\boldsymbol{A})$ determinant, 2
$\operatorname{diag}(\boldsymbol{A})$ vector of diagonal elements of $\boldsymbol{A}$, 2
$\operatorname{diag}(\boldsymbol{x})$ diagonal matrix with diagonal $\boldsymbol{x}$, 2
$\langle \mathcal{A}, \mathcal{B} \rangle$ inner product, 15
$\operatorname{logit}(x)$ logit function, 3
$\mathbb{R}_*^{p \times q}$ noncompact Stiefel manifold, 74
$\mathbb{R}_{\text{rank}=r}^{p \times q}$ matrices of fixed rank r, 74
$\mathrm{GL}_p(\mathbb{R})$ general linear group, 74
$\mathrm{St}^{p \times q}$ Stiefel manifold, 7, 74
$\mathrm{Sym}^{p \times p}$ symmetric matrices, 74
$\mathrm{Sym}_{++}^{p \times p}$ symmetric positive definite (SPD) matrices, 75
$\nabla_{\mathcal{X}} \mathcal{F}$ gradient of $\mathcal{F}$ w.r.t. $\mathcal{X}$, 18
$\operatorname{softmax}(x)$ softmax activation function, 41
$\operatorname{ReLU}(\boldsymbol{x})$ Rectified Linear Unit, 26
$\operatorname{tr}(\boldsymbol{A})$ trace, 2
$\operatorname{vec}(\mathcal{A})$ vectorization, 2, 15
$\operatorname{vech}(\boldsymbol{A})$ half vectorization, 2

$\mathcal{S}_{Y|\boldsymbol{X}}$ Central Subspace, 5
$\mathcal{S}_{\text{FMSDR}}$ first moment SDR subspace, 5
$\mathcal{S}_{Y|\mathcal{X}^{\otimes}}$ Central Dimension-Folding Subspace, 20
$\mathcal{S}_{\mathbb{E}[Y|\boldsymbol{X}]}$ Central Mean Subspace, 6

References

[AC09] K. P. Adragni and R. D. Cook. Sufficient dimension reduction and prediction in regression. *Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.*, 367(1906):4385–4405, 2009. DOI: 10.1098/rsta.2009.0110.

[AM05] K. M. Abadir and J. R. Magnus. *Matrix Algebra*. Econometric Exercises. Cambridge University Press, 2005. DOI: 10.1017/CBO9780511810800.

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008, pages xvi+224. DOI: 10.1515/9781400830244. Full Online Text https://press.princeton.edu/absil.

[Arn81] S. F. Arnold. *The theory of linear models and multivariate analysis*. eng. Wiley series in probability and mathematical statistics : Probability and mathematical statistics. Wiley, New York, NY [u.a.], 1981.

[AT20] J. Allaire and Y. Tang. *tensorflow: R Interface to 'TensorFlow'*. R package version 2.2.0. 2020.

[BC01] E. Bura and R. D. Cook. Estimating the structural dimension of regressions via parametric inverse regression. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 63(2):393–410, 2001.

[BDF+18] E. Bura, S. Duarte, L. Forzani, E. Smucler, and M. Sued. Asymptotic theory for maximum likelihood estimates in reduced-rank multivariate generalized linear models. *Statistics*, 52(5):1005–1024, 2018. DOI: 10.1080/02331888.2018.1467420.

[BDF16] E. Bura, S. Duarte, and L. Forzani. Sufficient reductions in regressions with exponential family inverse predictors. *J. Amer. Statist. Assoc.*, 111(515):1313–1329, 2016. DOI: 10.1080/01621459.2015.1093944.

[Bes74] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):192–225, 1974. DOI: 10.1111/j.2517-6161.1974.tb00999.x.

[BF15] E. Bura and L. Forzani. Sufficient Reductions in Regressions With Elliptically Contoured Inverse Predictors. *J. Amer. Statist. Assoc.*, 110(509):420–434, 2015. DOI: 10.1080/01621459.2014.914440.

