UCLouvain
Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

Extracting Information from Multiple Datasets by Matrix Factorization and Common Subspace Computation

Emilie Renard

Thesis submitted in partial fulfillment of the requirements for the degree of
Docteur en sciences de l'ingénieur

Dissertation committee:

Pierre-Antoine Absil (advisor)	Université catholique de Louvain
Vincent Blondel (advisor)	Université catholique de Louvain
Kyle Gallivan	Florida State University
Thibault Helleputte	DNAlytics
Raphaël Jungers (chairman)	Université catholique de Louvain
Michel Verleysen	Université catholique de Louvain
Andrew Teschendorff	University of Chinese Academy of Sciences, University College London

Abstract

With the constantly growing capability to measure and store data, dealing with different datasets representing similar phenomena is becoming increasingly common. An example in bioinformatics is gene expression data, where the same disease can be studied in different hospitals at different times by measuring the gene expression levels of patients. It is then desirable to merge the different measures obtained, in order to improve the robustness of the subsequent analysis.

In this thesis, we investigate different ways of extracting common information from multiple datasets with common features. We propose two different types of approach: removing the differences across datasets, and, keeping the common information.

To remove the differences across datasets, we use a matrix factorization approach: we develop a new spatiotemporal version of independent component analysis that we apply to all concatenated datasets, and use the resulting components to model the differences to be removed. The approach is validated on gene expression datasets, and compared to other existing methods in the wider context of a classification task.

In the second part of the thesis, we propose a minimax formulation to model the problem of finding a common subspace across a set of subspaces associated with the datasets. We develop multiple algorithms to solve this minimax problem, with some convergence guarantees, and compare their performances on synthetic and real data.

Remerciements

On ne le dira jamais assez: une thèse est un projet de longue haleine, avec beaucoup de hauts et de bas, et celle-ci n'existerait pas sans le soutien d'une multitude de personnes.

Un grand merci d'abord à mes promoteurs, Pierre-Antoine et Vincent, de m'avoir accueillie au bâtiment Euler et sans qui cette thèse n'existerait pas. Vincent, tes responsabilités ont assez vite limité nos interactions, mais je tenais à te remercier de m'avoir fait confiance en m'offrant la possibilité de travailler dans cette ambiance si accueillante de l'Euler. Pierre-Antoine, merci pour ta disponibilité, ta capacité à expliquer simplement des concepts très abstraits, tes relectures plus que détaillées, et ta patience. Patience qui a fini par payer car même si j'ai toujours été un peu l'outlier dans l'équipe de par mes sujets d'intérêt en recherche, j'ai fini par basculer volontairement du côté obscur de l'optimisation: les variétés. Et j'ai même apprécié l'expérience! Merci Michel de m'avoir convaincue de commencer une thèse, de m'avoir fait découvrir la recherche et le côté fascinant du machine learning sans oublier l'ambiance conviviale du Machine Learning Group et de l'ESANN. Thank you Andrew for your valuable advice on bioinformatics, thank you Kyle for all the discussions and suggestions on the minimax problem, and the more than detailed proofreading of this text. Discussing with you was always enjoyable and enriching. Merci Thibault d'avoir pris le temps de lire cette thèse, et pour tous les commentaires plus que détaillés qui ont participé à son amélioration. Merci enfin à Raphaël d'avoir accepté de présider le jury, avec une efficacité et une bonne humeur qui m'ont facilité cette étape (un peu) stressante qu'est la fin de thèse.

Je ne garderai probablement pas un aussi bon souvenir de cette thèse sans la très bonne ambiance de travail de l'Euler et de l'ICTEAM. Pouvoir travailler tous les jours dans un environnement presque familial comme l'Euler est une chance, et m'a permis de nombreuses rencontres enrichissantes. Merci à Adeline, Maguy, Romain, Matthew, Corentin, Pierre-Yves, Stéphanie, Allan, Estelle, Benjamin, Aaron, François, Adrien(s), Cécile, Charles, Guillaume(s), Nicolas, Loïc et tous les (nombreux) autres pour toutes les discussions scientifiques ou non, que cela soit au détour d'un couloir, à la sortie d'une réunion, autour d'un plat du jour au Galilée, de mots croisés, d'un morceau de tarte ou d'une tasse de thé/café. Les pauses à la cafet sont définitivement un plus de l'Euler! Merci au personnel administratif et technique, en particulier Marie-Christine, Pascale, Nathalie et Etienne, d'être toujours de bonne humeur quand on vient vous interrompre pour une question pratique, et d'avoir toujours trouvé une solution à chaque problème. L'Euler a beaucoup de chance d'avoir une équipe aussi dynamique et efficace (en plus d'être agréable!).

Merci aux (anciens et plus récents) membres du Machine Learning Group et assimilés, le pèlerinage annuel à Bruges n'aurait pas été pareil sans vous! Un merci particulier à Benoît et Gauthier, qui m'ont accueillie avec enthousiasme dans leur bureau durant ma première année de recherche. Ma consommation

de thé et mon intérêt pour la recherche en sont ressortis grandis. Thank you Tim for the nice discussions on what is the right dimension! Merci aussi à tous les représentants AsCII que j'ai côtoyés: les cougnous, ICTEAM days, projections de film, ateliers papier mâché/peinture et autres restent de très bons souvenirs, en grande partie grâce aux personnes qui s'y sont impliquées. En tant qu'assistante, l'enseignement a pris une partie non négligeable de mon temps ces dernières années, le plus souvent avec plaisir: merci à tous ceux avec qui j'ai eu la chance d'encadrer un cours et qui m'ont aidée à (j'espère) améliorer mes compétences pédagogiques.

Un merci tout particulier à Adeline et Estelle, deux collègues de bureau en or, toujours patientes et prêtes à remonter le moral quand c'est nécessaire. Merci d'avoir supporté mes plaintes avec le sourire (et du chocolat), et pour toutes les discussions interminables. Vos qualités scientifiques et humaines, que ce soit pour des conseils en stat, des (ré)explications sur les variétés, des relectures de mails, du maniement de tournevis en conférence ou du contrôle de trajectoire de gomme, m'ont été très précieuses.

Merci aux amis, plus ou moins récents, scientifiques ou pas, rencontrés lors des études, au travail, en secondaire, par la musique,... Merci plus particulièrement à Adeline, Andra, Aline, Maguy, Julien, François et Pierre-Antoine.

Merci à ma famille qui m'a toujours soutenue dans mes choix, particulièrement à mes parents qui m'ont appris la curiosité, à mes frères pour leurs sous-entendus que je n'ai (presque) jamais mal pris, et à mes grands-mères qui ont toujours été fières de leurs petits-enfants.

Merci enfin à tous ceux qui m'ont un jour encouragée dans ma thèse, que ce soit en s'intéressant (de manière plus ou moins approfondie) à mon travail ou en posant la question fatidique '*Et tu trouves?*'. Question à laquelle je peux enfin répondre '*Oui!*', tout du moins assez pour écrire cette thèse... J'ai cependant trouvé encore plus, à savoir beaucoup de rencontres enrichissantes et de discussions passionnantes!

Contents

Notations

Depending on the context, notations can vary slightly. Without further specifications, interpretations are as follows.

$\mathbb{R}_+$	set of null or positive real numbers
$\mathbb{R}_*^{p\times k}$	set of full rank matrices of size $p \times k$ with values in $\mathbb{R}$
$\mathcal{G}(k,p)$	Grassmann manifold: set of all k-dimensional subspaces in $\mathbb{R}^p$
$\mathrm{St}(k,p)$	Stiefel manifold: set of all orthonormal k-frames in $\mathbb{R}^p$
$GL(k)$	set of $k \times k$ invertible matrices
I_p	identity matrix of size p
$\mathbf{1}_p$	vector of all ones in $\mathbb{R}^p$
X (upper case)	matrix
X_{ij} or $X(i,j)$	(i,j)th element of matrix X
x (lower case)	vector
x_i or $X_{:i}$ or $X(:,i)$	ith column of matrix X
$\mathrm{cov}(X)$	covariance matrix associated to X
$\mathrm{var}(x)$	variance associated to x
$\mathrm{vec}(X)$	vectorization of X
$\mathrm{Tr}(X)$	trace of matrix X
$\mathrm{diag}(X)$	vector of the main diagonal elements of matrix X
$\mathrm{diag}(x)$	square diagonal matrix with vector x on the main diagonal
$\mathrm{span}(X)$	subspace generated by the columns of X
$\mathrm{conv}(X_1,\ldots,X_m)$	convex hull of $\{X_1,\ldots,X_m\}$
$\mathrm{aff}(X_1,\ldots,X_m)$	affine hull of $\{X_1,\ldots,X_m\}$
$\|\|X\|\|$	norm of X (without precision, norm 2)
$\langle X,Y\rangle$	scalar product between X and Y
p	number of features
n	number of samples
k	low-rank dimension
$\check{X}$	orthonormal basis of $\mathrm{span}(X)$
X^e	projection of X in the Euclidean tangent space
$X^{(t)}$, $X^{(t+1)}$	in an algorithm, value of X at current and next iterations
$d(X,Y)$	dissimilarity between X and Y
$U_{[a\ b]}$	uniform distribution on interval $[a\ b]$
SVD	Singular Value Decomposition
ICA	Independent Component Analysis
CCA	Canonical Correlation Analysis

Chapter 1

Introduction

We are today entering what many call the fourth industrial revolution, or the Industry 4.0. While the first industrial revolution is associated to the mechanization and steam power, the second one to electricity and mass production and the third one to computer, Industry 4.0 corresponds to the current trend of automation and data exchange in manufacturing. Closely linked to the internet-of-things and cloud computing, it relies heavily on data collection and analysis. The increasing number of sensors in our daily life, combined with the current facility to store data, generate more and more information everyday. This explosion of the amount of data available is in both the number of samples and the number of features, and leads to what is called today *Data Science*.

Data science is a general term used to describe a field that is not well-defined. It uses tools from many different fields like statistics, computer science, machine learning and mathematics to transform raw data, structured or not, into information useful for decision-making purposes. The rise of data science started in the 1960's, when statistics met computer science thanks to the development of the first desktop computers. This new grasp of data is described in "The Future of Data Analysis" by John W. Tukey : "For a long time I thought I was a statistician, interested in inferences from the particular to the general. But as I have watched mathematical statistics evolve, I have had cause to wonder and doubt... I have come to feel that my central interest is in data analysis... ". In the 1970's, the use of computers for statistics increased and the International Association for Statistics Computing was funded. The first workshop on data science was held in the 1980's. The beginning of marketing in the 1990's saw companies starting to collect and store all kinds of data about consumers. Overwhelmed by the quantity of data, companies did not always know how to use it efficiently but were convinced that it could be useful. The decreasing price of hard-discs also helped increase the collection of more and more data while software as a service began. The software is centrally hosted on a server and provided to the customer based on subscriptions – a model that eventually led to cloud services. In the 2000's, the first academic journals dedi-

cated to the field were officially launched. Since then, the term "data scientist" has become a buzzword and the number of data science jobs and conferences has exploded, with notably the huge success of deep learning since 2016. This increasing interest in data science is unlikely to fade in the near future with the rise of the internet of things and as more and more persons have access to computers, smartphones and others connected devices that generate data.

So while in the past the difficulty in data analysis fields like statistics was to obtain enough samples to have significant results that could be generalized with high confidence, in many cases today the problem may be switched to the opposite question: how to extract meaningful information from this deluge of data?

Beyond the problems of storing, cleaning, and handling such large datasets, another problem arising with this data abundance is that in some fields it is now usual to measure thousands of features for each sample. One difficulty is then to deal with this high features-to-samples ratio and the associated curse of dimensionality. The curse of dimensionality appears when the number of features, i.e. dimension, increases: the volume of the space increases quickly and the data points become isolated and sparse. The number of samples required to obtain statistically reliable results then grows exponentially with the dimension. For example, suppose we have 1000 features for 100 samples and we want to identify which features are related to a specific output. We can use a simple t-test to evaluate the relation between each feature and the output. If we select as relevant the features with an associated p-value less or equal to 0.05, then statistically we will have around 50 features that correlate with the output by chance. Appropriate methods are then necessary to deal with this high dimensionality: use adapted statistical tests and/or reduce this high features-to-samples ratio. To reduce this ratio, we can either increase the number of samples, either decrease the number of features.

A possibility to increase the number of samples is to aggregate datasets since it is increasingly common to have access to different datasets measuring the same kind of phenomena. For example, multiple batches of data can be collected from the same source at different times or multiple datasets can be collected at different sources measuring at least partially the same phenomenon, e.g., studies of a specific disease in different hospitals. Multiple datasets may also arise from different measurements collected from the same phenomenon but corresponding to various parameter values, e.g., simulation results of physical models depending on parameters. When dealing with a high number of features, it is tempting to merge all datasets to increase the number of samples and so increase statistical power. However, as those datasets were not collected in exactly the same conditions, large biases linked to the dataset membership can appear and should be taken into account in any analysis.

In this thesis, we concentrate on how to exploit information coming from different datasets in order to extract meaningful information. In particular, we address the problem of dataset integration using two approaches:

(I) merge all datasets by removing their differences (Chapter 2),

(II) or identify and keep the common part of the datasets (Chapter 3).

Another possibility to reduce the features-to-samples ratio is to reduce the number of features by computing a lower dimensional representation of the data that keeps most of the data characteristics. Beyond helping to handle the curse of dimensionality, this new representation can help to store and handle data efficiently, to make it easier to visualize and understand, to reduce noise or to reduce possible redundancy.

Dimensionality reduction is a large and well-studied field, and many methods exist in the literature. Those methods can be classified in two main families: linear and nonlinear methods. Nonlinear methods are more flexible, and can be very simple like a basic k-nearest neighbours where features are clustered based on distances between them, or more complex like kernel methods that first project the data in another space. Linear methods are restricted to extract new features that are linear combinations of the original ones. They are usually easier to manipulate, but by design are not able to capture nonlinear relationships between features.

We can also distinguish between feature extraction and feature selection. Feature selection selects a few of the original features to represent all of them, and is then a linear method. Feature extraction provides new features depending (linearly or not) on one or more original features.

When reducing the dimensionality of a dataset, it is important to keep in mind the final objective(s) which could be for example to visualize it, compress it, use it in a prediction model or understand it. In a significant number of cases, it is important to be able to interpret the derived features. This is necessary to understand which processes are behind the observed phenomenon, and/or to check that the derived features make sense. Interpretability of the derived features is especially important when the final goal is to provide those derived features to people outside the field to use in making decisions. It is important that those people can understand the logic behind the derived features to accept their use. A typical example is in medicine: who would like to take some specific medications without any rational explanations, just because some black-box model recommended it? Dimensionality reduction methods with the higher interpretability are arguably the feature selection methods, but such approaches are quite restrictive. When looking for interpretability, linear methods can be a good choice: more flexible but still easily interpretable.

Another interesting property to look for when reducing data dimensionality is the possibility to extend it easily to new data points. Many modern nonlinear dimensionality reduction techniques do not offer an explicit function linking original samples to their low-dimensional representatives: to extend those methods to new data points may require the minimization of a cost function, which can be computationally costly. On the contrary, linear methods are usually easy to extend.

In this thesis, in order to facilitate the interpretation of what is kept/removed in the dataset integration process, and to be able to extend easily this process to new data, we use linear methods linked to matrix factorization.

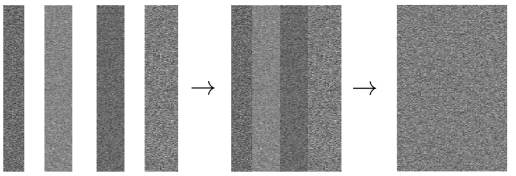
(I) In Chapter 2, we use low rank matrix factorization to represent the data as a linear combination of a small number of components. Each component can be interpreted as a new feature; the components associated to unwanted variations are then removed.

(II) In Chapter 3, we look directly for the components, linearly dependent on the initial features, that are present in all datasets.

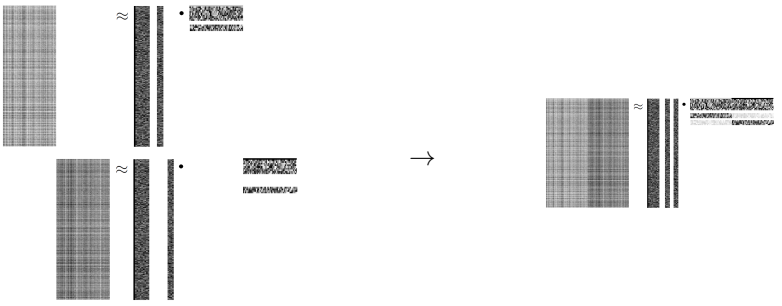
A schematic illustration of the models used in the thesis is given in Figure 1.1. In both cases those components are obtained from linear combinations of the initial features, so they can be easily interpreted. The second advantage is that representation of new samples using those features is also easy to obtain using a projection.

The two approaches are complementary. The second one is more specific than the first one in the sense that we are directly looking for a common part present in all datasets. The method returns this common low-dimensional part only. The assumption is that there exists a common signal which is present in all datasets, possibly with noise. If this signal is absent in at least one dataset, then approach of Chapter 3 is probably not adequate and could return useless components. On the contrary, the first method is based on weaker assumptions: the main hypothesis is that the matrix factorization used is able to find components representing unwanted effects that should be removed. If this is not the case, then nothing will be removed from the data and the datasets will not change.

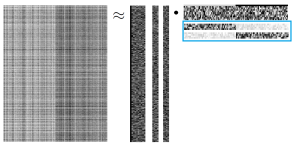
Objective: remove dataset variations specifically linked to the dataset membership.



Hypothesis: each dataset can be represented as a set of linear components, some of them being common to all datasets.

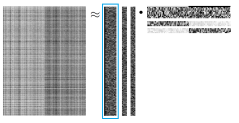


Methods: keep components not linked to variations across datasets.



Chapter 2: remove unwanted variations

or



Chapter 3: directly look for common components

Figure 1.1: Schematic illustration of the models used in the thesis.

Outline

We give here a short description of each section, along with the publications that were produced in the context of this thesis.

In Chapter 2, we investigate the case where we want to model the differences between the datasets and either integrate them as a factor in the analysis or remove them to obtain a normalized version of the data, that can then be used for further analysis. In the context of bioinformatics, more specifically in gene expression levels analysis, we investigate a matrix factorization approach with focus on a special case, a spatio-temporal version of the well-known Independent Component Analysis.

- Section 2.1 describes the general bioinformatics context: the way the data is generated and the accompanying challenges with a focus on batch effect removal.

- Section 2.2 focuses on a specific matrix factorization method called spatio-temporal ICA.
  Contribution: we develop an algorithm based on joint diagonalization that treats on an equal footing the spatial and temporal options. We then validate this factorization approach by modelling confounding factors and in a task of selecting differentially expressed genes.

  - Andrew E. Teschendorff, Emilie Renard, and Pierre A. Absil. "Supervised Normalization of Large-Scale Omic Datasets Using Blind Source Separation". In: *Blind Source Separation*. Ed. by Ganesh R. Naik and Wenwu Wang. Signals and Communication Technology. Springer Berlin Heidelberg, 2014, pp. 465–497
  - Emilie Renard, Andrew E. Teschendorff, and P.-A Absil. "Capturing confounding sources of variation in DNA methylation data by spatiotemporal independent component analysis". In: *22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN2014)*. 2014, pp. 195–200
  - Emilie Renard, Andrew E. Teschendorff, and Pierre-Antoine Absil. "Spatiotemporal ICA improves the selection of differentially expressed genes". In: *24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN2016)*. 2016, pp. 301–306

- In Section 2.3 the previous spatio-temporal ICA method is compared to other batch effect removal methods.
  Contribution: we conduct a more extensive comparison of the different batch effect removal methods in order to better distinguish their strengths and weaknesses, with a focus on the comparison of matrix factorization versus location-scale methods.