[BFG+22] E. Bura, L. Forzani, R. García Arancibia, P. Llop, and D. Tomassi. Sufficient reductions in regression with mixed predictors. *J. Mach. Learn. Res.*, 23(102):1–47, 2022.

[Bot98] L. Bottou. Online algorithms and stochastic approximations. In D. Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998. http://leon.bottou.org/papers/bottou-98x revised, Oct 2012.

[BP07] P. J. Basser and S. Pajevic. Spectral decomposition of a 4th-order covariance tensor: applications to diffusion tensor mri. *Signal Processing*, 87(2):220–236, 2007. DOI: 10.1016/j.sigpro.2006.02.050. Tensor Signal Processing.

[Bur10] C. J. C. Burges. Dimension reduction: a guided tour. *Foundations and Trends® in Machine Learning*, 2(4):275–365, 2010. DOI: 10.1561/2200000002.

[Bur95] D. S. Burdick. An introduction to tensor products with applications to multiway data analysis. *Chemometrics and Intelligent Laboratory Systems*, 28(2):229–237, 1995. DOI: 10.1016/0169-7439(95)80060-M.

[CC02] F. Chiaromonte and R. D. Cook. Sufficient dimension reduction and graphics in regression. *Ann. Inst. Statist. Math.*, 54(4):768–795, 2002. DOI: 10.1023/A:1022411301790.

[CF08] R. D. Cook and L. Forzani. Principal fitted components for dimension reduction in regression. *Statistical Science*, 23(4):485–501, 2008.

[CF09] R. D. Cook and L. Forzani. Likelihood-based sufficient dimension reduction. *Journal of the American Statistical Association*, 104(485):197–208, 2009. DOI: 10.1198/jasa.2009.0106.

[CKG22] A. Chakraborty, M. Katzfuss, and J. Guinness. Ordered conditional approximation of potts models. *Spatial Statistics*, 52:100708, 2022. DOI: 10.1016/j.spasta.2022.100708.

[CKT21] Y.-L. Chen, M. Kolar, and R. S. Tsay. Tensor canonical correlation analysis with convergence and statistical guarantees. *Journal of Computational and Graphical Statistics*, 30(3):728–744, 2021. DOI: 10.1080/10618600.2020.1856118.

[CL02] R. Cook and B. Li. Dimension reduction for conditional mean in regression. *The Annals of Statistics*, 30(2):455–474, 2002. DOI: 10.1214/aos/1021379861.

[CL95] R. J. Carroll and K.-C. Li. Binary regressors in dimension reduction models: a new look at treatment comparisons. *Statist. Sinica*, 5(2):667–688, 1995.

[CLW+14] J. Cheng, E. Levina, P. Wang, and J. Zhu. A sparse ising model with covariates. *Biometrics*, 70(4):943–953, 2014. DOI: 10.1111/biom.12202.

[CN94] R. D. Cook and C. J. Nachtsheim. Reweighting to achieve elliptically contoured covariates in regression. *Journal of the American Statistical Association*, 89(426):592–599, 1994.

[Com09] P. Comon. Tensors versus matrices usefulness and unexpected properties. In *2009 IEEE/SP 15th Workshop on Statistical Signal Processing*, pages 781–788, 2009. DOI: 10.1109/SSP.2009.5278471.

[Con97] J. B. Conway. *A Course in Functional Analysis*, number 96 in Graduate Texts in Mathematics. New York, 2nd ed edition, 1997.

[Coo07] R. D. Cook. Fisher Lecture: Dimension Reduction in Regression. *Statistical Science*, 22(1):1–26, 2007. DOI: 10.1214/088342306000000682.

[Coo18] R. D. Cook. Principal components, sufficient dimension reduction, and envelopes. *Annual Review of Statistics and Its Application*, 5(1):533–559, 2018. DOI: 10.1146/annurev-statistics-031017-100257.

[Coo98] D. R. Cook. *Regression Graphics: Ideas for studying regressions through graphics*. Wiley, New York, 1998.

[CW91] R. D. Cook and S. Weisberg. Sliced inverse regression for dimension reduction: comment. *Journal of the American Statistical Association*, 86(414):328–332, 1991.

[Daw81] A. P. Dawid. Some matrix-variate distribution theory: notational considerations and a bayesian application. *Biometrika*, 68(1):265–274, 1981.

[DC07] L. De Lathauwer and J. Castaing. Tensor-based techniques for the blind separation of ds-cdma signals. *Signal Processing*, 87(2):322–336, 2007. DOI: 10.1016/j.sigpro.2005.12.015. Tensor Signal Processing.

[DC15] S. Ding and R. D. Cook. Tensor sliced inverse regression. *Journal of Multivariate Analysis*, 133:216–231, 2015. DOI: 10.1016/j.jmva.2014.08.015.

[DDW13] B. Dai, S. Ding, and G. Wahba. Multivariate Bernoulli distribution. *Bernoulli*, 19(4):1465–1483, 2013. DOI: 10.3150/12-BEJSP10.

[dFM07] A. L. de Almeida, G. Favier, and J. C. M. Mota. Parafac-based unified tensor modeling for wireless communication systems with application to blind multiuser equalization. *Signal Processing*, 87(2):337–351, 2007. DOI: https://doi.org/10.1016/j.sigpro.2005.12.014. Tensor Signal Processing.

[DKH20] M. Drton, S. Kuriki, and P. D. Hoff. Existence and uniqueness of the kronecker covariance mle. *The Annals of Statistics*, 2020.

[DKZ09] I. L. Dryden, A. Koloydenko, and D. Zhou. Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging. *The Annals of Applied Statistics*, 3(3):1102–1123, 2009. DOI: 10.1214/09-AOAS249.

[Dup13] T. Duplessis. Lichess.org open database, 2013. visited on December 8, 2023.

[Dut99] P. Dutilleul. The mle algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2):105–123, 1999. DOI: 10.1080/00949659908811970.

[Fan93] J. Fan. Local linear regression smoothers and their minimax efficiencies. *Annals of Statistics*, 21:196–216, 1993.

[FG92] J. Fan and I. Gijbels. Variable bandwidth and local linear regression smoothers. *The Annals of Statistics*, 20(4):2008–2036, 1992.

[Fis22] R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222:309–368, 1922.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[GGK+21] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley. Sufficient dimension reduction for high-dimensional regression and low-dimensional embedding: tutorial and survey, 2021. DOI: 10.48550/arXiv.2110.09620. arXiv: 2110.09620 [stat.ME].

[GGL+24] F. Girka, A. Gloaguen, L. Le Brusquet, V. Zujovic, and A. Tenenhaus. Tensor generalized canonical correlation analysis. *Information Fusion*, 102, 2024. DOI: 10.1016/j.inffus.2023.102045.

[GH14] K. H. Greenewald and A. O. Hero. Robust kronecker product pca for spatio-temporal covariance estimation. *IEEE Transactions on Signal Processing*, 63:6368–6378, 2014.

[GT03] A. Globerson and N. Tishby. Sufficient dimensionality reduction. *Journal of Machine Learning Research*, 3, 2003.

[Gur97] K. Gurney. *An Introduction to Neural Networks*. Taylor & Francis, Inc., USA, 1997.

[Har97] D. A. Harville. *Matrix Algebra From a Statistician's Perspective*. Springer-Verlag, New York, 1st edition, 1997. Chapter 15.

[Hin12] G. E. Hinton. Neural Networks for Machine Learning, 2012. Coursera Lecture 6 - Online; accessed Jan 18, 2024.

[HL13] C. J. Hillar and L.-H. Lim. Most tensor problems are np-hard. *J. ACM*, 60(6), 2013. DOI: 10.1145/2512329.

[Hof11] P. D. Hoff. Separable covariance arrays via the Tucker product, with applications to multivariate relational data. *Bayesian Analysis*, 6(2):179–196, 2011. DOI: 10.1214/11-BA606.