- Emilie Renard, Samuel Branders, and P.-A. Absil. "Independent component analysis to remove batch effects from merged microarray datasets". In: *Algorithms and Bioinformatics - 16th International Workshop*. Ed. by Martin C. Frith and Christian Nørgaard Storm Pedersen. Vol. 9838. Lecture Notes in Bioinformatics. Springer, 2016, pp. 281–292
- Emilie Renard and P.-A. Absil. "Comparison of location-scale and matrix factorization batch effect removal methods on gene expression datasets". In: *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 2017, pp. 1511–1518

In Chapter 3, we investigate the complementary approach, that is to directly extract the common part of the datasets.

– Section 3.1 gives the general context of the problem and its formulation as an optimization problem.
Contribution: we propose to formulate the problem as looking for a subspace minimizing the maximal dissimilarity between subspaces associated with the datasets.

– Section 3.2 gives a brief overview of useful optimization tools to handle the difficulties associated with the nonconvexity, nondifferentiability and constrained characteristics of the problem.

– Section 3.3 presents the various algorithms proposed to solve this problem.
Contribution: we develop five main approaches to tackle the problem: a greedy algorithm; an algorithm based on the first order condition for optimality; a projected subgradient; a descent direction approach with a proof of convergence; and an approximation by smoothing.

- Emilie Renard, Kyle A. Gallivan, and P.-A. Absil. "A Grassmannian Minimum Enclosing Ball Approach for Common Subspace Extraction". In: *Latent Variable Analysis and Signal Separation*. Vol. 10891. Lecture Notes in Computer Science. Springer International Publishing, 2018, pp. 69–78
- Emilie Renard, P.-A. Absil, and Kyle Gallivan. "Minimax center to extract a common subspace from multiple datasets". In: *27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning ,* 2019, pp. 275–280

– Section 3.4 details experimental results.
Contribution: we compare performances of the algorithms on synthetic data. The approach is then applied on gene expression datasets and on a parametric model order reduction problem.

Additional publications

We also had the opportunity to work on other subjects related to data analysis. More particularly, we worked on an extension of the well-known nonlinear dimensionality reduction method SNE from Kullback–Leibler divergence to a Jensen-Shannon divergence.

- John A. Lee et al. "Type 1 and 2 mixtures of Kullback-Leibler divergences as cost functions in dimensionality reduction based on similarity preservation". In: *Neurocomputing* 112 (2013). Advances in artificial neural networks, machine learning, and computational intelligence, Selected papers from the 20th European Symposium on Artificial Neural Networks (ESANN 2012), pp. 92–108

While such nonlinear methods are often used to project data in two or three dimensions in order to visualize them, some depend on parameters that can be difficult to tune a priori. As computing various projections for many parameters values can be computationally intensive, we proposed an interpolation-based method to better identify the parameters value(s) of interest while limiting the number of projections to compute.

- Emilie Renard, Pierre Dupont, and Michel Verleysen. "User control for adjusting conflicting objectives in parameter-dependent visualization of data". In: *VAMP 2013 (EuroVis 2013), Visual Analytics using Multidimensional Projections Workshop*. Leipzig (Germany), June 2013

We also had the opportunity to work with the medical doctor Emmanuel André on tuberculosis tests data. We analysed results from machines used to diagnose tuberculosis in the Democratic Republic of Congo in order to identify sooner the dysfunctional ones.

- Emilie Renard et al. "Failing Xpert MTB/RIF modules: When? What warranty is needed from the customer's perspective? Experience in the DRC". in: *The International Journal of Tuberculosis and Lung Disease* 19 (2015), S206

Chapter 2

Merging multiple datasets in genomic data: modelling batch effects via matrix factorization

With the development of bioinformatics, more and more gene expression datasets are available, which represent a powerful tool to better understand the mechanisms of life. However, in practice, analysis of such datasets should be carefully done due to their characteristics. For example, generation of such datasets cannot generally avoid the presence of noise, such as batch effects, and they often have large features-to-samples ratios that require statistically robust tools.

In this chapter, we examine the problem of batch effect removal when merging multiple gene expression datasets. We first describe in Section 2.1 the context: data generation procedures and the accompanying challenges with a focus on batch effect removal. Section 2.2 focuses on a specific matrix factorization method called spatio-temporal ICA. We first present the method, and then show that this approach can help deal with batch effects. We validate the method by modelling confounding factors in a task of selection of differentially expressed genes. In Section 2.3 a more extensive comparison of different batch effect removal methods is performed in order to better distinguish their advantages and weaknesses, with a focus on the comparison of matrix factorization versus location-scale methods.

2.1 Merging datasets in bioinformatics

Bioinformatics is a field that combines tools from statistics, computer science and mathematics to develop methods in order to better understand biological data. With recent development of tools like high throughput sequencing, an increasing amount of biological data is available. Being able to process such data is important since its understanding can literally save lives. The data we focus on in this chapter are gene expression datasets. In this section, gene expression analysis and its context are described with a focus on the problem of batch effect.

2.1.1 Gene expression datasets

Among all the biological data generated today, genomics is a promising field producing an increasing amount of data. Genomics is a biological field that aims to study the functions of an organism at the genome scale [168]. In contrast to genetics, that study one gene at a time, genomics looks at the complete set of DNA of an organism. While sequencing of genome (i.e., determine the sequence of nucleotides in DNA, that is the adenine A, guanine G, cytosine C and thymine T) remains a very important part of genomics, more and more scientists focus on functional genomics. Every cell in the body has the same DNA, but different proteins have to be produced depending on the cell and on the time. While structural genomics focuses on static aspects like DNA sequencing, functional genomics uses all the available data to study genes functions and interactions like gene transcription or translation, protein-protein interaction or gene regulation.

Gene expression. Functional genomics aims to study dynamical aspects of genomics, among which the link between genes and proteins. Proteins, that are essential to determine our tissues' properties, are produced from genes, that correspond to DNA fragments. The whole process going from DNA to protein is called gene expression [166, 69]. DNA fragments are first copied in a primary transcript of messenger RNA, pre-mRNA (transcription step). This pre-mRNA is then processed in a mature messenger RNA (RNA processing step), and the information is then used to build the corresponding proteins (translation step).

Since every cell contains the same DNA, the differences in functions lies in the regulation of gene expression that allows the production of a specific protein when it is needed [69]. Gene expression can be regulated by acting on any step of the process and the ability to measure gene expression can provide a large amount of valuable information. Nowadays, the development of sequencing technologies allows measuring gene expression levels at a reasonable cost. The analysis of the resulting data helps to better understand how genes work, with the goal of developing better cures for genetic diseases such as cancer.

DNA methylation. Transcription of genes are initiated by promoters, that are specific sequences occurring just upstream of coding regions of the associated genes [69]. Such promoters activation can be affected by methylation, a process in which a methyl group ($-CH_3$) is added to a cytosine base [153, 165]. DNA methylation is an epigenetic modification, i.e., the modification is reversible. Such modifications affect mainly the cytosine base of CG dinucleotides, usually called CpG. CpG sites are statistically under-represented in humans, but often concentrated in regions with high density of CpG, called CpG islands, which are mostly unmethylated. Many of the islands are located next to promoters. The CpG islands are then often associated to a gene, and the methylation can affect the expression of this gene (often, repressing the gene transcription): addition of this methyl group can alter the structure of a DNA region, modifying its accessibility and therefore its expression. For example, it is shown that in many diseases such as cancer, gene promoter CpG islands become hypermethylated [82]. DNA methylation of a particular class of genes (PolyComb Group Target) is also known to increase with age [151]. DNA methylation data have therefore been generating much interest in view of associations with diseases such as cancer, diabetes, and Alzheimer's [82, 48, 40, 17].

Microarray. One way to measure gene expression is to evaluate the quantity of mRNA produced. Hybridization microarray is a method used in expression profiling or high-throughput analysis of many genes at once to evaluate the amount of mRNA corresponding to a gene or a DNA sequence in a sample. A microarray is usually presented as a checkerboard of cells; each cell is filled with many copies of the same probe where each probe corresponds to a specific DNA sequence designed to be complementary to a specific gene. First mRNA is collected from the sample of interest and corresponding cDNA is extracted [140, Section 1.3]. This cDNA is then labelled, usually with fluorescent dies, and hybridized with the microarray probes. A washing step follows to remove unwanted DNA not complementary to the features on the array. The intensity of fluorescence on a spot, evaluated with image processing, is proportional to the abundance of the corresponding mRNA. Since all of these steps are prone to noise, at the very least, a basic normalization step is needed before any further analysis.

Each database thus takes the form of a p-by-n feature-by-sample matrix X, where p (the number of methylation sites or the number of genes) is typically around 20 000 and n (the number of individuals in the dataset) a few hundreds or less. Each value X_{ij} belongs then to $\mathbb{R}^+$ or $\mathbb{R}$, if the intensity scale is kept or a $\log_2$ transform is applied.

RNA-Seq. While we concentrate on microarrays datasets in this thesis, other techniques that can be used to study gene expression are described briefly here. A more recent technique is RNA-Seq, which is the process of using

next-generation sequencing (NGS) techniques to obtain a snapshot of the transcriptome at a given moment in time [37, 161]. The transcriptome corresponds to the set of all RNA molecules in one cell, and unlike the genome that corresponds to DNA, the transcriptome can vary with external environmental conditions. NGS technology reduces the time required by parallelizing the sequencing process. Unlike microarray methods, sequence-based approaches directly determine the cDNA sequence. Since RNA-Seq does not operate on a predetermined selection of cDNA probes, the detectable targets are not limited by the gene probes present on a microarray. The RNA-seq technique converts a sample of RNA to a cDNA library. The mRNA extracted from a sample is converted to cDNA using reverse transcription and sheared into fragments. Fragments within lengths of a certain range are selected and ligated with sequencing adapters, and then are sequenced and mapped against a reference genome.

Polymerase chain reaction. Another poplular method in molecular biology is Polymerase chain reaction which makes many copies of a specific DNA segment [105]. First the double-stranded DNA is denaturated by breaking the hydrogen bonds between complementary bases, in order to obtain two single-stranded DNA molecules. Secondly the annealing step allows primers to attach to each of the single-stranded DNA templates. In the extension step, the DNA polymerase synthesizes a new DNA strand complementary to the DNA template strand by adding free deoxyribonucleotides from the reaction mixture that are complementary to the template.

Differential analysis. Once the data is acquired and normalized, the analysis of the data can begin. An usual goal of gene expression experiments is differential analysis, that is performing statistical analysis to discover quantitative changes in expression levels between groups of samples, in order to identify which genes are linked to a specific clinical condition. The aim can be to identify biomarkers to help diagnosis or prognosis of diseases, to identify the class membership of a sample to determine the appropriate therapy, to monitor a response to a therapy, or to discover subgroups that share common features in a set of samples to understand the mechanisms involved in a disease. Basically, all these tasks require detecting the differences in expression between two (or more) conditions. A statistical test is then used to test the null hypothesis that there is no difference in expressions between the conditions. This is usually done using a basic t-test comparing the mean of two populations, however in gene expression analysis it is not always so simple. The main difficulty comes from the size of the data: the number of genes to test is generally much higher than the number of samples available, which implies using methods adapted to such large p small n cases.

Once a subset of differentially activated genes has been selected, another challenge is the validation of such a list. Sometimes the genes are subsequently studied on a case-by-case basis in new experiments in order to determine their

role in the biological processes but this is time consuming. Other possibly faster and simpler validation processes exist.

The first one is to compare the obtained subset to pre-existing lists of genes known to influence the process studied. A pathway is a series of molecules that interact in a cell and regulate, directly or indirectly, the gene expression and their associated products [118, 167]. Comparison of the selected genes to genes associated to known pathways is then a way to validate the obtained results.

Another possible validation if the final goal is diagnosis, prognosis or classification of samples, is to assess if using those specific genes improves performances in the corresponding prediction task. A simple example in a classification task would be to compare a classifier performances when using the selected genes or random ones to predict a specific phenotype of interest. In general, the phenotype of an organism corresponds to its physical characteristics (like eye color, heigth, or even some disease) resulting among others from its specific genetic encoding, or genotype [69]. In this thesis we mainly consider one specific characteristic at a time, that we call the phenotype of interest.

2.1.2 Confounding factors and batch effects

While increasingly cheaper and easier to obtain, gene expression datasets still require careful preprocessing and analysis steps to extract useful information. There are many sources of variations in expressions [38]. Biological variation may be influenced by genetic or environmental factors such as age [143]. Extraction, labelling and hybridization of samples lead to technical variation coming from the experimental conditions, the choice of the technological platform used, the protocol and analysis followed [75]. Measurement errors occur when reading the signals and may be affected by factors such as dust on the array. The statistical treatment in the post-processing step may also lead to variations in analysis results.

Sources of variations in measurements may also be classified depending on how they are integrated in a statistical model of the expression measures [90]. The primary measured variables represent the variables explicitly included in the statistical model used, for example the disease type. Such variables can be related to the expression variations, or not. The unmodelled factors represent sources of variations linked to some variables measured too but not used in the model. The reasons not to use such a variable in the model can be statistical (number of samples too low to include so many variables in the model), because the relationship between the variable and its effect on expressions is difficult to model or the variable may be subject to unacceptably large uncertainty or measurement error [150]. There are also the unmeasured factors that cannot be linked to measured variables in the study. Finally, the last type of variations is the noise independently present at each gene. When a variable is not directly available a surrogate may also be used. For example if we assume that experimental conditions does not vary too much on the same day, processing dates may be used as surrogate.

To limit the influence of all these factors and being able to extract analysis robust enough to be generalizable to other data, statistical inferences need a high number of samples. However, the size of such datasets is often limited due to the limited number of samples that can be processed at the same time in an experiment. As more and more datasets are available on public repositories such as GEO `http://www.ncbi.nlm.nih.gov/geo/`, merging and combining different datasets appears as a simple solution to increase the number of samples analyzed and potentially improve the relevance of the biological information extracted. Increasing this number enhances the statistical power, or probability to accept the alternative hypothesis H_1 when it is true, in following analysis. For the same effect size and the same level of type I error, i.e., α value, or probability to accept H_1 while H_0 is true, we have more chance to detect this effect if the samples number is higher.

When merging datasets, some of the main confounding factors are typically due to the fact that the samples were not processed in exactly the same conditions for all batch/datasets. Batch effects can be quite large and hide the effects related to the biological process of interest. Not including those effects in the analysis process may adversely affect the validity of biological conclusions drawn from the datasets [90, 91, 150, 138]. It is therefore important to be able to combine datasets from different sources while removing the unwanted variations such as batch effects. This implies the need for an appropriate normalization of the data before any further analysis. That is, to clear, as much as possible, genomics datasets from confounding sources of variation, without significantly obscuring biological variations of interest.

Batch effect removal is particularly challenging due to the many possible sources of variations that are unknown or only partly known through limited information, such as batch number and processing date. When studying a specific phenotype of interest, an additional difficulty in the process of removing batch effect is that this phenotype of interest could partially correlate with the batches [110]. For example, if we want to combine two datasets with respectively 75/25% and 25/75% of cases/controls, we should check that what is removed during the normalization step is really only the batch effect and does not contain potential useful information about cases/controls.

Different methods exist to tackle the problem of batch effect removal when merging different datasets, each having its strengths and weaknesses [85, 34]. They can be classified in two main approaches: location-scale methods and matrix factorization methods. The location-scale methods assume a model for the distribution of the data within batches, and adjust the data within each batch to fit this model. The goal is to obtain genes with similar mean and/or variance for each batch. A main expectation is that by adjusting the gene distributions no biological information is removed.

The matrix factorization methods assume that the variations across the datasets (biological or due to confounding factors) can be represented by a small set of components which can be estimated by means of matrix factorization. The components associated with the batch effects are then removed to obtain

the normalized dataset. With this approach, the main expectation is that the factorization method is able to pick up the batch effects in some of its resulting components. Among all those methods, most are unsupervised in the sense that after the batch effect removal step, any analysis can be applied. However, some methods are more specific and depend explicitly on the final analysis goal (typically, predict a particular feature of the samples). Some hybrid methods exist that mix location-scale and matrix factorization approach. A comparison of some methods and more detailed descriptions are given in Section 2.3.

2.2 Modelling confounding factors using Independent Component Analysis

The matrix factorization methods assume that the variations can be represented by a small set of components, that is, the sample expressions depend on a combination of hidden regulatory variables that inhibit or activate a subset of genes. Likewise, it is reasonable to assume that the batch effects are also due to a small set of components. Those components can for example be linked to the room temperature or platform type. Expression levels of genes are then seen as a linear function of common hidden variable that are related to different biological causes, that can be estimated by means of matrix factorization.

The measured dataset $X \in \mathbb{R}^{p \times n}$ can be approximated as the matrix product of two lower rank matrices $A \in \mathbb{R}_*^{p \times k}$ and $B \in \mathbb{R}_*^{n \times k}$, with $\mathbb{R}_*^{n \times k}$ the set of full rank matrices in $\mathbb{R}^{n \times k}$:

$$X \approx AB^\top \tag{2.1}$$

such that $X - AB^\top$ is reasonably small. $A(:, l)$, the lth column of A, can be interpreted as the gene activation pattern of component l and $B(:, l)$ as the weights of this pattern in the samples. Of course, such a decomposition is not unique, as we have $AB^\top = AWW^{-1}B^\top$ for any invertible matrix $W \in \mathbb{R}^{k \times k}$. Therefore additional conditions should be added to be able to compute a specific factorization.

In this section, we propose a new matrix factorization method, based on Independent Component Analysis (ICA), to represent the different signals present in a dataset. First we briefly introduce in Section 2.2.1 the Singular Value Decomposition (SVD), a tool that is useful in ICA and later in this thesis. The ICA framework is detailed with a focus on the JADE method in Section 2.2.2. The adaptation to the proposed spatiotemporal ICA is then described in Section 2.2.3. The method is validated on two cases: in Section 2.2.4 the computed components are compared to known confounding factors and in Section 2.2.5 some of those components are used as surrogate variables in a linear model in order to select differentially expressed genes.

2.2.1 Singular Value Decomposition

The most famous matrix factorizations are probably eigendecomposition for square matrices, and the Singular Value Decomposition (SVD) for any matrix. The SVD of $X \in \mathbb{R}^{p\times n}$ is

$$X = U\Sigma V^\top$$

with $U \in \mathbb{R}^{p\times p}$ and $V \in \mathbb{R}^{n\times n}$ orthonormal matrices, and $\Sigma \in \mathbb{R}^{p\times n}$ a matrix with nonnull coefficients only on the diagonal. Columns of U and V are respectively called the left and right singular vectors of X, while the diagonal entries σ_i of Σ are called the singular values. By convention those σ_i's are all positive or null and usually listed in decreasing order, the greater being called leading singular values. The number of nonzero singular values is the rank of X.

Since $XX^\top = UD^2U^\top$ and $X^\top X = VD^2V^\top$, U and V can be seen as the eigenvectors of $XX^\top$ and $X^\top X$ respectively, and the σ_i^2's as the corresponding eigenvalues. Depending on the context various versions of the SVD can be defined, we detail some below.

Thin SVD. For a matrix $X \in \mathbb{R}^{p\times n}$ with $p \geq n$, its thin SVD is given by :

$$X = U(:, 1:n)\Sigma(1:n, 1:n)V^\top$$

Compact SVD. For a matrix $X \in \mathbb{R}^{p\times n}$ with rank $r \leq \min(p, n)$, its compact SVD is given by removing the null singular values and associated singular vectors:

$$X = U(:, 1:r)\Sigma(1:r, 1:r)V(:, 1:r)^\top$$

k-truncated SVD. For a matrix $X \in \mathbb{R}^{p\times n}$, its k-truncated SVD ($k \leq r = \text{rank}(X)$) is given by keeping the k largest singular values and their associated singular vectors:

$$X \approx U(:, 1:k)\Sigma(1:k, 1:k)V(:, 1:k)^\top$$

with equality if $k = r$. The k-truncated SVD also corresponds to the solution of

$$\min_{U,\Sigma,V} ||X - U\Sigma V^\top||_F.$$

for $U^\top U = I_k$, $V^\top V = I_k$ and Σ a diagonal matrix of rank k. The k-truncated SVD is then often taken as a dimensionality reduction preprocessing step.

Dimensionality reduction. SVD has many applications in data analysis, one of the best known being Principal Component Analysis (PCA) and its use as a dimensionality reduction tool [24, Section 12.1]. For a (centered) dataset $X \in \mathbb{R}^{p\times n}$ of n samples with p features, PCA looks for a linear combination of the initial features maximizing its variance, in order to keep a maximum of

the information present in the dataset:

$$\min_{||w_1||_2=1} w_1^\top X X^\top w_1 = \min_{w_1} \frac{w_1^\top X X^\top w_1}{w_1^\top w_1}.$$

The optimal w_1 corresponds to the leading eigenvector of $XX^\top$, or first left singular vector of X. It is called the first principal direction, while $w_1^\top X$ is called the first principal component. Looking for the next components $w_i^\top X$ maximizing variance such that $w_i^\top w_j = 0$ for all $i \neq j$ gives the next left singular vectors of X.

PCA is often used as a dimensionality reduction preprocessing step by removing the last principal components, since it tends to minimize information loss under some assumptions [51]. If each feature in the dataset contains independent identically distributed Gaussian noise, the principal components will also contain similarly identically distributed Gaussian noise. However, PCA concentrates the variance in the first few principal components: they have a higher signal-to-noise ratio. PCA thus concentrates the signal into the first few principal components while the noise is more present in the later principal components. Removing those last component should then lead to a smaller loss of information.

Choice of k. In many algorithms used in this thesis, PCA or the k-truncated SVD is used as a preprocessing step to reduce dimensionality of the datasets analysed. An often crucial choice to make is the value of k. It must be small enough to reduce the size of data to manipulate but big enough to capture all the relevant information. The optimal choice of k is a field in itself, and many methods have been proposed.

An advantage of PCA is that the components may be computed by deflation: computing all k first components at once or computing the $k-1$ first ones, remove those components from data and compute the next component associated to this modified data gives the same answer. However, when PCA is only a preprocessing step and the resulting components are only a particular initialization step for a more complex method, this deflation approach is not always possible and k should be chosen beforehand.

The more common approach is to look at the distribution of the singular values and keep only the components associated with singular values of significant magnitude. In the presence of a large gap between the lth largest singular value and the next one, we can take $k = l$ and assume that the $l+1$ and following components represent noise or irrelevant information. Unfortunately it is more common to have a smooth decay in the singular values and choice of k is then less evident. A solution is to choose k in order to preserve at least a chosen proportion of the data variance. In some specific cases, for example when each point is interpreted as a subspace, an optimal value of the variance proportion to keep can be found [134].

Another possibility to evaluate k is Random Matrix Theory (RMT) [114,

150]. The number of components of a covariance matrix is estimated by comparing its observed eigenvalues to the distribution of the eigenvalues of a random correlation matrix. Such a distribution is known [114], and the number of eigenvalues larger than the theoretical maximum gives then an approximation of k.

More complex methods exists like Akaike or Bayesian information criterion, based on information theory. In this thesis we fix k arbitrary in most cases.

Computation. There are many approaches to compute the SVD with their efficiencies depending on the size of the matrix ($p \geq n$ or $n \geq p$), if only the singular values are needed, or if only a truncated version is necessary. For more details, see for example [53].

2.2.2 Independent Component Analysis

The SVD can be interpreted as looking for a another representation of the data that maximizes the second order statistic of each new component, that is the variance. ICA goes a step further with its goal is to maximize their statistical independence by means of higher order statistics. In this section, mainly inspired by [29, 131, 70], we introduce the basic principles of ICA.

ICA is a popular approach to model batch effects, as well as other technical and biological artifacts. ICA was first applied to microarray datasets in [93] where it is shown that the method can recover some experimental or biological effects. In [98], applying ICA to ovarian cancer data is shown to extract meaningful gene co-expression profiles. In [59] genes are clustered using their correlation with reference patterns obtained from ICA. In [88], various ICA methods are compared on microarray data in a gene clustering task. The usual method is to extract clusters of genes from columns of the obtained A in the model $X \approx AB^\top$ of Equation (2.1). A gene is considered as differentially activated in component l if its level in $A(:, l)$ is beyond some threshold, usually the mean or median plus or minus a few times the standard deviation. Many later studies investigated various versions and applications of ICA in gene expressions analysis, see for example [78, 175] for a review.

ICA can also be integrated within other methods. For example, it has been shown that in the surrogate variable analysis method (SVA) [90], replacing the underlying PCA by ICA—yielding a method dubbed ISVA—improves identification of confounders and subsequent statistical inference [150].

In the gene expression analysis context, independence is usually imposed on the columns of A. However it could also be imposed on the columns of B and this option is investigated in Section 2.2.3. In the following description of ICA approaches however, unless otherwise mentioned, independence is imposed on the columns of A. The main hypothesis in ICA is then that each signal $a_i = A(:, i)$ is statistically independent of a_j, $j \neq i$. In constrast to the SVD, ICA can be modelled in various ways and many implementations exist to solve the corresponding optimization problems, see for example [29, 131, 70] or [39]

for more details. The main idea is to look for a matrix W such that the new components defined by XW will maximize their statistical independence. Note that ICA components are fixed up to some indeterminacies: the order of the signals, and the magnitude and signs of the signals.

2.2.2.1 Preprocessing: whitening and dimensionality reduction

When looking for statistically independent signals, it is natural to ask for uncorrelatedness, i.e. requiring that the covariance matrix of the estimated A is the identity. To this end, a usual preprocessing step is the mean-centering of data, followed by a spatial whitening or standardization. In this thesis, unless otherwise indicated, centering will mean removing the mean from the data. The centered data X_c are then transformed by $Z = X_c W$ where W is chosen such that $\text{cov}(Z) = Z^\top Z = W^\top X_c^\top X_c W = I$. One usual way to achieve this goal is to take $Z = U$ with $UDV^\top$ an SVD of X_c. Of course imposing an identity covariance matrix does not imply a unique solution as we have $(Q^\top Z^\top)(ZQ) = I$ for any Q orthogonal. Most ICA algorithms aim to fix this matrix Q by minimizing some objective function $f(A = ZQ)$ measuring the independence, called a contrast function. Examples of such functions are given in the next section.

Another usual preprocessing step, especially when dealing with high dimensional data, is a dimensionality reduction step. Based on the hypothesis that the interesting processes are among the most present in the data, the complexity of the problem can be reduced by getting rid of the components associated with null or lower variance. This is easily done using a k-truncated SVD of X_c.

2.2.2.2 Estimating independence

Independence between two signals y_1 and y_2 is usually verified by the fact that their joint probability density function $p(y_1, y_2)$, abbreviated pdf, should be factorizable as

$$p(y_1, y_2) = p_1(y_1)p_2(y_2)$$

with $p_i(y_i)$ the marginal pdf of y_i. Since the pdf's are usually difficult to evaluate, it is Nongaussianity of the signals that is exploited instead [29, 131, 70]. Intuitively, we can use the Central Limit Theorem to recover the original Nongaussian signals. Indeed, the Central Limit Theorem tells us that under certain conditions the distribution of a sum of independent random variables tends toward a normal distribution. Thus, a sum of two independent random variables should have a distribution that is closer to a Gaussian than any of the two original random variables. Suppose we have observed signals X that are linear combinations of hidden independent signals A, that is $X = AB^\top$. Let us consider

$$y = XW = AB^\top W = AC$$

which is a linear combination of initial signals a_i's, with a_i representing the ith component stored in A. The only way to have y less Gaussian than each x_i is if coefficients in C are all null except one, that is y is one independent original signal [70].

A classical way to measure Gaussianity of a signal y is its kurtosis $\text{kurt}(y)$:

$$\text{kurt}(y) = E\left\{y^4\right\} - 3\left(E\left\{y^2\right\}\right)^2$$

with the operator $E\{y\}$ denoting the expected value of y. As with the skewness, the kurtosis describes the shape of a pdf and is related to tails of the distribution: a higher kurtosis is the result of more frequent outliers. After the centering and whitening of data, $E\left\{y^2\right\} = 1$ and therefore optimizing the kurtosis amounts to optimizing the fourth moment. Since kurtosis is null for Gaussian variables, ICA aims to maximize the absolute value of kurtosis.

Another family of approaches is based on the fact that among all random variables of equal variance, Gaussian ones have the largest entropy. This leads to the definition of negentropy J:

$$J(y) = H(y_{gauss}) - H(y)$$

with y_{gauss} a Gaussian random variable with the same covariance as y, and the entropy $H(y) = -\int p(y)\log p(y)dy$ where $p(y)$ is the pdf of y. $J(y)$ is always nonnegative and null if and only if y is Gaussian. As negentropy can be difficult to estimate it is usually approximated, for example, with

$$J(y) \approx \sum_{i=1}^{p} k_i \left(E\left\{G_i(y)\right\} - E\left\{G_i(\nu)\right\}\right)^2$$

where k_i are positive constants, $\nu \sim N(0,1)$, and G_i is a nonquadratic function.

Mutual information, based on entropy, can also be used to measure the dependence between random variables. It is always nonnegative and null if and only if the variables are statistically independent. Mutual information is closely related to negentropy if the signals are uncorrelated and of unit variance [29, 70].

2.2.2.3 JADE method

Among the most used method, JADE [29, 131] aims to maximize the fourth order auto-cumulants while minimizing the cross-cumulants, which are closely related to the kurtosis of the signal. JADE computes a rotation matrix that approximately joint diagonalizes this initial cumulant tensor.

Cumulants. Cumulants of a probability distribution are a set of specific quantitative measures of the shape of a function [39, Section 9.3]. They can be related to the moments of the distribution: the first cumulant is the mean, the

second cumulant is the covariance matrix, and the third cumulant is the same as the third central moment (higher-order cumulants are not equal to central moments).

Let x_i be the ith random variable of $X = [x_1, \ldots, x_n]^\top$. Cumulants of order two are computed as

$$\mathrm{Cum}(x_i, x_j) = E\left\{\bar{x}_i \bar{x}_j\right\}$$

with $\bar{x}_i = x_i - E\left\{x_i\right\}$. Fourth order cumulants are given by

$$\begin{aligned}\mathrm{Cum}\left\{x_i, x_j, x_k, x_l\right\} =& E\left\{\bar{x}_i \bar{x}_j \bar{x}_k \bar{x}_l\right\} - E\left\{\bar{x}_i \bar{x}_j\right\} E\left\{\bar{x}_k \bar{x}_l\right\} \\ & - E\left\{\bar{x}_i \bar{x}_k\right\} E\left\{\bar{x}_j \bar{x}_l\right\} - E\left\{\bar{x}_i \bar{x}_l\right\} E\left\{\bar{x}_j \bar{x}_k\right\}.\end{aligned}$$

Cumulants are closely related to kurtosis:

$$\mathrm{kurt}(x_i) = \mathrm{Cum}\left\{x_i, x_i, x_i, x_i\right\} = E\left\{\bar{x_i}^4\right\} - 3E^2\left\{\bar{x_i}^2\right\}.$$

Cumulants have nice properties such as multilinearity: under a linear transform $Y = XL$, we have

$$\mathrm{Cum}\left\{y_i, y_j, y_k, y_l\right\} = \sum_{pqrs} \mathrm{Cum}\left\{x_p, x_q, x_r, x_s\right\} L_{pi} L_{qj} L_{rk} L_{sl}$$

with L_{ij} the (ij)th entry of L. Moreover, independence of the a_i's implies

$$\mathrm{Cum}\left\{a_p, a_q, a_r, a_s\right\} = \mathrm{kurt}(a_p)\delta(p, q, r, s).$$

This gives for $X = AB^\top$ with independent entries a_i

$$\mathrm{Cum}\left\{x_i, x_j, x_k, x_l\right\} = \sum_{u=1}^{n} \mathrm{kurt}(a_u) B_{iu} B_{ju} B_{ku} B_{lu}. \tag{2.2}$$

For a random vector $X = [x_1, \ldots, x_n]$ and any matrix $M \in \mathbb{R}^{n \times n}$, the associated cumulant matrix $Q^X(M)$ is the $n \times n$ matrix defined by

$$\left[C^X(M)\right]_{ij} = \sum_{k,l=1}^{n} \mathrm{Cum}\left\{x_i, x_j, x_k, x_l\right\} M_{kl}. \tag{2.3}$$

If X is centered, it becomes

$$\left[C^X(M)\right]_{ij} = E\left\{(X^\top M X) X X^\top\right\} - R^X \mathrm{Tr}(M R^X) - R^X M R^X - R^X M^\top R^X$$

with $\left[R^X\right]_{ij} = \mathrm{Cum}(x_i, x_j)$, that is R^X is the covariance matrix of X.

So a given cumulant matrix may be estimated with a cost similar to estimating a covariance matrix as it is products of M with covariance matrices; there is no need for computing all fourth order cumulants to compute $C^X(M)$ for a fixed value of M. Estimating one cumulant matrix provides part of the fourth

order information on X. A maximal set of cumulant matrices is then a set of cumulant matrices $\left\{C_1^X, \ldots, C_P^X\right\}$ with $C_i^X = C^X(M_i)$ for $M_i \in \mathbf{M}$, where $\mathbf{M} = \{M_1, \ldots, M_P\}$ is an orthonormal basis for the linear space of $n \times n$ matrices; the corresponding C_i^X's will then represent completely the fourth-order information. Such a basis should contain n^2 matrices, however due to the symmetry present in cumulants, it is sufficient to consider only $n + n(n-1)/2$ symmetric cumulant matrices. Joint diagonalization of a maximal set guarantees blind identifiability of B if $\mathrm{kurt}(a_i) = 0$ for at most one signal a_i (this is a necessary condition) [30].

In the ICA model, using Equations (2.2) and (2.3) we have then for $X = AB^\top$

$$C^X(M) = B\Delta(M)B^\top$$

with

$$\Delta(M) = \mathrm{diag}\left(\mathrm{kurt}(a_1)B_{:1}^\top M B_{:1}, \ldots, \mathrm{kurt}(a_n)B_{:n}^\top M B_{:n}\right).$$

That is, B should be such that $B^{-1}C^X(M)B^{-T} = \Delta(M)$ for any M. Note that the kurtosis is present only in the diagonal matrix $\Delta(M)$. As it is usually impossible to diagonalize all C_i^X's at once, we try instead to minimize the off-diagonal elements.

JADE algorithm. The JADE algorithm is described in Algorithm 1. After the mean-centering and whitening of the data, a maximal set of cumulants matrices is computed. This allows the computation of the contrast function

$$\sum_i \mathrm{Off}\left(V^\top C_i^Z V\right)$$

where $\mathrm{Off}(Y)$ returns the sum of squares of the off-diagonal elements of Y. The Jacobi method is used to find a rotation matrix V minimizing the contrast function, i.e., a V that approximately jointly diagonalizes the set of cumulant matrices. The final factors A and B are then computed.

Algorithm 1 JADE algorithm

Require: Matrix X centered

1: Estimate a whitening matrix W, set $Z = XW$.
2: Estimate a maximal set $\left\{C_i^Z\right\}$ of cumulant matrices.
3: Solve $Q = \arg\min \sum_i \mathrm{Off}\left(Q^\top C_i^Z Q\right)$ using Jacobi method.
4: $A = ZQ^{-1}$, $B = W^{-T}Q^\top$

By selecting a maximal set of cumulant matrices, the joint diagonalization criterion is equivalent to the contrast function

$$\sum_{ijkl \neq iikl} \mathrm{Cum}^2\left\{(V^\top Z)_i, (V^\top Z)_j, (V^\top Z)_k, (V^\top Z)_l\right\} = f_{JADE}(V^\top Z).$$

Jacobi algorithm. The Jacobi method allows to optimize over the set of orthonormal matrices using an iterative approach [29, Section 4]. The orthonormal transform is seen as a sequence of 2D rotations, i.e., rotations applied to a pair of coordinates (y_i, y_j). The coordinates (y_i, y_j) are rotated through an angle θ_{ij}, minimizing the contrast function chosen, while leaving the other coordinates unchanged. Each $n(n-1)/2$ possible pairs of coordinates are considered to complete one sweep. Since the rotation of θ_{ij} can affect the previous rotations on i or j, a few sweeps across all pairs of coordinates may be necessary to complete optimization.

2.2.3 Spatiotemporal ICA method

When computing an ICA decomposition, the question arises whether one should minimize the mutual information between the columns of A or those of B in the decomposition $X \approx AB^\top$ [141, 152]. As suggested in [93], both options are justifiable a priori. Most studies use the "A" option, however, if there are reasons to believe that the technical factors and the factors of interest are less conditionally dependent given the feature, then the "A" option is used, and if this is the case given the sample then the "B" option is used. In earlier times, the very vertical shape of matrix X in genetic datasets—small number of individuals compared to the number of genes—naturally pointed to the "A" option, since A contains many more observations (rows) than B to which an estimator of mutual information can be applied. However, the applicability of this argument has waned with the rise of larger databases. It was shown in [76] that the "B" option can improve results in gene clustering, and in [132] that some intermediate options can help the recovering of existing pathways. ICA methods that support a trade-off between both options are called "spatiotemporal" methods. The term "spatiotemporal" comes from the pixel-by-time data in medical imaging for which the concept was introduced [141].

The main idea to extend ICA to spatiotemporal ICA is straightforward and takes as the contrast function a weighted sum of two contrasts functions [141, 152, 132, 76]:

$$f_{stICA}(X) = \alpha f_{ICA}(X) + (1-\alpha) f_{ICA}(X^\top).$$

The algorithm depends on a *spatiotemporal parameter* $\alpha \in [0, 1]$ that facilitates a continuum between imposing independence solely on A ($\alpha = 0$) and solely on B ($\alpha = 1$).