[Hof15] P. D. Hoff. Multilinear tensor regression for longitudinal relational data. *The Annals of Applied Statistics*, 9(3):1169–1193, 2015. DOI: 10.1214/15-AOAS839.

[Hor91] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. https://doi.org/10.1016/0893-6080(91)90009-T.

[HT90] T. J. Hastie and R. J. Tibshirani. *Generalized additive models*, volume 43 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, Ltd., London, 1990, pages xvi+335.

[Isi25] E. Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258, 1925. DOI: 10.1007/BF02980577.

[JAJ19] S. Jung, J. Ahn, and Y. Jeon. Penalized orthogonal iteration for sparse estimation of generalized eigenvalue problem. *Journal of Computational and Graphical Statistics*, 28(3):710–721, 2019. DOI: 10.1080/10618600.2019.1568014.

[JHK21] M. Jenny, M. Haselmayer, and D. Kapla. Measuring incivility in parliamentary debates : validating a sentiment analysis procedure with calls to order in the austrian parliament. In A. S. Walter, editor, *Political Incivility in the Parliamentary, Electoral and Media Arena : Crossing Boundaries*, Routledge studies on political parties and party systems, pages 1–11. Routledge, London, 2021.

[JKB97] N. L. Johnson, S. Kotz, and N. Balakrishnan. *Discrete Multivariate Distributions*. Wiley Series in Probability and Statistics: Applied Probability and Statistics. John Wiley & Sons, Inc., New York, 1997, pages xxii+299. A Wiley-Interscience Publication.

[JWH+21] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning—with applications in R*. Springer Texts in Statistics. Springer, New York, second edition, 2021, pages xv+607. DOI: 10.1007/978-1-0716-1418-1.

[Kal21] M. Kaltenbäck. *Aufbau Analysis*. Berliner Studienreihe zur Mathematik. Heldermann Verlag, 27th edition, 2021.

[KB09] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009. DOI: 10.1137/07070111X.

[KFB22] D. Kapla, L. Fertl, and E. Bura. Fusing sufficient dimension reduction with neural networks. *Comput. Statist. Data Anal.*, 168:Paper No. 107390, 20, 2022. DOI: 10.1016/j.csda.2021.107390.

[Kol06] T. G. Kolda. Multilinear operators for higher-order decompositions. 2006. DOI: 10.2172/923081.

[KR05] E. Kofidis and P. A. Regalia. Tensor approximation and signal processing applications. In 2005.

[Kra91] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991. https://doi.org/10.1002/aic.690370209.

[KvR05] T. Kollo and D. von Rosen. *Advanced Multivariate Statistics with Matrices*. M. Hazewinkel, editor. Springer Dordrecht, 2005. DOI: 10.1007/1-4020-3419-9.

[KW19] D. P. Kingma and M. Welling. An Introduction to Variational Autoencoders. arXiv:1906.02691 [cs.LG], 2019. http://arxiv.org/abs/1906.02691.

[Lau96] S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996. DOI: 10.1093/oso/9780198522195.001.0001.

[LBB+98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. DOI: 10.1109/5.726791.

[Lee12] J. M. Lee. *Introduction to Smooth Manifolds*. Springer New York, 2012. DOI: 10.1007/978-1-4419-9982-5.

[Lee18] J. M. Lee. *Introduction to Riemannian Manifolds*. Springer International Publishing, 2018. DOI: 10.1007/978-3-319-91755-9.

[Len20] W. Lenz. Beitrag zum verständnis der magnetischen erscheinungen in festen körpern. *European Physical Journal A*, 21:613–615, 1920.

[Li18] B. Li. *Sufficient dimension reduction*, volume 161 of *Monographs on Statistics and Applied Probability*. CRC Press, Boca Raton, FL, 2018, pages xxi+283. DOI: 10.1201/9781315119427. Methods and applications with R.

[Li91] K.-C. Li. Sliced Inverse Regression for Dimension Reduction. *J. Amer. Statist. Assoc.*, 86(414):316–327, 1991. DOI: 10.1080/01621459.1991.10475035.