In [132], a continuum between the "A" and "B" options was investigated on gene expression databases using a spatiotemporal ICA method-based, in the spirit of the JADE method [30, 29], on joint diagonalization of cumulant matrices. The first step consists of centering the gene-by-sample data matrix X by subtracting the row and column means:

$$X_c = \left(I - \frac{1}{p}\mathbf{1}_p\mathbf{1}_p^\top\right) X \left(I - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\right)$$

where $\mathbf{1}_p$ denotes the vector of all ones in $\mathbb{R}^p$. The next step is a dimensionality reduction by means of a k-truncated SVD of X_c, yielding a new matrix

$$\tilde{X} = U_k D_k V_k^\top.$$

All the possible decompositions of $\tilde{X}$ are given by $\tilde{X} = AB^\top = AQ^{-1}QB^\top$ with Q a $k \times k$ invertible matrix. In [132], two approaches are proposed. A time-biased approach considers the model

$$\tilde{X} = \underbrace{U_k D_k Q^{-1}}_{=:A} \underbrace{Q V_k^\top}_{=:B^\top}$$

and minimizes the objective function

$$\begin{aligned} f_\alpha(Q) &= \alpha \sum_i \mathrm{Off}\left(C_i^{B^\top}\right) + (1-\alpha) \sum_i \mathrm{Off}\left((C_i^{A^\top})^{-1}\right) \\ &= \alpha \sum_i \mathrm{Off}\left(Q C_i^{V_k^\top} Q^\top\right) + (1-\alpha) \sum_i \mathrm{Off}\left(Q (C_i^{D_k^\top U_k^\top})^{-1} Q^\top\right) \end{aligned}$$

where Q is restricted to the orthogonal group $\mathrm{O}(k) = \{Q \in \mathbb{R}^{k\times k} : Q^\top Q = I\}$ and $C_i^X = C^X(M_i)$ where the C_i matrices define a maximal set. A space-biased approach is also described, that minimizes the same objective function but considers

$$\tilde{X} = \underbrace{U_k Q}_{=:A} \underbrace{Q^{-1} D_k V_k^\top}_{=:B^\top}.$$

The method was validated by assessing if known biological pathways were enriched in columns of A, and it was concluded that a markedly better enrichment may be observed at intermediate points of the continuum.

The method that we proposed in [124] eliminates the time- and space-biases discussed in [132] while staying within the orthogonal ICA framework. It is based on the decomposition

$$\tilde{X} = \underbrace{U_k D_k^\alpha Q^{-1}}_{=:A} \underbrace{Q D_k^{1-\alpha} V_k^\top}_{=:B^\top}, \quad Q \in \mathrm{O}(k). \tag{2.4}$$

Consequently, the columns of A and B are structurally decorrelated for $\alpha = 0$ and $\alpha = 1$ respectively. As a consequence, the new algorithm enjoys the following invariance property that is seemingly not present in other spatiotemporal ICA methods. If the input (X, α) yields the output (A, B), then the input $(X^\top, 1-\alpha)$ yields the output (B, A). In other words, the temporal and spatial flavors are treated on an equal footing.

In the spirit of the JADE ICA algorithm [29], the objective function to

minimize is of the form:

$$f_\alpha(Q) = \alpha \sum_i \operatorname{Off}\left(C_i^{B^\top}\right) + (1-\alpha)\sum_i \operatorname{Off}\left(C_i^{A^\top}\right)$$
$$= \alpha \sum_i \operatorname{Off}\left(QC_i^{D_k^{1-\alpha}V_k^\top}Q^\top\right) + (1-\alpha)\sum_i \operatorname{Off}\left(QC_i^{D_k^{\alpha}U_k^\top}Q^\top\right).$$

with $Q \in \mathrm{O}(k)$. The minimization of f_α is thus a joint approximate diagonalization problem, which is addressed in Algorithm 2 as in JADE using Jacobi rotations. The Jacobi algorithm is initialized with $Q = I$, ensuring that both A and B initially have decorrelated columns.

Algorithm 2 Spatiotemporal JADE algorithm

Require: Matrix X double-centered, dimension k

1: Set $A_0 = U_k D_k^\alpha$, $B_0 = V_k D_k^{1-\alpha}$ with $U_k D_k V_k^\top$ a k-truncated SVD of X.
2: Estimate two maximal sets $\mathbf{M}_A = \left\{C^{A_0}(M_i)\right\}$ and $\mathbf{M}_B = \left\{C^{B_0}(M_i)\right\}$ of cumulant matrices.
3: Solve $\hat{Q} = \arg\min_Q f_\alpha(Q, \mathbf{M}_A, \mathbf{M}_B)$ using Jacobi method.
4: Compute $A = A_0 Q^\top$, $B = B_0 Q^\top$

2.2.4 Modelling confounding factors

In this section, we present results from [124] building on [132] and [149, §7]. We study how the tradeoff between favoring independence on A versus B may affect the ability of an ICA method to model confounding factors. Specifically, we investigate how the tradeoff affects the correlation between each column of B and the beadchip effect. Beadchip effects are changes in gene expression levels linked to the position of the sample: the same sample in different physical positions on the array could be measured as different methylation levels [150]. This is because it is known which samples have been done on which beadchip and beadchip effects normally affect all samples on the chip. Hence, the confounder is known and the modelling of it by the ICA algorithm can be more objectively assessed. The presence of a high correlation between the beadchip and certain components suggests that cleaner data may be obtained by removing those components from the data.

Method. For each database, Algorithm 2 is applied to the p-by-n feature-by-sample matrix X, using the spatiotemporal parameter α and the number of components k to compute the matrices $A \in \mathbb{R}^{p\times k}$ and $B \in \mathbb{R}^{n\times k}$.

In order to assess the ability of the ICA algorithm to model confounding sources of variations, we measure the correlation between each column of B and a known source of confounding, namely beadchip variations. Since beadchip variations is a categorical information and the components $B[:,k]$ are continu-

Dataset	# features	# samples
UKOPSset1	25642	108
UKOPSset2	25642	148
T1D	22486	187
WBBC	26624	84
BCT	24589	113

Table 2.1: Number of features and samples for each dataset used.

ous, the usual correlation formula (Pearson or Spearman) cannot be used. To estimate which components are related to the beadchip effect, we compute the coefficient of determination R^2, that measures how well a variable x can predict a variable y in a linear model [159, Section 11.8]:

$$R^2(x, y) \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \qquad \in [0, 1] \tag{2.5}$$

where

- $SS_{tot} = \sum_i (y_i - \bar{y})^2$ is the sum of squares of the prediction errors if we take the mean $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ as predictor of y,
- $SS_{res} = \sum_i (y_i - \hat{y}_i)^2$ is the sum of squares of the prediction errors if we use a linear model $\hat{y}_i = f(x_i)$ as predictor.

The R^2 value indicates the proportion of the variance in y that can be predicted from x, and has the advantage of being easily generalized to categorical variables x. If x is continuous the prediction model is a linear regression and if x is categorical we use a class mean. So the higher the R^2 value, the better the association between both variables.

Since the beadchip information is categorical, $R^2(beadchip, B(:, k))$ compares the prediction of B_{ik} by beadchip mean $\sum_{j \in C_j} B_{jk}/n$ (where C_j represents all samples in the same beadchip as sample j) or by a a general mean $\sum_j B_{jk}/n$. An R^2 close to 1 means a high correlation between $B(:, i)$ and beadchip number, revealing that $B[:, i]$ is strongly affected by beadchip and thus presents a potential for modelling beadchip-related confounding sources of variation.

Data. We consider various DNA methylation databases where the samples are distributed over different beadchips with a maximum of 12 samples per chip. Tests were performed on DNA methylation datasets UKOPSset1, UKOPSset2, T1D, WBBC [149] and BCT [174]. The number of features and samples for each dataset is given in Table 2.1.

Results. In the central plot of Figure 2.1, for each database, the best R^2 value found over the k columns of B is given versus the value of the spatiotem-

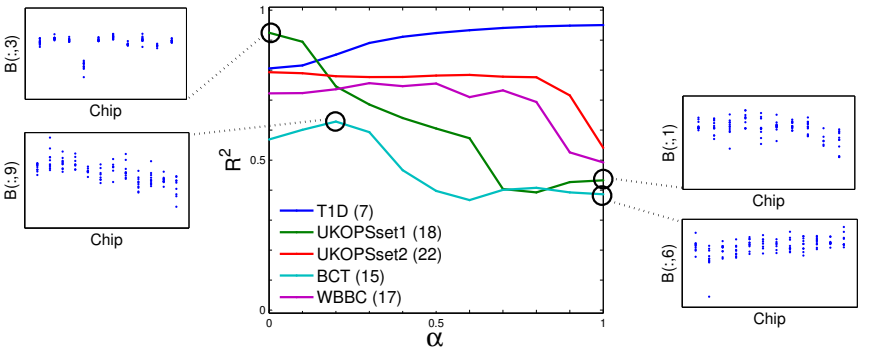


Figure 2.1: Central plot: for the various databases, maximal R^2 value with beadchip number as independent variable, as a function of α. The number k of components extracted is indicated between parentheses. The smaller graphics show how some R^2 values were achieved.

poral parameter α. The R^2 value is high in most cases, showing that at least one of the columns of B strongly correlates with beadchip. The trend is for R^2 to decrease as α goes from 0 to 1, as would be expected due to dimensions. However, the opposite trend is observed for database T1D, revealing that imposing independence on B has value for this database. The different behavior of T1D is probably linked to the fact that most of the variance present in the dataset can be associated to two confounding factors. The first component is associated with sex while the second one clearly separates the beadchips in two clusters: two beadchips appear to behave differently from the others, as can be seen on Figure 2.2. Those clusters are then more easily recovered, even for higher values of α. Such clusters are not present as clearly in the other datasets.

The smaller plots on Figure 2.1 plot the column of B that achieves the largest R^2 value against beadchip number, for a dataset and a value of α that can be read on the central plot. These smaller plots allow us to visualize how the R^2 value was achieved.

In Figure 2.1, the number k of components was estimated based on random matrix theory, as done in [149]. In other experiments reported in Figure 2.3, we investigate how the maximal R^2 value is influenced by the number k of components extracted. We observe that R^2 for fixed α is seldom monotonically increasing with respect to k. As a consequence, choosing k adequately is an important issue, and a comparison between Figures 2.1 and 2.3 reveals that random matrix theory produces a reasonable choice of k in our experiments. One also observes that R^2 does not necessarily depend monotonically on α, suggesting that the continuum between imposing independence fully across genes and across samples is worth exploring.

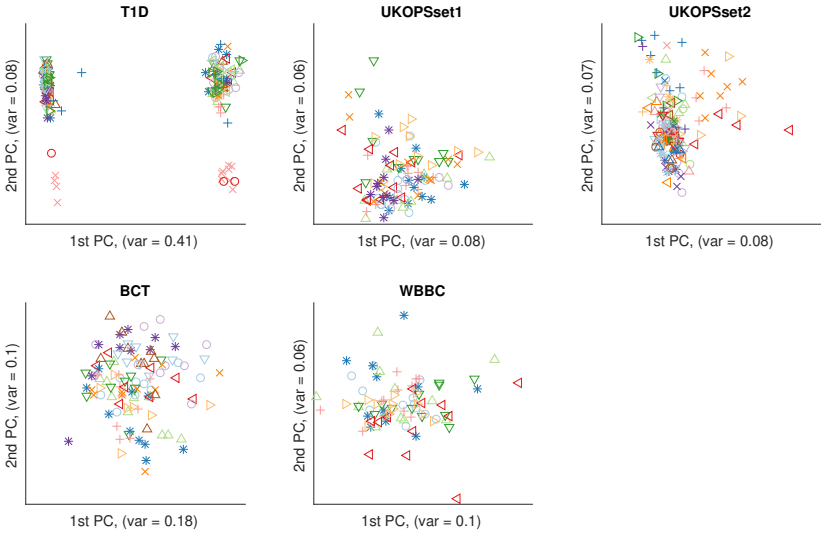


Figure 2.2: Plots of the first two principal components for the various datasets. Different types of marker and color represent a different batch number. The percentage of variance explained by each component is given in parentheses.

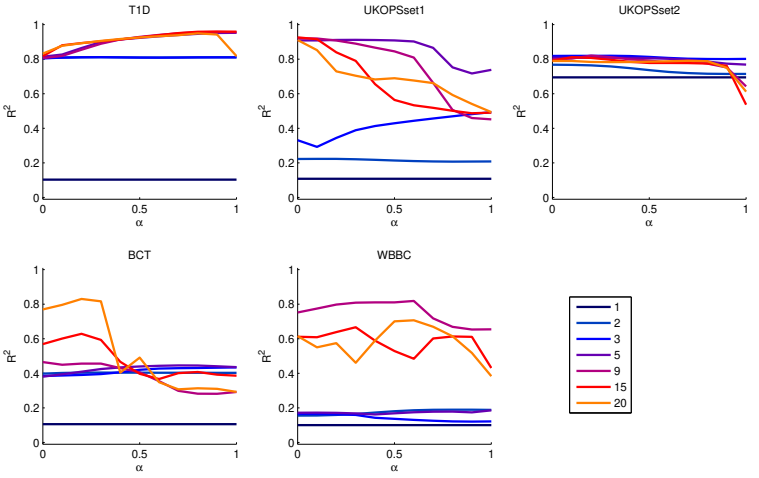


Figure 2.3: Maximal R^2 values of the columns of B in a linear ANOVA model with beadchip number as independent variable, plotted versus α for different real datasets, with the number k of components varying.

2.2.5 Selection of differentially expressed genes

Modelling confounding factors is one step in gene expressions analysis. The final goal is generally to use the potential confounding factors in a model to identify genes linked to a specific phenotype of interest. An example is to select a set of genes whose expression levels will help oncologists to determine the cancer subtype and therefore the best associated treatment.

As shown in Section 2.2.4 and in [149], varying the spatiotemporal tradeoff represented by the parameter α affects the results and can potentially improve the modelling of confounding factors. However, it is not always clear which value of α gives the best results or even if there is one optimal value. The main question is then: if there is no optimal value of α, can we aggregate information from different α values? Moreover, an important aspect in such methods is stability. A good algorithm should give similar results in similar conditions. In this section we present results of [125] that investigate this question based on a gene selection approach.

Method. As in [149], the underlying model assumes that the expression level of a gene is the result of interactions between different phenomena, biological or physical, where one in particular is of interest (the phenotype of interest, or POI) and the others are not. We cannot know exactly all such phenomena, but we can form an idea of the behavior of some of them by means of available external information (confounding factors, or CF) such as chip type and reagent quality. Under the assumption that those interactions are linear, the model can be written as:

$$\underbrace{X_{i:}}_{\substack{activation\\ levels\\ of\ gene\ i}} = f_i(\underbrace{\mathbf{y}^\top}_{\substack{phenotype\\ of\ interest}}) + \sum_{l=1}^{k} \mathbf{A}_{il} h_l(\underbrace{\mathbf{r}_l^\top}_{\substack{confounding\\ factors}}) + \underbrace{\epsilon_{i:}}_{noise} \quad i = 1, ..., p.$$

The $f(x)$ notation stands for a linear prediction model based on the knowledge of variables x. If x is continuous, a linear regression is used while a class mean is used if x is categorical.

Since we do not know exactly the effects $h_l(\mathbf{r}_l^\top)$ of the confounding phenomena, we try instead to find a basis of surrogate variables v_l that spans the same space, with possible insights from CFs:

$$X_{i:} = f_i(\mathbf{y}^\top) + \sum_{l=1}^{k} \mathbf{A}_{il} \underbrace{\mathbf{v}_l^\top}_{\substack{surrogates\\ variables}} + \epsilon_{i:}. \tag{2.6}$$

Under the assumption of linear interactions, the variables $\mathbf{v}_l$ can be inferred from X and $\mathbf{y}$ by means of a matrix factorization which is done here using the spatiotemporal ICA developed in [124].

The surrogate variables $\mathbf{v}_l$ are taken as the columns of the B matrix resulting from applying Algorithm 2 to X. Note that building the surrogate variables is unsupervised in the sense that no information from the POI or the CF is used in the spatiotemporal ICA method. Including in model 2.6 surrogate variables that correlate with the POI would remove biological information and prevent the detection of a link between $X_{i:}$ and y. To avoid this, we can check if the surrogate variables do not correlate strongly with the phenotype of interest. However, such a correlation could be due to a confounding factor. It is generally difficult to check if the confounding factors are unknown, however we can at least check for the ones that are known. That is, if the p-value of association between a surrogate variable and the POI is below some threshold (typically under 0.05), and if the surrogate variable is not more strongly associated with a confounding factor, then this surrogate variable is discarded. Such a p-value is computed using a t-test comparing the linear model

$$\mathbf{v}_l = \alpha + \beta\mathbf{y} + \epsilon$$

with $\beta = 0$ and $\beta \neq 0$, under the assumption that ϵ is normally distributed with 0 mean.

The final goal is to find a subset of genes differentially expressed with the POI. If gene i is differentially expressed with the POI y, then a good model of its behavior has to include y in the variables. That is, we must have that $\hat{X}^{(1)}_{i,:} = f(y, v_1, ..., v_k)$ gives a better approximation of $X_{i,:}$ than $\hat{X}^{(2)}_{i,:} = f(v_1, ..., v_k)$. The significance of the difference between both predictions is evaluated by computing associated p-values.

Since a p-value gives the probability of a false positive (saying that a gene is differentially expressed when it actually is not) on a single test and that many genes imply many statistical tests, the p-values are corrected in the corresponding q-values, that give the probability of false positive in all significant results [142].

As shown in Section 2.2.4, varying α allows recovering different components linked to confounding factors. So if a gene is selected in most cases corresponding to different α values, we can hypothesize that its link to POI is more robust to the presence of different confounding factors. So to improve the stability of the list of selected genes, we compare all sets of selected genes obtained for different values of the tradeoff parameter α. We retain as final selection the genes present in most of those lists.

To evaluate the final output, i.e., the list of selected genes, we compared it with lists of genes known to be differentially activated with the POI or CF using a hypergeometric test (such a list of genes is described in next paragraph). If we know the total number of genes N and among them the total number of POI-related genes L, the probability to have l POI-related genes in the n genes

chosen randomly is given by

$$P(l) = \frac{\binom{L}{l}\binom{N-L}{n-l}}{\binom{N}{n}}.$$

The hypergeometric test computes the probability that we observe m or more POI-related selected genes if we select n genes at random, i.e., it computes the p-value of $P(l \geq m)$.

Data. To validate our approach, we tested it on breast cancer expression. We combined different datasets which can be accessed under GEO numbers GSE2034 [160], GSE5327 [102], GSE7390 [42], GSE2990 [139], GSE3494 [101], GSE6532 [95] and from [156, 157].[1] ER status is known to correlate with tumour grade while cell-cycle related genes can separate low and high grade breast cancer independently of the ER status [95, 139]. So, as in [150], we took histological grade as the phenotype of interest, and oestrogen receptor status and tumour size as potential confounder. As reference genes related to confounding factors, we took genes linked to ER status [44, 157, 52]. For the POI, we took a list of cell-cycle related genes from https://reactome.org/. We removed all samples with missing information about grade or oestrogen receptor status to create a combined dataset of 1473 samples for 22282 genes.

Results. Figure 2.4 shows the list of genes with a q-value under 10^{-3} for 21 equispaced values of the spatiotemporal parameter α between $\alpha = 0$ (mutual information minimized solely on the first factor A in Equation (2.4)) and $\alpha = 1$ (solely on the second factor B). The main observation is that the list of selected genes differs from one α value to another, but with significant overlaps. Moreover, this overlap is proportionally bigger for POI-related genes than for others genes: $\sim 43\%$ of all POI-related genes are selected in 85% of cases, against $\sim 14\%$ for other genes. If we compare the number of genes selected in 85% of cases to the median over α of the number of genes selected (dark blue line), the proportions are respectively about 72% and 32% for POI-related and other genes. So returning the most stable genes across α values clearly improves the gene selection. The number of POI-related selected genes is reduced, but the number of non POI-related selected genes decreases much more. The proportion of selected genes that are POI-related nearly doubles with this method, which we term $\texttt{aggr}_1$.

Figure 2.5 shows a comparison of the proposed $\texttt{aggr}_1$ method with three other methods:

[1] We used the dataset from the *breastCancerNKI* R package, available on http://bioconductor.org/packages/breastCancerNKI/.

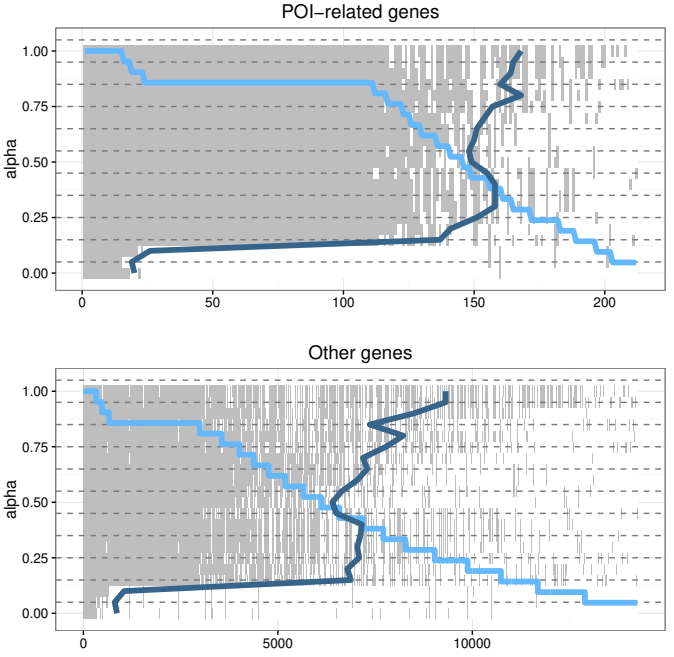


Figure 2.4: Selection of POI-related and other genes (p-value $\leq 10^{-3}$), depending on α (showing only genes selected at least once). A grey tile indicates that the gene was selected for the corresponding α value. The light blue curve shows the number of selections of each gene (in %). The dark blue curve shows the number of genes selected for each α value.

- `ISVA`, the method proposed in [149], which returns the list obtained with $\alpha = 0$;
- `LR`, where the gene selection is based on a linear regression with the POI only;
- `LR+CF`, where the gene selection is based on a linear regression with the POI and the CFs.

For the same q-value threshold, the methods select different numbers of genes. Unsurprisingly, $\texttt{aggr}_1$ selects among the lowest number of genes. All methods select similar proportions of CF-related genes for q-values thresholds larger than 10^{-3}. However, due to the high number of selected genes for `LR`, this proportion implies a much smaller p-value. Clearly, adding CF information in the model will improve the POI/CF distinction as can be seen when comparing p-values of the different models. We can see that for thresholds between 0.001 and 0.3, the proportion of POI-related genes among the selected genes is between 100% and 40% higher for method $\texttt{aggr}_1$ compared to the others. This higher proportion is translated in a smaller associated p-value.

2.2.6 Conclusion

Building on [132, 141, 29], an unbiased spatiotemporal version of the JADE ICA method eliminating the time- and space-biases is proposed. Testing on DNA methylation datasets shows that taking an α greater than 0, i.e., moving a bit away from the independence on "A" option, can improve correlation of the resulting components with known confounding factors. Using those components as surrogate variables in a linear model showed that using various α values can also help feature selection.

2.3 Batch effect removal: matrix factorization versus location-scale methods

In the previous section, we showed that components $B[:, i]$ recover, at least partially, signals present in gene expression datasets and that using those components as surrogate variables for potential confounding factor in the linear model (2.6) helps improve the selection of differentially expressed genes. Here we investigate if those components can be used to remove batch effects in a broader sense. Specifically, we examine the case where we want to analyse at once samples coming from different datasets, where each of those dataset is considered as a batch. If we consider directly the aggregated dataset made of the concatenation of all those batches, the main variations are generally linked to the batches themselves due to experimental conditions. The goal of batch effect removal methods is then to remove those variations in order to facilitate

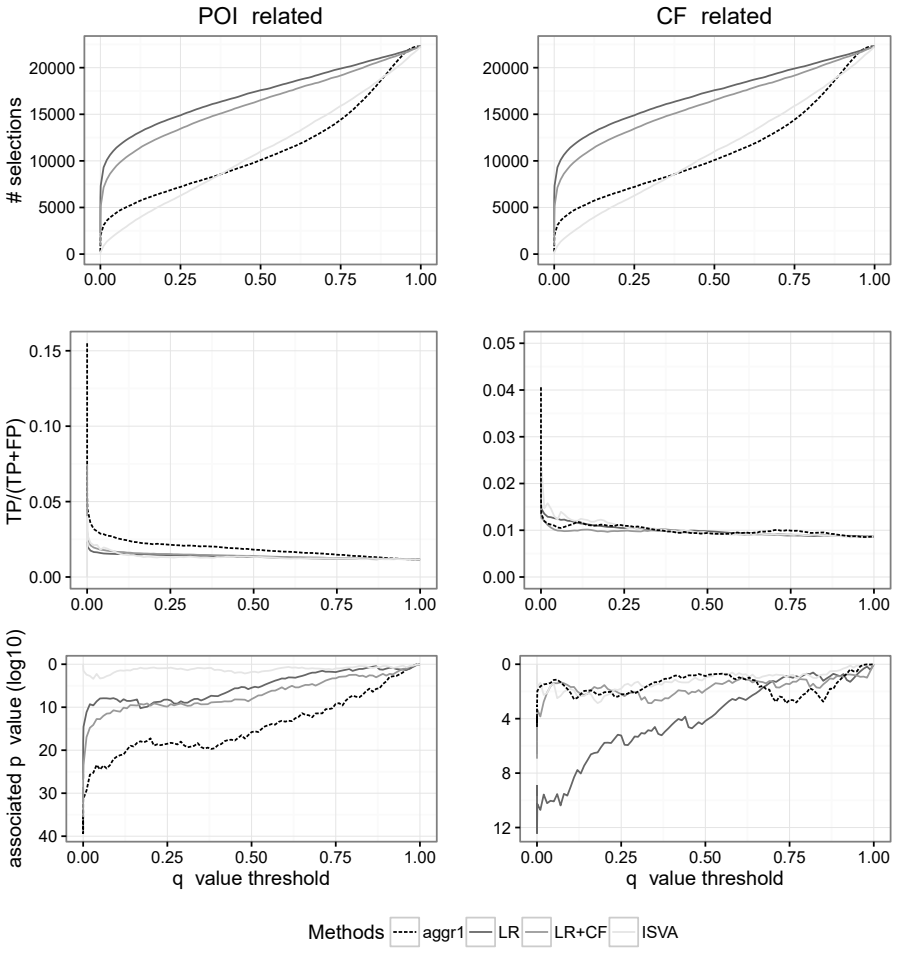


Figure 2.5: Number of genes selected, positive predicted value for POI or CF-related genes and associated p-values depending on the selection threshold.

further analysis. Most of the material in this section was published in [121, 119].

Available batch effect removal methods can be classified in two main approaches: location-scale methods and matrix factorization methods. The location-scale methods assume a model for the data distribution within batches, and adjust the data within each batch to fit this model. This approach is the most straight-forward one and many methods have already been proposed: XPN [135], DWD [19], ratio-based methods [96], ComBat [74], quantile-based methods [162], mean or median centering [136, 171], etc. The matrix factorization-basedmethods assume that the gene-by-sample expression matrix can be represented by a small set of rank-one components which can be estimated by means of matrix factorization. The components that correlate with the batch number are then removed to obtain the normalized dataset [8, 121]. In [91, 150], matrix factorization is used to model covariates in a differentially expressed gene (DEG) detection process. Matrix factorization-based methods are used less, probably because of the indirect approach to the problem: if no clear batch effect is recovered in the rank-one components, then the data matrix is left unchanged. However, not forcing the direct removal of differences between batches can be an advantage since it is more adaptable to cases where real technical artifacts are not exactly linked to the known batches.

Another option is to look directly for patterns present in all batches (or latent variables). Various implementation of such an approach have been proposed in the case of multi-omic data integration (combining data where the common dimension is the samples) and a recent adaptation was proposed in [129] in the specific context of classification based on independent datasets.

The presence or absence of batch effects in a dataset—and thus the effectiveness of batch effect removal methods—can be evaluated by different methods (see, e.g., [85] for an overview), which can be classified in three main groups: local approaches, global approaches and, in supervised cases, performance-based approaches. Global methods aim to illustrate the global behavior of the dataset: the evaluation of batch effect presence uses many features at once. For example, the global behavior of all genes can be summarized by a clustering dendogram or a plot of the first principal components. A clustering of samples by batch shows presence of batch effects, i.e., the predominant trend present in the data is linked to batches. However, it does not imply that all genes are affected to the same extent, or even affected at all.

Local evaluation methods examine the behavior of one gene at a time since expression levels of a gene should have the same behavior (typically similar probability distributions) for all batches. When evaluating a batch effect removal method, the clustering by batch and/or the differences in behavior across batches should disappear or at least be weaker.

These evaluation methods are unsupervised in the sense that there is no ground truth to which a comparison can be made. Therefore, the methods are unable to determine if the evaluated batch effect removal method has removed differences between batches that are due to technical artifacts (as they should)

or to biological differences between batches (as they should not). Batch effect removal methods that also remove differences between batches that have biological origins—a major danger when the batches are unbalanced—can thus be favored by such unsupervised evaluation methods.

If some ground truth is available, then a more meaningful evaluation of batch effect removal methods can be carried out. If some genes are known to be truly (not) differentially expressed, the proportion of those genes in the DEG list after removal of batch effect should ideally be higher (lower). P-values corresponding to null genes or negative control genes should be uniformly distributed across $[0, 1]$. The list of DEG should also be more stable from one batch to another. In prediction tasks where the final goal is, for example, to determine to which class a new sample belongs, performance should be improved after batch effect removal.

Comparisons of different batch effect removal methods can be found in [145, 96, 130, 135, 34, 85, 61]. While some methods appear more often among the best ones, there is not one best method identified in these studies since the context and objective of comparison varies. The set of methods tested varies from one study to another, but matrix factorization methods are generally less investigated.

In this section, we carry out a more thorough investigation of the impact of general purpose batch effect removal methods on sample classification tasks. We compare the two families of approaches (location-scale vs matrix factorization) to understand their advantages and weaknesses. For this we use nine different methods (15 with variants) and five classifiers, taking care of separating training and testing data from the very beginning (i.e., as an add-on method such as explained in [61]). Our salient finding is that some matrix factorization methods, though less widely investigated in the literature, have the edge over location-scale methods; see Section 2.3.5 for details.

2.3.1 Batch effect removal methods

Among the many batch effect removal methods mentioned above, we choose to investigate various well-known location-scale methods and a few matrix factorization-based ones. With the choice of those specific methods, we try to represent the different cases. Some methods, such as standardization, are very simple while others, such as ComBat, are more complex. FAbatch combines location-scale and matrix factorization approaches, and rgCCA is, to the best of our knowledge, tested for the first time in this context.

2.3.1.1 Matrix factorization-based methods

The main assumption is that the gene-by-sample matrix X can be represented using a low-rank factorization :

$$X \approx AB^\top = \sum_l A(:, l)B(:, l)^\top. \tag{2.7}$$

$A(:,l)$ can be interpreted as the gene activation pattern of component l and $B(:,l)$ as the weights of this pattern in the samples. For each l, the highly activated genes in $A(:,l)$ can be viewed as a group of genes working together in a specific condition, and $B(:,l)$ is the intensity of this condition among samples. The assumption is that some of the components represent the biological conditions of interest (and should be kept) while others represent the batch effects (and should be removed from the data). In other words, we are looking for a basis $U \subset A$ representing the gene activation patterns not linked to batch. This basis U can then be used in an "add-on" way as described in Section 2.3.3.1.

When dealing with different batches, two approaches can be taken. The first is to look directly for components present in all batches (or latent variables) such that $X_b \approx UV_b^\top$ (where b represents the batch) . The other option is to directly apply the factorization on X, the concatenation of the m batches, to obtain $X = [X_1...X_m] \approx AB^\top = A[B_1...B_m]^\top$ and then remove components correlating with batch number. Specific supervised methods were also developed to integrate class labels in the process (see Section 2.3.1.3 and use of information c_2 in Algorithm 3 for example).

Keeping common components. The first approach is inspired by Canonical Correlation Analysis [64], which aims to find linear combinations of the variables of two datasets with a maximum correlation. A generalization to more than two datasets was proposed by [146]:

$$\max_{t_b} \sum_{j \neq k} c_{jk} g(\text{cov}(X_j t_j, X_k t_k))$$
$$s.t. \quad \tau_b ||t_b||^2 + (1 - \tau_b) \,\text{var}(X_b t_b) = 1 \quad \forall b$$

where in our case X_b represents the matrix of expression values of batch b. The function g is usually the identity (the method is then called $rgCCA_{id}$), the absolute value or the square function (the method is then called $rgCCA_{sq}$). The parameter c_{jk} represents the link between batches j and k, and $0 \leq \tau_b \leq 1$ let us choose between the type of constraint on t_b and/or $X_b t_b$. We took all $c_{jk} = 1$ and $\tau = 1$, common basis U was then obtained as $U = XV(V^\top V)^{-1}$ with $V = [t_1...t_m]$.

Removing batch related components. A template for the second approach, adapted from [121], is detailed in Algorithm 3. The first step is to factor the matrix X (line 1). Many methods exist to factorize a matrix, depending on the properties the factorization components must fulfill. Imposing orthogonality among components leads to an SVD. Analysis of gene expression data using SVD was first proposed in [8]. Minimizing statistical dependence across component leads to ICA methods, which are shown to better model the different sources of variation than SVD [150]. Different variants exist for such methods depending on how statistical dependence is evaluated, and whether

we want to impose independence among genes or samples dimension, or even using a trade-off between both options [121]. Other factorization methods exist in the literature, such as Nonnegative Matrix Factorization [87] which can be used when dealing with nonnegative matrix values to obtain components with nonnegative values only.

Once the factorization is computed, we select the $B(:, l)$'s that correlate with the batch. If a component presents enough correlation with the batch (line 3), then this component is selected. Since batch is a categorical information and the $B[:, l]$'s are continuous, as in Section 2.2.4 we use the R^2 value defined in Equation (2.5) that measures how well a variable x (here, c) can predict a variable y (here, $B(:, l)$) in a linear model.

An additional step can be added in the process to check if the selected components do not correlate with some information of interest (lines 4-6, optional). The selected components are then removed from the matrix X to obtain a cleaned dataset (line 7). The common basis U is then generated by the matrix A cleaned from the selected components (line 8).

As matrix factorization methods, we tested the singular value decomposition (called SVD) and spatio-temporal ICA as presented in Section 2.2.3 with $\alpha \in \{0, 0.25, 0.5, 1\}$ (called $stICA_0$, $stICA_{0.25}$, $stICA_{0.5}$ and $stICA_1$).

Algorithm 3 Removing batch related components with matrix factorization

Require: $X \in \mathbb{R}^{p \times n}$ the aggregated dataset to be normalized, $c \in [1, \dots, m]^n$ a categorical variable indicating the batch number, $matfact$ the matrix factorization method, $t \in [0, 1]$ the threshold to consider a component associated to c, [optional] $c_2 \in \mathbb{R}^n$ categorical/continuous information that we want to preserve, $t_2 \in [0, 1]$ the threshold to consider a component associated to c_2

1: $A, B \leftarrow \texttt{matfact}(X)$
2: $R \leftarrow \texttt{cor}(c, B)$
3: $ix \leftarrow \texttt{which}(R \geq t)$
4: $R_2 \leftarrow \texttt{cor}(c_2, B)$ ▷ optional
5: $ix_2 \leftarrow \texttt{which}(R_2 \geq t_2)$ ▷ optional
6: $ix \leftarrow ix \setminus ix_2$ ▷ optional
7: $\hat{X} \leftarrow X - A(:, ix) * B(:, ix)^\top$
8: $U \leftarrow A(:, -ix)$

2.3.1.2 Location-scale methods

Location-scale methods are maybe the most intuitive way to handle the data. Choosing a reasonable model representing the probability distribution of gene expressions, and assuming that genes behave in the same way in each batch, expression values are adapted to fit this model within each batch. The goal is generally to let each gene have a similar mean and/or variance in each batch. A main hypothesis in such methods is that by adjusting the gene distributions

no biological information is removed, and that each batch groups samples with reasonably similar experimental conditions.

The simplest way to normalize a dataset in order to remove batch effects is to center (called *Centering*) or standardize (called *Std*) each batch separately. That is, for each gene in each batch, the expression values are centered, and divided by their standard deviation if desired.

Other simple approaches are ratio-based methods [96], which scale each value with a reference. Usual references can be arithmetic (called *RatioA*) or geometric (called *RatioG*) mean value of the corresponding variable in the same batch.

A widely used and more complex location-scale method is ComBat [74] (called *ComBat*). The expression value of gene i for sample j in batch b is modelled as

$$X_{bij} = \alpha_i + \beta_i C_j + \gamma_{bi} + \delta_{bi}\epsilon_{bij}$$

where α_i is the overall gene expression, and C_j is the vector of known covariates representing the sample conditions (such as batch membership). The error term ϵ_{bij} is assumed to follow a normal distribution $N(0, \sigma_i^2)$. Additive and multiplicative batch effects are represented by parameters γ_{bi} and δ_{bi}. ComBat uses a Bayesian approach to model the different parameters, and then removes the batch effects from the data to obtain the clean data $X^*_{bij} = \hat{\alpha}_i + \hat{\beta}_i C_j + \hat{\epsilon}_{bij}$. By pooling information across genes, this approach is more robust to outliers in small sample sizes.

Location-scale in a matrix factorization form. Some location-scale methods can be rewritten in a matrix form: using the same notation as ComBat, that is indices (b, i, j) denoting the $(batch, gene, sample)$, we have

$$\begin{aligned} X_{bij} &= \alpha_i + \beta_i^\top C_j + \gamma_{bi} + \delta_{bi}\epsilon_{bij} \\ &= \alpha_i + \begin{pmatrix} \beta_{1i} & \dots & \beta_{li} \end{pmatrix} \begin{pmatrix} c_{1j} \\ \vdots \\ c_{lj} \end{pmatrix} + \gamma_{bi} + \delta_{bi}\epsilon_{bij} \end{aligned}$$

where l is the number of the different sample conditions. The usual goal in batch effect removal methods is to estimate γ_{bi} and δ_{bi} to remove them from data to obtain normalized data $X^*_{bij} = \hat{\alpha}_i + \hat{\beta}_i C_j + \hat{\epsilon}_{bij}$.

The additive noise due to batches can be rewritten as

$$\gamma_{bi} = \begin{pmatrix} \gamma_{1i} & \dots & \gamma_{mi} \end{pmatrix} \begin{pmatrix} w_{1j} \\ \vdots \\ w_{mj} \end{pmatrix}$$

with m the number of batches and $w_{bj} = 1$ if sample j belongs to batch b,

0 otherwise. So we have:

$$X_{bij} = \underbrace{\begin{pmatrix}\alpha_i & \beta_{1i} & \dots & \beta_{li} & \gamma_{1i} & \dots & \gamma_{mi}\end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix}1\\ c_{1j}\\ \vdots\\ c_{lj}\\ w_{1j}\\ \vdots\\ w_{mj}\end{pmatrix}}_{C_j} + \underbrace{\delta_{bi}\epsilon_{bij}}_{E_{ij}}$$

Since knowing j allows us to know b, it sums up to the model

$$X = AC + E \tag{2.8}$$

with A_i the ith row of matrix A and C_j the jth column of C.