[Lin19] Z. Lin. Riemannian geometry of symmetric positive definite matrices via cholesky decomposition. *SIAM Journal on Matrix Analysis and Applications*, 40(4):1353–1370, 2019. DOI: 10.1137/18M1221084.

[LKA10] B. Li, M. K. Kim, and N. Altman. On dimension folding of matrix- or array-valued statistical objects. *The Annals of Statistics*, 38(2):1094–1121, 2010. DOI: 10.1214/09-AOS737.

[LL16] W. Luo and B. Li. Combining eigenvalues and variation of eigenvectors for order determination. *Biometrika*, 103(4):875–887, 2016. DOI: 10.1093/biomet/asw051.

[LL20] A. J. Landgraf and Y. Lee. Dimensionality reduction for binary data through the projection of natural parameters. *Journal of Multivariate Analysis*, 180:104668, 2020. DOI: 10.1016/j.jmva.2020.104668.

[LR02] S. L. Lauritzen and T. S. Richardson. Chain graph models and their causal interpretations. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 64(3):321–361, 2002.

[LR92] S. Leurgans and R. T. Ross. Multilinear Models: Applications in Spectroscopy. *Statistical Science*, 7(3):289–310, 1992. DOI: 10.1214/ss/1177011225.

[LZ05] N. Lu and D. L. Zimmerman. The likelihood ratio test for a separable covariance matrix. *Statistics & Probability Letters*, 73(4):449–457, 2005. DOI: 10.1016/j.spl.2005.04.020.

[MAP+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[MD13] A. M. Manceur and P. Dutilleul. Maximum likelihood estimation for the tensor normal distribution: algorithm, minimum sample size, and empirical bias and dispersion. *Journal of Computational and Applied Mathematics*, 239:37–49, 2013. DOI: 10.1016/j.cam.2012.09.017.

[MG93] K. V. Mardia and C. R. Goodall. Spatial-temporal analysis of multivariate environmental monitoring data. In *Multivariate environmental statistics*. Volume 6, North-Holland Ser. Statist. Probab. Pages 347–386. North-Holland, Amsterdam, 1993.

[MN86] J. R. Magnus and H. Neudecker. Symmetry, 0-1 matrices and jacobians: a review. *Econometric Theory*, 2(2):157–190, 1986.

[MN99] J. R. Magnus and H. Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. Wiley Series in Probability and Statistics. John Wiley & Sons, Ltd., Chichester, 1999, pages xviii+395. Revised reprint of the 1988 original.

[MP43] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[MZ13] Y. Ma and L. Zhu. A review on dimension reduction. *Int. Stat. Rev.*, 81(1):134–150, 2013. DOI: 10.1111/j.1751-5823.2012.00182.x.

[Nis05] M. Niss. History of the Lenz-Ising Model 1920–1950: From Ferromagnetic to Cooperative Phenomena. *Arch. Hist. Exact Sci.*, 59(3):267–318, 2005. DOI: 10.1007/s00407-004-0088-3.

[NZB17] H. C. Nguyen, R. Zecchina, and J. Berg. Inverse statistical problems: from the inverse Ising problem to data science. *Advances in Physics*, 66(3):197–261, 2017. DOI: 10.1080/00018732.2017.1341604.

[OAvR13] M. Ohlson, M. R. Ahmad, and D. von Rosen. The multilinear normal distribution: introduction and some basic properties. *Journal of Multivariate Analysis*, 113:37–47, 2013. DOI: 10.1016/j.jmva.2011.05.015.

[PFB12] R. Pfeiffer, L. Forzani, and E. Bura. Sufficient dimension reduction for longitudinally measured predictors. *Statistics in medicine*, 31:2414–27, 2012. DOI: 10.1002/sim.4437.

[PKB21] R. Pfeiffer, D. Kapla, and E. Bura. Least squares and maximum likelihood estimation of sufficient reductions in regressions with matrix-valued predictors. *International Journal of Data Science and Analytics*, 11, 2021. DOI: 10.1007/s41060-020-00228-y.

[Ros58] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. DOI: 10.1037/h0042519.