While matrix factorization methods look for more global constraints on A and C like orthogonality or statistical independence, location-scale methods imposes matrix C and evaluate A based on the chosen assumptions. For example, in mean-centering, γ_{bi} is the mean of X_{ij} on all samples j belonging to batch b and $\delta_{bi} = 1$. Its variant, median centering, takes γ_{bi} as the median of X_{ij} on all samples j belonging to batch b. Standardization choses γ_{bi} as the mean of X_{ij} on all samples j belonging to batch b and δ_{bi} as the standard deviation of X_{ij} on all samples j belonging to batch b. ComBat is a bit more sophisticated. Parameters values γ_{bi} and δ_{bi} are both estimated using priors by pooling information across genes, and, in contrast to to the previous methods, it explicitly allows modelling possible phenotypes in C.

2.3.1.3 Supervised methods

A well-known batch effect removal method is surrogate variable analysis [90] (called SVA), which combines SVD and a linear model analysis to estimate the eigengenes using a residual expression matrix from which biological variation has already been removed:

$$x_{ij} = \mu_i + f_i(y_j) + \sum_k \lambda_{ki} g_{kj} + \epsilon_{ij}$$

with $\mu_i + f_i(y_j)$ a linear regression to predict x_{ij} from the phenotype y_j, and surrogate variables g_{kj} estimated using SVD.

Similarly, a method combining latent factor adjustment and location-scale was proposed in [60] (called FA):

$$x_{bij} = \alpha_i + \mathbf{a}_{jb}^\top \beta_i + \gamma_{bi} + \sum_{l=1}^{k_b} b_{bil} Z_{jbl} + \delta_{bi}\epsilon_{bij}$$

with $Z_{jbl} \sim N(0,1)$ for all j in $1, ..., k_b$ and $\epsilon_{bij} \sim N(0, \sigma_i^2)$. All known factors of interest, such as the phenotype to predict, are represented in the term $\mathbf{a}_{jb}^\top$. Latent variables, including batch effects, are represented by $b_{bil} Z_{jbl}$ and estimated using an expectation-maximization algorithm.

The last two batch effect removal methods are supervised in the sense that they explicitly integrate information about the phenotype to predict in their models. It is, of course, possible to go a step further by computing specifically components correlating with this information, as done in [129]. FA and SVA can be seen as weakly supervised in the sense that they try to avoid removing the class information, while [129] aims to find components specifically representing this information and is then more strongly supervised. The method in [129] is then less general than the ones detailed previously, so we did not consider it as a general purpose batch effect removal method.

2.3.2 Global effect of batch effect removal methods

Data. We tested all methods on breast cancer expression. More details on the datasets used is given in Table 2.2. We consider each dataset with a specific platform as a batch. Features of the different datasets were matched using the common Entrez ID, and samples and features with more than 1% and 10% of missing information, respectively, were removed, giving an aggregated dataset of 9371 genes and 2209 samples. The missing information was imputed for each batch using the 'impute' R package. The 1% and 10% thresholds were chosen arbitrarily, in order to remove data with too many missing values but still keep a significant number of points.

Some of the datasets include additional pieces of information such as age of the patient, grade and size of the tumor, if a lymphatic node is affected, the estrogen receptor status, the treatment and the subtype. We took estrogen-receptor status prediction as the classification task. ER+ and ER- denote positive and negative estrogen-receptor status respectively. The proportion of ER+ samples depends on the dataset but is almost always in majority (exact numbers are detailed in Table 2.2).

Applying batch effect removal methods on the whole dataset. We first investigated the global effect of the batch effect removal methods by applying each one on the whole dataset (so the 15 batches concatenated). Effects of different normalization methods on the association between genes and batches or ER are shown in Figure 2.6. Compared to the initial values, all methods increase the maximum R^2 value associated with the ER factor, SVD and ICA more than double it. Effects on association with batches are more varied. FA and ComBat remove all association with the batches. The methods based on matrix factorization are less sharp, the association with batches decreases as α increases.

To evaluate more globally the presence of batch effects, we plotted the first two principal components of the datasets in Figure 2.7. Those components cap-

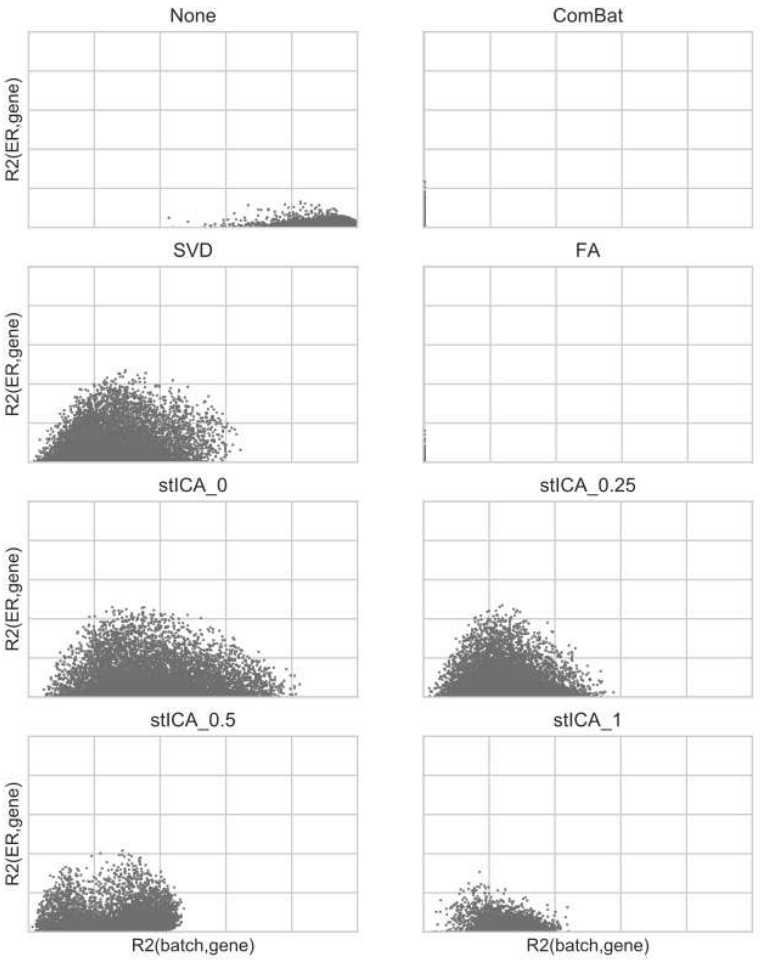


Figure 2.6: R^2 values between the gene expression values and the batches versus the ER factor, for different normalizations of all aggregated dataset.

	Batch	Technology	# ER-	# ER+
Train	$GSE2034_{96}$	Affymetrix	77	209
	$GSE21653_{570}$	Affymetrix	113	150
	$GSE2990_{96}$	Affymetrix	34	85
	$GSE17040_{887}$	Agilent	11	44
	$GSE1992_{887}$	Agilent	16	23
	NKI2	Agilent	43	159
Test	$GSE17705_{96}$	Affymetrix	0	298
	$GSE3494_{96}$	Affymetrix	34	213
	$GSE5327_{96}$	Affymetrix	58	0
	$GSE6532_{570}$	Affymetrix	0	87
	$GSE6532_{96}$	Affymetrix	11	115
	$GSE7390_{96}$	Affymetrix	64	134
	$GSE32641_{887}$	Agilent	84	0
	$GSE8465_{887}$	Agilent	17	16
	NKI	Agilent	40	74

Table 2.2: Summary of batches used, named after their GSE references and the platform used.

ture most of the variability in the data. In Figure 2.7, we can see that without removal of batch effect (subplot *None*), the first two principal components of the datasets show a global clustering by batch. Moreover, all Agilent datasets are clustered together on the right. The two batches corresponding to the same platform (GPL570) are also close.

The three other subplots give an idea of how the different types of batch effect removal method work. Clearly, applying a batch effect removal method reduces the batch clustering, and even tends to cluster samples by ER status. Note that by forcing each dataset to have the same mean, ComBat cannot merge the two clusters present in $GSE6532_{96}$. Such distinct clusters are no longer present for SVD. However, samples belonging to the same batch still tend to stay in the same neighborhood, probably because the batch information is not directly used when computing the new representation of the data. In FA, which combines location-scale and matrix factorization approaches, there is no clustering by batch.

In Figure 2.8 the effect of using c_2 information to prevent the removal of a component possibly related to the POI is illustrated. This step is of particular interest if the POI and the batches could be correlated, as shown in Figure 2.8. The dataset used was subsampled to artificially introduce correlation between batches and ER status (dataset described in Table 2.3). The right plot corresponds to the initial dataset, and clearly shows clustering by batches. The central plot corresponds to the results of Algorithm 3 with SVD for $(t, t_2) = (0.5, 1)$ (so without c_2), shows no clustering by batch or by ER

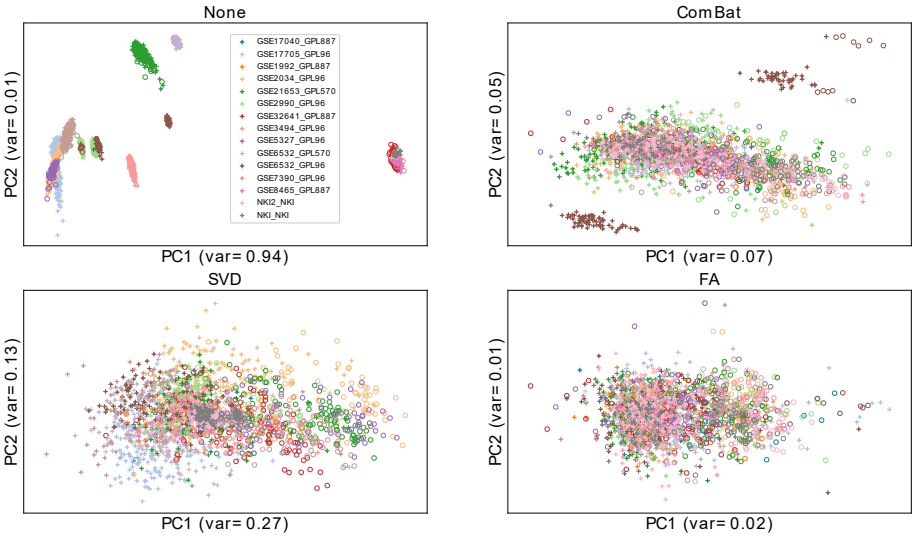


Figure 2.7: Global evaluation of batch effects: plots of the first two principal components before and after batch effect removal by ComBat, FA and SVD. Colors represent the batch membership, o/+ corresponds to ER-/ER+ status. The percentage of variance explained by each component is given in parenthesis.

status. The left plot corresponds to the results of Algorithm 3 with SVD for $(t, t_2) = (0.5, 0.3)$, which corresponds to an intermediate situation. The clustering by batch is less pronounced while the clustering by ER status is more pronounced.

To investigate more the influence of the parameter α on stICA, we computed the ICA factorization of the unnormalized aggregated dataset for different values of α, yielding the components $A(\alpha)$ and $B(\alpha)$ as in Equation (2.7). The maximal R^2 values between components of the matrix B and the external information, i.e., $\max_i R^2(info, B_{.i})$, are shown in Figure 2.9. The quality of the recovery of the external information depends on the α value. ICA and SVD recover the external information with similar R^2 values. While association to ER is higher for SVD, association to $Size$ and $Node$ is higher for ICA. The influence of batches appears to be captured in the SVD decomposition, and in all values of α in ICA. However if we examine the relation between components and external information the behaviors differ.

In Figure 2.10 we show for different decompositions the R^2 values between all components and the external information, and the correlation between the components themselves. SVD gives six components highly associated to batches. ICA increases the number of components associated to batch, especially for α values close to 0. However, some of those components are redundant: for $\alpha = 0$, components 11 and 13 have high R^2 values but are correlated with each other. Increasing the value of α increases the required independence

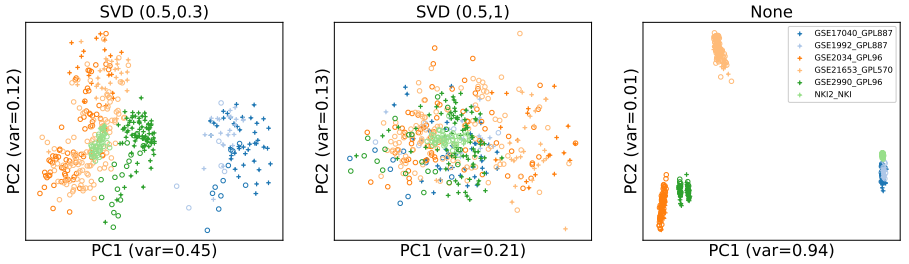


Figure 2.8: Illustration of the use of c_2 information in Algorithm 3 on a dataset presenting correlation between the batches and the POI (dataset described later in Table 2.3). Global evaluation of batch effects: plots of the first two principal components before and after batch effect removal using SVD with $(t, t_2) = (0.5, 0.3)$ and $(0.5, 1)$. Colors represent the batch membership, $o/+$ corresponds to ER-/ER+ status. The percentage of variance explained by each component is given in parenthesis.

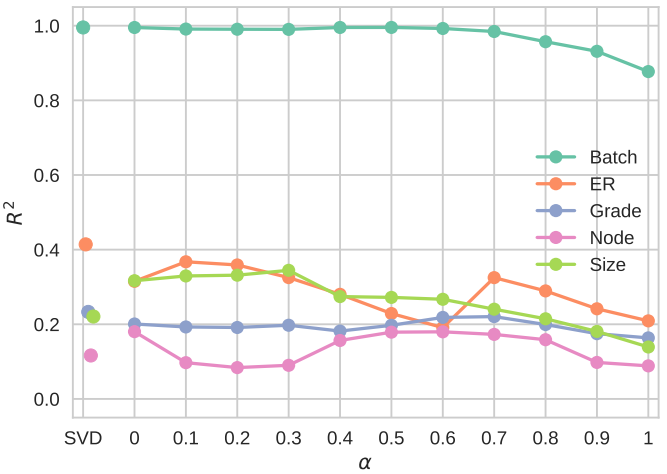


Figure 2.9: Maximal R^2 values between components of the ICA factorization and the ER, Grade, Node, Size and Batch information depending on α. The isolated dots on the left hand side give the same information but for the SVD decomposition.

on B and so enables reduction of the redundancy between components. A good trade-off between recovering external information and avoiding redundancy would be an intermediate value of α.

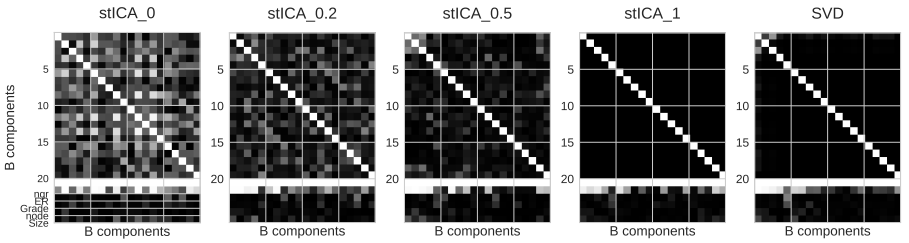


Figure 2.10: Top: Pearson correlation coefficients between the different components. Bottom: R^2 values between components and the external information. Black implies a null correlation (0), and white a perfect one (1).

2.3.3 Validation by impact on classification

We tested the batch effect removal methods SVD, $stICA_0$, $stICA_{0.25}$, $stICA_{0.5}$, $stICA_1$, $rgCCA_{id}$ and $rgCCA_{sq}$ for matrix factorization; and $ComBat$, Std, $Centering$, $RatioA$ and $RatioG$ for location-scale. We also tested SVA and FA, two methods using class information, and the case without batch effect removal (called $None$) as reference.

2.3.3.1 General procedure description

To compare the different batch effect removal methods, we used them in a classification process where the ER status is predicted[2]. The ER status is thus the phenotype to predict, i.e. the outcome. The whole process is described in Algorithm 4. The first step is to separate training and testing sets (lines 1–2). To limit disparity of the samples in the training set, we chose three datasets using the Affymetrix technology and three using the Agilent one. Each training dataset contains samples of both classes, with at least 20% of ER- (see Table 2.2). Next batch effects are removed from the training datasets with the chosen method (line 3). Optionally, the training labels can be used as extra information to preserve. This was only done for SVA and FA in the experiments below. For all other methods, c_2 is not used in Algorithm 3. In matrix factorization-based methods, we computed the $k = 20$ first components and removed the components with an R^2 value higher than $t = 0.5$.

A basic feature selection is performed by selecting the genes with the best association with the ER label (line 4). For each gene, the mean activation levels of ER+ samples is compared to the mean of ER- samples using a t-test with unequal variance. Genes with associated lower p-values are then selected.

A classifier is trained based on the selected genes (lines 5–6). When training the classifier model, training data are first mean-centered and scaled to ensure the equally weighted treatment of all features.

[2]Implementation of Algorithm 4 is available on `https://sites.uclouvain.be/absil/`

Batch effects are then removed from the test dataset X_{te} (line 7) using an add-on counterpart of the chosen method. In case of a matrix factorization-based approach, in order to comply with the add-on requirement [61], the batch effect free dataset $\hat{X}_{te}$ is computed using the projection of X_{te} on the basis U (see Section 2.3.1.1):

$$\hat{X}_{te} = U(U^\top U)^{-1}U^\top X_{te}. \tag{2.9}$$

After centering and scaling, the ER labels of the testing set are predicted using the classifier (line 8).

We repeated lines 5 to 8 using different numbers of selected genes (line 4) and different classifiers (line 6). We started with 2000 features, repeatedly removing 20% of them until reaching less than 150 features; then we removed 10% until a minimum of 10 features.

In order to comply with the 'add-on' scenario [61] where the model is built once and for all on some studies and then validated on other separate studies, the parameters of batch effect removal methods and classifiers are estimated using only the training datasets. We applied the whole procedure on any combination of four training batches ($C_6^4 = 15$ experiments), and then tested it on all the nine testing batches (see Table 2.2).

Algorithm 4 Classification process

Require: $X \in \mathbb{R}^{p\times n}$ aggregated matrix of gene expression, $y \in [-1,1]^n$ the label to predict, $c \in [1,\dots,m]^n$ the batch information

1: $y_{tr}, y_{te} \leftarrow y$
2: $X_{tr}, X_{te} \leftarrow X$
3: $\hat{X}_{tr}, param_{BE} \leftarrow \texttt{BEremoval}(X_{tr}, c, y_{tr})$
4: $idx_{bestGenes} \leftarrow \texttt{ttest}(y_{tr}, \hat{X}_{tr})$
5: $X_{class} \leftarrow \hat{X}_{tr}[idx_{bestGenes}, :]$
6: $model_{class} \leftarrow \texttt{classifier}(X_{class}, y_{tr})$
7: $\hat{X}_{te} \leftarrow \texttt{BEremoval}(X_{te}, param_{BE})$
8: $\hat{y}_{te} \leftarrow \texttt{prediction}(model_{class}, \hat{X}_{te}[idx_{bestGenes}, :])$

We made specific choices to define the classification process and there exist many possible others choices to explore. We choose parameter values that seemed reasonable, but of course other values could have been tested such as the number of components k or the threshold of $t = 0.5$. Some steps could also have been replaced by similar procedures, for example selecting the components to remove based on a p-value and not on R^2.

Since it was shown in [61] that add-on batch effect removal generally improves classification performances, we chose to apply the model built in the training test to clean test datasets before the classification step. This relies on the assumption that general behavior of training and testing set are similar, which does not necessary always holds. Another possibility is to directly apply the classification model without the add-on batch effect removal of line 7.

Since stability and reproducibility are also important points, another interesting experiment is to compare the list of selected genes in line 4, for example with the stability measure used in [1].

We considered a different classifier for each of the $C_6^4 = 15$ experiments to investigate the influence of the corresponding subset of training sets on performance. However, another possibility is to evaluate each of the 15 corresponding models obtained from the two remaining training sets and to build from them a final model to apply on the tests sets. This could, for example, simply be an average of all models, or using a smart subset of features based on a stability measure.

2.3.3.2 Classifiers tested

In order to cover a large spectrum of the existing classification approaches, five different classifiers were tested: k-Nearest Neighbors, Support Vector Machine, Naive Bayes, Random Forest, and Adaboost with logistic regression (using their scikit-learn implementation [111]). General principles of those classifiers are given here, more details can be found, for example, in [57].

A **k-Nearest Neighbors** (knn) classifier simply looks for the k nearest neighbors of the point to classify, and assigns as the label the one which has the most representatives within those neighbors. We used the basic Euclidean distance with $k = 20$.

A **Support Vector Machine** (SVM) classifier looks for a hyperplane providing the best separation between the two classes, that is such that the distance from the hyperplane to the closest points is maximized. To improve classification performance, the points can first be projected in a higher dimensional space using the kernel trick. Here, as the number of features can be high compared to the number of points, we used a linear kernel. To take into account the unbalanced repartition of labels, classes are given weights inversely proportional to their frequencies.

A **Naive Bayes** (NB) classifier is a probabilistic classifier based on Bayes theorem

$$p(y|x_1, ..., x_p) = \frac{p(y)p(x_1, ..., x_p|y)}{p(x_1, ..., x_p)}.$$

Assuming that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$ given the class y, we can write

$$p(y|x_1, ..., x_p) \approx \frac{p(y)\Pi_i p(x_i|y)}{p(x_1, ..., x_p)}$$

and a new label is predicted as

$$y = \arg\max_y p(y|x_1, ..., x_p)$$

with $p(x_i|y)$ estimated from the data. Here we used a Gaussian pdf for $p(x_i|y)$.

A **Random Forest** (RF) classifier generates many decision trees built from a sample drawn with replacement from the training set. The predicted y is then taken as the average of the probabilistic prediction of all trees.

An **Adaboost** (adaboost) classifier uses a weighted sum of predictions from other classifiers to decide the predicted y. Each point is given a weight that is adapted depending on the difficulty to predict its label correctly. We use **logistic regression** as sub-classifier. Logistic regression models the probability of having label y depending on features x as

$$p(y|x) = \frac{1}{1 + e^{-w^\top x}}.$$

Parameters w are estimated by minimizing the error function

$$E(w) = -\log p(y|w).$$

We choose to use the **balanced classification rate** (BCR) as the performance measure. The balanced classification rate, or balanced accuracy, is computed as $0.5(\frac{TP}{TP+FN} + \frac{TN}{TN+FP})$ with TP, TN, FP, FN representing the number of true positives, true negatives, false positives and false negatives respectively. BCR allows taking into account a potential imbalance between classes. The reference value is then 0.5, which corresponds to assigning randomly the labels based on the classes' probabilities. If a dataset presents only one type of sample, BCR is taken as the proportion of correct predictions.

2.3.4 Results

Influence of the test dataset. As described in Table 2.2, the datasets used vary in different aspects such as the technology used, the number of samples or the proportion of ER+/ER- classes. Figure 2.11 shows for each dataset and each batch effect removal method the mean BCR obtained when averaging on the different classifiers and training sets. In order to stay concise, results were averaged on all number of selected features since similar results were obtained when considering different numbers of selected features.

We can see in Figure 2.11 that in some cases, such as *RatioA* or $rgCCA_{sq}$, some methods have worse BCR compared to the reference case without any batch effect removal. Datasets GSE5327_{96} and GSE32641_{887} have very bad results (BCR $\leq$ 0.5), probably due to the fact that their class frequencies

(100% of ER-) are very far from the training ones (majority of ER+). For the remainder of the experiments these two datasets are excluded from the testing set. Among the seven other test datasets, all have quite similar behavior except for GSE6532_{96} which leads to bad results for location-scale methods.

Detailed results for dataset GSE6532_{96} are shown in Figure 2.12, where it can be seen that applying a specific method can significantly decrease or increase the performance. Any location-scale method worsens the results while FA and SVD improve it significantly. The bad behavior of location-scale methods can probably be explained by the two clusters initially present in the dataset and that cannot be removed by a simple location-scale method (see Figure 2.7). Matrix factorizations methods, however, due to their unsupervised foundations (namely, line 1 of Algorithm 3 is unsupervised), capture trends within batches.

Influence of the classifier. The effects of the different batch effect removal methods and classifiers on the testing sets were compared in Figure 2.13. We can see in Figure 2.13 that all classifiers have similar behavior for the majority of methods. Some methods such as *RatioA*, SVA or $rgCCA_{sq}$ worsen the BCR while FA, SVD and $stICA_{0.25,0.5}$ lead to the best results, with a slightly higher mean and a smaller variance than $None$. Note that while using class information, SVA performances are among the worse, probably due to the fact that the method does not use the batch information.

Influence of the technology used. Since initially the datasets tend to cluster by the technology used (Affymetrix or Agilent), we investigated more carefully the influence of this factor in the training and testing sets in Figure 2.14. For this purpose two more classification models were trained, one on the 3 Agilent training datasets only and one on the 3 Affymetrix datasets. Testing set was also separated in two, depending on the technology used. Differences between both technologies are limited but matrix factorization methods seem to be less sensitive to the technology used.

Influence of the training set. Figure 2.15 shows the average performances depending on the training set used. Whereas the results depend strongly on the training test when no batch effect removal method is used, applying any method increases the stability of results. The best methods regarding stability are the same: FA, SVD, $stICA_{0.25,0.5}$. $ComBat$, Std and $Centering$ are slightly less stable due to $GSE6532_{96}$.

Influence of class frequencies in the training set. As seen previously, the models used have noticeable difficulties to make correct predictions for datasets with only ER- samples. In sample classification tasks, if the class labels, i.e., the phenotype to predict, are balanced across batches, then batch effects are less likely to adversely affect the outcome [145, 110]. In the extreme case of perfect confounding of batches with the phenotype, it is much more difficult

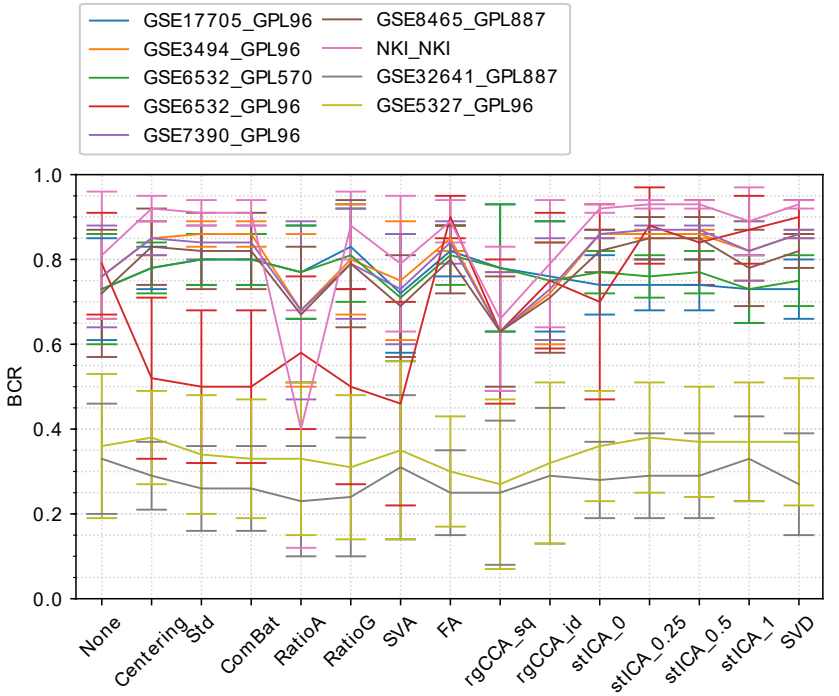


Figure 2.11: Influence of the batch effect removal methods on BCR for the different testing datasets. Mean on all training datasets, on all classifiers and on all numbers of selected features.

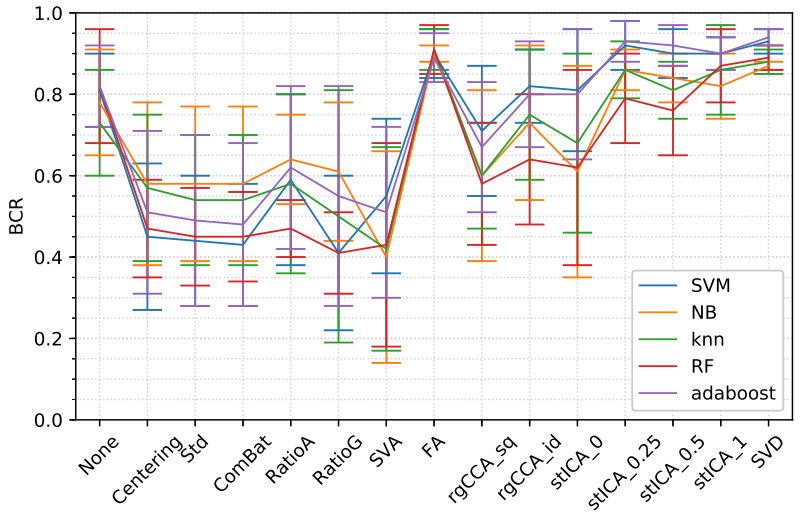


Figure 2.12: Influence of the batch effect removal methods and classifiers on BCR for $GSE6532_{96}$

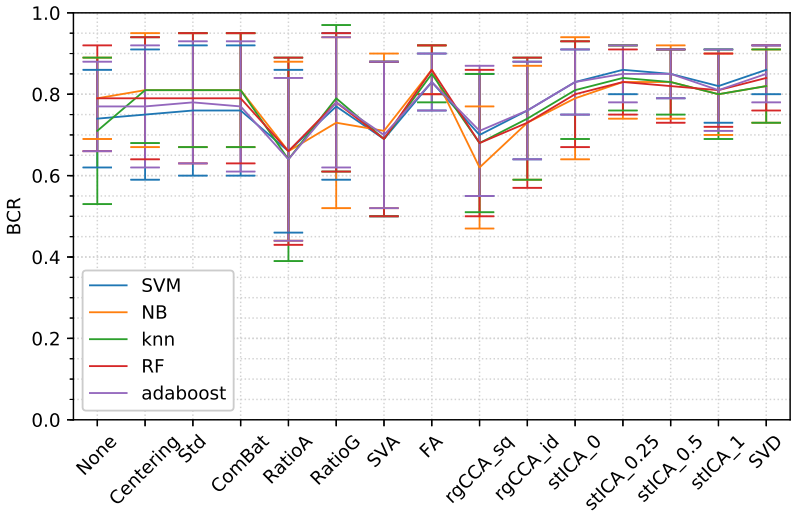


Figure 2.13: Influence of the batch effect removal methods and classifiers on BCR. Mean on test datasets and on numbers of selected features.

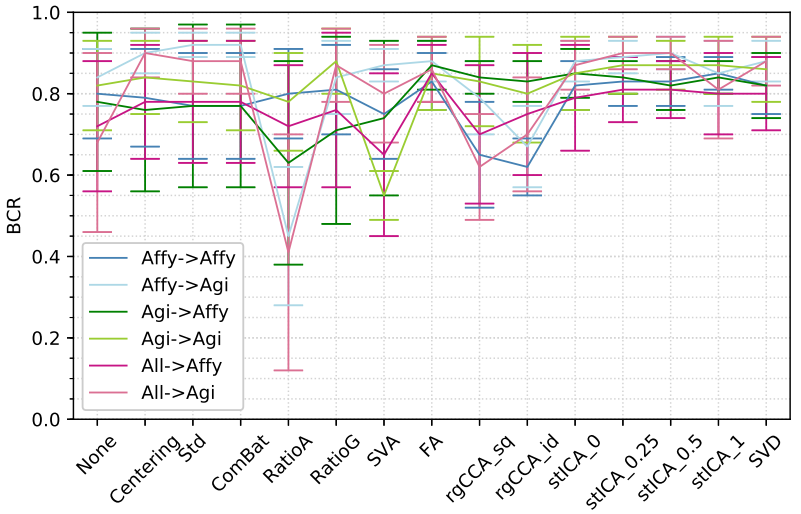


Figure 2.14: Influence of the batch effect removal methods and technology used on BCR. $X \rightarrow Y$ means that the model was trained on datasets using technology X, and tested on technology Y. 'All' corresponds to the means on the C_6^4 models precedently trained.

to be sure that the differentially expressed genes are linked to the batch or the phenotype. Such a case can occur when, for example, a laboratory first analyzes the disease samples and then matches them with the controls. When combining data from existing studies, the reality is often somewhere between the two extremes: group repartition of the phenotype to predict is often unbalanced, as for example, in the breast cancer batches we use in this section (see Table 2.2). In cases where the repartition of the phenotype of interest is highly unbalanced among batches, being able to separate batch effects from phenotype is more challenging [138]. Using datasets only preprocessed with either RMA or fMRA summarization, authors of [110] tried to predict estrogen receptor status in a breast cancer dataset when batch and status are perfectly confounded. Their results suggest that the algorithm tends to predict the batch more than the status, and that a careful feature selection can improve the results. In [109], a sanity check using random numbers is proposed. Replacing real data with normal random numbers shows that the F-statistic is inflated in the presence of an unbalanced repartition. In [130], the list of DEG obtained from two batches separately is compared to the DEG list obtained from the normalized merge of the two batches. When introducing unbalance in the repartition of group treatment among batches, most location-scale normalization methods they tested tend to detect less DEG. So if batch effects should be taken into account to avoid to predict batch instead of phenotype, we should also check that what is removed during the normalization step is actually only the batch effects and does not contain potential useful information about the phenotype to predict.

To investigate the effect of unbalanced repartition of classes in the training set, we trained our models on training datasets randomly subsampled to increase the proportion of ER- samples (described in Table 2.3). Performances obtained with those training datasets are shown in Figure 2.16. The two worse test datasets are the two with only ER+ samples. All datasets with a majority of ER+ samples have degraded BCR while performances of those with ER- only improved compared to Figure 2.11. The general behavior of the different methods is similar to the previous observations, except for FA which is now among the worse methods.

	Batch	Technology	# ER-	# ER+
Train	$GSE2034_{96}$	Affymetrix	77	19
	$GSE21653_{570}$	Affymetrix	113	28
	$GSE2990_{96}$	Affymetrix	21	85
	$GSE17040_{887}$	Agilent	11	44
	$GSE1992_{887}$	Agilent	6	23
	NKI2	Agilent	43	11

Table 2.3: Summary of batches subsampled to modify the class frequencies.

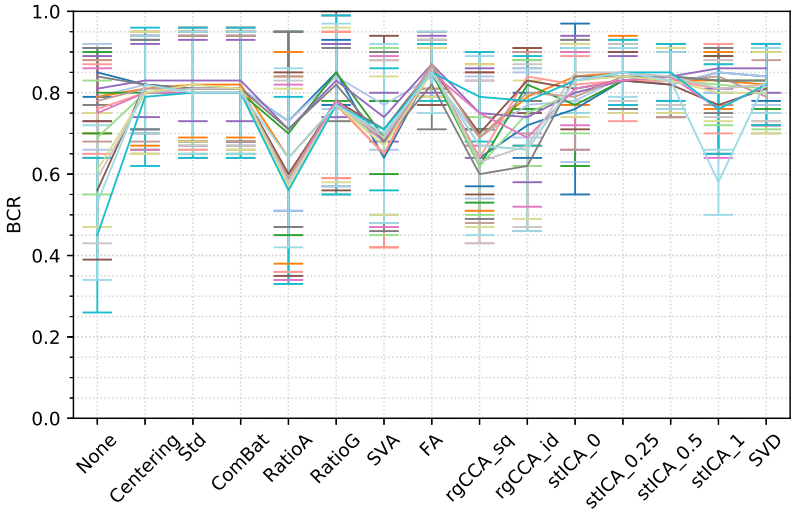


Figure 2.15: Influence of batch effect removal methods and training sets on BCR, averaged on all test datasets and all classifiers. Each curve corresponds to a different training set.

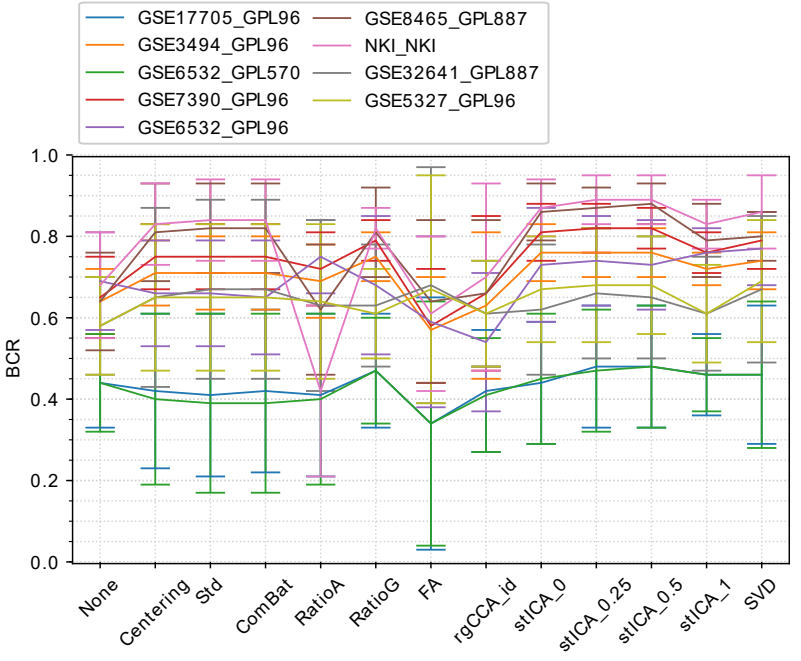


Figure 2.16: Influence of the batch effect removal methods on BCR for the different testing datasets when training set is modified to have a higher proportion of ER- samples.

2.3.5 Conclusion

In the context of merging gene expression datasets, we have investigated two types of batch effect removal approaches: location-scale and matrix factorization-based methods. We have compared the methods used as preprocessing tools for various basic classifiers. To comply with real-life scenario, we first applied the batch effect removal method, selected the features and trained the classifiers on a subset of datasets. After all the models were trained based on the training set, we applied on a separate test set the batch effect removal method and the previously computed classifiers.

The first observation is that in a specific case (dataset $GSE6532_{96}$), applying location-scale methods can worsen the classification performances. This may be due to the presence of a batch effect within the dataset. Matrix factorization-based methods significantly improve the performances.

In most cases we examined, applying a batch effect removal method did not increase significantly the performances compared to the $None$ case, some even nearly systematically decreased it ($RatioA$, SVA, and $rgCCA$). However, even those methods appear to decrease the influence of the training set used (see Figure 2.15), which is essential in the reproducibility of results. The best methods in our tests are FA, SVD, and $stICA_{0.25,0.5}$. FA being among the best methods is not surprising since it uses the class information in the training set, whereas the other methods do not use the class information in any way. In other words, FA is a classification-task-aware method, whereas the other batch effect removal methods are more general purpose. More unexpected is that SVD and some of the ICA-based methods, that unlike FA use only batch information, behave as well as FA (see Figure 2.13). Note that if we do not consider dataset GSE6532, location-scale methods $ComBat$, Std and $Centering$ behave as well as the previous ones.