[RW94] D. Ruppert and M. P. Wand. Multivariate locally weighted least squares regression. *The Annals of Statistics*, 22(3):1346–1370, 1994.

[SF08] The Stockfish developers (see AUTHORS file). Stockfish, since 2008. Stockfish is a free and strong UCI chess engine. URL: https://stockfishchess.org.

[SHK+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[UV20] A. Uschmajew and B. Vandereycken. *Geometric methods on low-rank matrix and tensor manifolds*. In *Handbook of Variational Methods for Nonlinear Geometric Data*. P. Grohs, M. Holler, and A. Weinmann, editors. Springer International Publishing, Cham, 2020, pages 261–313. DOI: 10.1007/978-3-030-31351-7_9.

[van98] A. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[VP93] C. F. Van Loan and N. Pitsianis. *Approximation with kronecker products*. In *Linear Algebra for Large Scale and Real-Time Applications*. M. S. Moonen, G. H. Golub, and B. L. R. De Moor, editors. Springer Netherlands, Dordrecht, 1993, pages 293–314. DOI: 10.1007/978-94-015-8196-7_17.

[Whi90] J. Whittaker. *Graphical models in applied multivariate statistics*. Wiley Series in Probability and Mathematical Statistics: Probability and Mathematical Statistics. John Wiley & Sons, Ltd., Chichester, 1990, pages xiv+448.

[WJ08] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008. DOI: 10.1561/2200000001.

[WSS+22] Y. Wang, Z. Sun, D. Song, and A. Hero. Kronecker-structured covariance models for multiway data. *Statistics Surveys*, 16(none):238–270, 2022. DOI: 10.1214/22-SS139.

[Xia07] Y. Xia. A constructive approach to the estimation of dimension reduction directions. *Ann. Statist.*, 35(6):2654–2690, 2007. DOI: 10.1214/009053607000000352.

[XTL+02] Y. Xia, H. Tong, W. K. Li, and L.-X. Zhu. An Adaptive Estimation of Dimension Reduction Space. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 64(3):363–410, 2002. DOI: 10.1111/1467-9868.03411.

[YLC08] X. Yin, B. Li, and R. D. Cook. Successive direction extraction for estimating the central subspace in a multiple-index regression. *J. Multivariate Anal.*, 99(8):1733–1757, 2008. DOI: 10.1016/j.jmva.2008.01.006.

[ZZ05] D. Zhang and Z.-H. Zhou. (2D)2PCA: two-directional two-dimensional PCA for efficient face representation and recognition. *Neurocomputing*, 69(1):224–231, 2005. DOI: 10.1016/j.neucom.2005.06.004. Neural Networks in Signal Processing.

Curriculum Vitae

Personal data

Name	**Daniel Kapla**
Date of birth	[redacted]
Birth place	Vienna
Nationality	Austria
Email	[redacted]
Homepage	https://kapla.at

Education

2019 - present	Doctoral Studies in Mathematics at TU Wien
2016 - 2019	Masters's Degree in Mathematics at TU Wien
2011 - 2016	Bachelor's Degree in Mathematics at TU Wien

Publications

M. Jenny, M. Haselmayer, and D. Kapla. Measuring incivility in parliamentary debates : validating a sentiment analysis procedure with calls to order in the austrian parliament. In A. S. Walter, editor, *Political Incivility in the Parliamentary, Electoral and Media Arena : Crossing Boundaries*, Routledge studies on political parties and party systems, pages 1–11. Routledge, London, 2021

R. Pfeiffer, D. Kapla, and E. Bura. Least squares and maximum likelihood estimation of sufficient reductions in regressions with matrix-valued predictors. *International Journal of Data Science and Analytics*, 11, 2021. DOI: 10.1007/s41060-020-00228-y

D. Kapla, L. Fertl, and E. Bura. Fusing sufficient dimension reduction with neural networks. *Comput. Statist. Data Anal.*, 168:Paper No. 107390, 20, 2022. DOI: 10.1016/j.csda.2021.107390