Class frequency in the training set influences also the results, and extra caution should be taken in the presence of unbalanced repartition of the classes among the datasets, especially with supervised methods such as FA.

To conclude, we have shown that matrix factorization-based methods behave at least as well as more common location-scale methods, and even better when all the batch information is not specifically known. We only tested a few factorizations, but different approaches such as nonnegative matrix factorization or other versions of $rgCCA$ should be investigated. Such methods are often used in multi-omics data integration, but could also be adapted in the case of batch effect removal.

Chapter 3

Extracting a common subspace from multiple datasets: Grassmannian minimum enclosing ball

This chapter proposes a minimax center approach to the problem of extracting a common subspace from multiple datasets. This can be viewed as the combination of two fundamental tasks—subspace extraction and minimax center computation—that appear separately in multiple fields such as bioinformatics, signal processing or computer graphics; we briefly describe a few of such applications below.

Extracting common components in bioinformatics. It is increasingly common to have access to different biological datasets measuring the same phenomenon. For example, the same disease is measured for different sets of patients in various experimental conditions, and the goal is to detect differentially expressed genes related to this disease. To obtain more robust results, it is desirable to select the gene pattern linked to the disease that is present in all those datasets [129]. Another possible case is when different kinds of data are collected using different technologies for the same set of patients [9, 99]: proteomic, genomic or transcriptomic data for example can be available for the same set of samples.

Blind equalization of SIMO systems. In digital communication, an input signal is transmitted through a channel to a receiver. In wireless communication for example, the channel used is the electromagnetic field. However, the presence of hills, buildings or utility wires can scatter the wavefronts which

then take many paths to reach the receiver. This scattering effect can reduce the data speed and increase the numbers of errors. Using multiple receivers can help reduce those troubles: the initial common input signal is then estimated based on the multiple received signals. Such a problem is called blind equalization of Single-Input/Multiple-Output (SIMO) systems [158, 92].

Collision detection in computer graphics. In the real world, when two solid objects collide they do not penetrate each other. To prevent this from happening in interactive graphics applications, it is crucial to be able to detect such a collision quickly to avoid latency. A possible approach is to represent each object as a set of simple geometric shapes like boxes or spheres, the object being entirely contained in this set of shapes. If those simplified representations intersect, they are then refined as far as necessary. Using spheres has the advantages that their storage cost is low (only center and radius) and the collision checking can be very efficient [83, 67, 68]. Looking for the center can be interpreted as looking for a simplified representative of the set of points representing the object surface.

Facility location problem. In operational research, the facility location problem is a well-studied problem that attempts to locate a set of facilities such that a set of demands is satisfied while minimizing associated costs [104]. For example, the problem could be to build a warehouse to supply a set of customers such that the total distances from the warehouse to the customers is minimized. If this warehouse corresponds to a public or emergency service like a hospital, then all customers should be able to access it in a reasonable time and therefore the maximal distance should be minimized, leading to a 1-center problem. Such problems also arise when locating more than one facility, and with various way of defining the distances. In some cases the new facilities can be built anywhere while in other cases the possibles locations are restricted [113].

Clustering. A usual goal in unsupervised learning is to try to find common patterns across data points by clustering them. A possible approach is to check if those points can be partitioned in k subsets or clusters, such that all points within a cluster are at a distance of at most d from the center of the cluster [7]. The points can also be first mapped to a higher dimensional feature space using a kernel before computing distances [18]. The center of each cluster, i.e., the center of the corresponding minimum enclosing balls can then be used as a representative of all associated points to decrease computational complexity in another task like classification [32].

Support Vector Machines. Support Vector Machines (SVM) methods are prediction algorithms that have achieved very good performance for classifications tasks. SVM training is generally formulated as a quadratic programming problem for which naive optimization takes $O(n^3)$ time and $O(n^2)$ space for n

samples. When n is large it is then important to scale up the performance of the training phase. The authors of [155] showed that the quadratic programming problem could be reformulated as a minimum enclosing ball problem, allowing the use of existing fast approximation algorithms.

In this chapter, we develop a minimax approach to address the problem of extracting common information from multiple datasets. Section 3.1 gives the general context of the problem and its formulation as an optimization problem. Section 3.2 gives a brief overview of useful optimization tools. Section 3.3 presents the various algorithms proposed and Section 3.4 compares their performances.

3.1 Problem: context and formulation

We describe here the general context of the problem and how we formulate it. Methods to compute a common lower dimensional representation of different datasets are detailed in Section 3.1.1 while the methods looking for a representation minimizing the maximal distance are presented in Section 3.1.2. The choice of dissimilarity and its implications are discussed in Section 3.1.3.

Let X_1, ..., $X_m \in \mathbb{R}^{p \times n_i}$ represent different datasets. We assume that the datasets have a common part U_c, and that each X_i can be represented as

$$X_i = \begin{pmatrix} U_{ci} & U_{2i} \end{pmatrix} \begin{pmatrix} V_{ci}^\top \\ V_{2i}^\top \end{pmatrix}$$

with X_i of rank n_i, U_{ci} of rank k, U_{2i} of rank $n_i - k$ and V_{ci}, V_{2i} of ranks k and $n_i - k$ respectively. U_{ci} represents a possibly noisy version of the common U_c: in an ideal case, we have $U_c = U_{ci}$ for all i. Note that we have

$$X_i = \begin{pmatrix} U_{ci}M & U_{2i} \end{pmatrix} \begin{pmatrix} M^{-1}V_{ci}^\top \\ V_{2i}^\top \end{pmatrix}$$

for any matrix $M \in GL(k)$, the set of $k \times k$ invertible matrices. In this chapter, our goal is to recover U_c while keeping in mind three objectives:

- Each dataset can have information not present in other datasets: we can have $k < n_i$, that is U_c could be of low dimension. U_c should summarize efficiently the common information, and not capture noise or variations specific to a subset of the dataset.
- To represent all datasets, U_c should be close to all datasets.
- To avoid an hypothesis on the specific representation of U_c, we consider U_c and U_cM as equivalent.

The implications of these three objectives are detailed below.

Lower dimension. We want to find U_c representing well enough the original X_i's but of lower rank. That is, we want to find $U \in \mathbb{R}_*^{p\times k}$, the set of full rank matrices in $\mathbb{R}^{p\times k}$, such that $d(U, X_i)$ is small for all i, with d a function evaluating how well U represents X_i. One of the most used approach to summarize one dataset using a lower dimensional representation is Principal Component Analysis (PCA). PCA looks for a linear combination of the initial features maximizing its variance, in order to keep a maximum of the information present in the dataset. Solution of the PCA problem can be computed with Singular Value Decomposition (SVD). Many extensions to multiple datasets were derived from this approach, that look for linear combinations of the initial datasets to build new components with a maximal covariance or correlation. Such methods are described in Section 3.1.1.

Close to all datasets. A central question when using more than two datasets is the importance to give to those different datasets. Common approaches like CCA [64] are to give all datasets the same importance. However, what if the common component in the X_i's is present in a noisy version? We should be careful to not recover another signal which is less noisy but not present in all X_i's. For instance, if we are dealing with a set of datasets all very similar except one (for example, because measured using another technology), giving all the datasets the same importance can lead to components representing very well technical features specific to all the similar datasets but not present at all in the last one. To avoid this situation, and in order to take all X_i's into account we propose to minimize the maximal dissimilarity d between the common component $U_c \in \mathbb{R}^{p\times k}$ and all datasets $X_i \in \mathbb{R}^{p\times n_i}$:

$$U_c = \arg \min_{U\in\mathbb{R}^{p\times k}} \max_i d(U, X_i). \tag{3.1}$$

This formulation can be viewed as looking for the center of the smallest-radius sphere enclosing all X_i's, and can be linked to the minimum enclosing ball, 1-center problem or minimax optimization problem. This kind of optimization problem is well-studied, especially in Euclidean space, and a few of the existing methods are summarized in Section 3.1.2.

Independent on the specific representation. To deal with the degrees of freedom linked to M, many methods add constraints. Some impose orthogonality on the columns like SVD, statistical independence like ICA, nonnegativity like Nonnegative Matrix Factorization [86],... If $k < n_i$, such methods usually aim to find U such that a maximum of the variations present in the datasets is kept. However, if we have no information on the properties of U_c, the common components may be hidden by other variations in the datasets, stronger but not present in all the X_i's. To give all variations the same importance, and ensure that we do not depend on the specific representation of U_c, a solution is to work with $\mathcal{U}_c$ and $\mathcal{X}_i$, the subspaces generated by the columns of U_c and X_i.

That is, to solve Problem (3.1) such that $d(U, X_i) = d(\mathcal{U}, \mathcal{X}_i)$ is a dissimilarity measure between $\mathcal{U}$ and $\mathcal{X}_i$:

$$\mathcal{U}_c = \arg\min_{\mathcal{U}} \max_i d(\mathcal{U}, \mathcal{X}_i). \tag{3.2}$$

Two dissimilarity measures ensuring $d(U, X_i) = d(\mathcal{U}, \mathcal{X}_i)$ and their implications on Problem (3.2) are presented in Section 3.1.3.

A simple illustration. To illustrate the above-mentionned challenges, let us consider datasets generated from an orthonormal basis U_c of a common subspace of dimension 3. We generated the jth component of X_i as

$$X_i(:, j) = w_j \, \text{rotate}(U_c(:, j), \phi_{ij}). \tag{3.3}$$

The w_{ij} controls the importance given to component j in X_i, and the angle ϕ_{ij} in $[0,\ \pi/2]$ represents the noise added to the original component $U_c(:, j)$. A $\phi_{ij} = 0$ implies that $U_c(:, j)$ is present in X_i without noise while a $\phi_{ij} = \frac{\pi}{2}$ implies that $U_c(:, j)$ and $X_i(:, j)$ are orthogonal, i.e. $U_c(:, j)$ is not present at all in X_i. We take

$$\phi_{ij} = \frac{\pi}{2} \begin{cases} U_{[a_j\ b_j]} & \text{with probability } p_j \\ 1 & \text{with probability } 1 - p_j \end{cases}$$

with $U_{[a_j\ b_j]}$ denoting the uniform distribution on $[a_j\ b_j]$. Values for parameters p_j, w_j, a_j and b_j are taken as:

$p =$	[	1	0.8	0.5	]	presence in the various X_i's
$w =$	[	0.5	1	2	]	importance of the component
$a =$	[	0.25	0	0	]	noise lower bound
$b =$	[	0.45	0.1	0.1	]	noise upper bound.

So

- The first component of U_c is present in all X_i's, weakly and with high noise (U_1 in Figure 3.1),
- The second component of U_c is present in most X_i's (but not all), with small noise (U_2 in Figure 3.1),
- The last component of U_c is present in half of the X_i's, strongly and with small noise (U_3 in Figure 3.1).

Ideally, we want to recover U_1, the first component of U_c that is present in all X_i's. If we apply a PCA on $Y = [X_1, ..., X_m]$, the first principal component will recover the stronger variations in Y, which corresponds to the last component U_3. If we apply a PCA on $Y = [\check{X}_1, ..., \check{X}_m]$ with $\check{X}$ an orthonormal basis of X, we will recover the less noisy and most present signal, which is the second

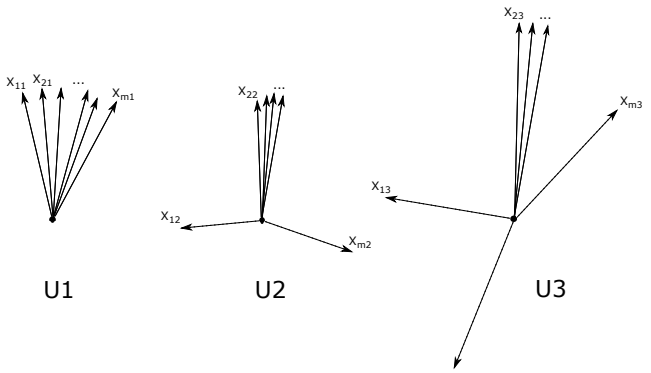


Figure 3.1: Illustration of the datasets generated by model (3.3). One X_i is generated as $[X_{i1}, X_{i2}, X_{i3}]$.

one U_2. To recover the first one, we need to minimize the maximal distance. This model is used in Section 3.3 to illustrate the proposed algorithms, and experimental results with a similar data model are reported in Section 3.4.1.

3.1.1 Extracting a representative of lower dimension

To extract a common part from various datasets X_1, ..., X_m, the most basic possibility is to simply concatenate them all into a larger dataset $Y = [X_1 \ \dots \ X_m]$ and apply standard methods such as PCA on Y. However more specific approaches exist that we detail partially here.

A method to factorize two datasets with a common factor is proposed in [9]. The authors assume that both datasets can be represented as

$$X_i = U_i D_i V^T \qquad i = 1, 2$$

with V common to both matrices, D_i diagonal matrices and each U_i orthogonal. The common dimension considered here are the columns, i.e. $n_1 = n_2$ while the p_i may be different. As $p_i \gg n$, such a V always exists and the constraints of orthogonality on the U_i has to be added to ensure uniqueness. A closed-form solution can be computed using the Generalized Singular Value Decomposition (GSVD).

An extension to more than two datasets was proposed in [117], but requires relaxing the orthogonality constraints on the U_i's. Such methods assume that the common dimension of the datasets, n, corresponds to the rank of the X_i's, an hypothesis that is not met in Problem (3.2).

When looking for common trends in two datasets, the best known method is probably Canonical Correlation Analysis (CCA) [64]. For each dataset, CCA

aims to find a linear combination of the initial features such that the correlation between those two combinations is maximized:

$$\begin{aligned} &\max_{t_1,t_2} \operatorname{corr}(X_1 t_1, X_2 t_2) \\ &\text{s.t.} \quad \operatorname{var}(X_i t_i) = 1 \qquad\qquad \forall\ i = 1, 2. \end{aligned}$$

An exact solution can be computed based on the covariance matrix: t_i is the leading eigenvector of $\Sigma_{X_i X_i}^{-1} \Sigma_{X_i X_j} \Sigma_{X_j X_j}^{-1} \Sigma_{X_j X_i}$ with $\Sigma_{X_i X_j} = \operatorname{cov}(X_i, X_j)$. In order to find more than one pair of correlated combinations of features, deflation is usually used: the same procedure (CCA) is repeated on the data from which the previous components were removed using the projections of X_1, $X_2 j$ on $X_1 t_1$ and $X_2 t_2$ respectively.

We consider here that

$$\operatorname{corr}(X, Y) = \frac{\operatorname{cov}(X, Y)}{\sqrt{\operatorname{var}(X)}\sqrt{\operatorname{var}(Y)}}.$$

If $X \in \mathbb{R}^{n \times p_X}$ and $Y \in \mathbb{R}^{n \times p_Y}$ are centered then $\operatorname{cov}(X, Y) = X^\top Y$ and $\operatorname{var}(X) = \operatorname{cov}(X, X)$.

Another well-known method is Partial Least Square regression (PLS) [169] that aims to find linear combinations of features for the two datasets such that the covariance between those two new representations is maximal:

$$\begin{aligned} &\max_{t_1,t_2} \operatorname{cov}(X_1 t_1, X_2 t_2) \\ &\text{s.t.} \quad \operatorname{var}(t_i) = 1 \qquad\qquad \forall\ i = 1, 2. \end{aligned}$$

As in CCA a closed-form solution exists (the optimal t_i is the leading eigenvector of $X_i^\top X_j X_j^\top X_i$), and deflation can be used to compute the next components.

Many extensions of CCA and PLS methods to more than two datasets have been proposed, see for example [99, 117, 146, 164]. Those extensions can usually be written as:

$$\begin{aligned} &\max_{t_i} \sum_{j \neq k} c_{jk}\, g(\rho(X_j t_j, X_k t_k)) \qquad (3.4) \\ &\text{s.t.} \quad \tau_i ||t_i||_2^2 + (1 - \tau_i) \operatorname{var}(X_i t_i) = 1 \qquad \forall\ i. \end{aligned}$$

where $\tau_i \in [0, 1]$, $c_{jk} = 1$ if X_i and X_j are connected and 0 otherwise, g is the absolute value or some real positive power function, and ρ is the covariance or correlation function.

Joint and Individual Variation Explained (JIVE) [94] is another method looking for decompositions of the $X_i \in \mathbb{R}^{p_i \times n}$ as a common part and individual components:

$$X_i \approx U_i S + A_i$$

with $U_i S \in \mathbb{R}^{p \times n}$ of rank r the common part and $A_i \in \mathbb{R}^{p_i \times n}$ of rank r_i the individual part. Matrices U_i, S and A_i are found by minimizing the approximation error $||X_i - U_i S - A_i||_F^2$ for all i.

Problem (3.4) can also be rewritten as looking for a unique central vector u correlating with all X_i's:

$$\max_{t_i, u} \sum_{j=1}^{m} c_j g(\rho(X_j t_j, u))$$

with $u = Yt$ and $Y = [X_1, \ldots, X_m]$ the concatenation of all the X_i's. An example linking both formulations is detailed in Example 3.1.1. Feature selection can also be enforced by adding the constraint $||t_i||_1 \leq s_j$, with s_j a positive constant determining the level of sparsity wanted. For some specific cases of ρ, g and τ_i a closed form solution can be computed based on the leading eigenvector of $\sum_i X_i X_i^\top$, but for others an iterative algorithm must be used. Table 3.1 lists some existing methods.

Example 3.1.1. Rewriting using a central vector. We consider the problem

$$\begin{aligned} &\max_{t_k} \sum_{i \neq j} \operatorname{cov}(X_i t_i, X_j t_j) \\ &\text{s.t.} \quad \sum_i t_i^\top t_i = 1 \end{aligned} \tag{3.5}$$

with centered X_i's, so $\operatorname{cov}(X_i t_i, X_j t_j) = t_i^\top X_i^\top X_j t_j$. This is equivalent to

$$\begin{aligned} &\max_t t^\top Y^\top Y t \\ &\text{s.t.} \quad \sum_i t_i^\top t_i = t^\top t = 1 \end{aligned}$$

with $Y = [X_1 \ldots X_m]$, $t = [t_1^\top \ldots t_m^\top]^\top$. To solve this optimization problem we can write the Lagrangian:

$$L(t, \lambda) = t^\top Y^\top Y - \lambda(t^\top t - 1).$$

Setting its derivative to 0 gives:

$$\begin{aligned} Y^\top Y t &= (\lambda - 1) t \\ t^\top t &= 1 \end{aligned}$$

which implies that t is the leading eigenvector of $Y^\top Y$, which is equivalent to the right singular vector of Y. If we are looking for a central component u, the problem can be rewritten as

$$\begin{aligned} \max_{t_k} & \sum_i t_i^\top X_i^\top \underbrace{Yt}_{u} \\ \text{s.t.} & \sum_i t_i^\top t_i = 1. \end{aligned}$$

Let $UDV^\top$ be an SVD of Y, the optimal solution to (3.5) is then t, the first column of V. We have

$$u = Yt = UDV^\top t = U[:, 1]D[1, 1]$$

that is the central component u is the scaled leading left singular vector of Y.

Observe that some formulations in Table 3.1 can be equivalent depending on the assumptions on the data since

$$\operatorname{corr}(X, Y) = \frac{\operatorname{cov}(X, Y)}{\sqrt{\operatorname{var}(X)}\sqrt{\operatorname{var}(Y)}}.$$

If $\operatorname{var}(X) = \operatorname{var}(Y) = 1$, we have $\operatorname{corr}(X, Y) = \operatorname{cov}(X, Y)$. If $X \in \mathbb{R}^{n \times p_X}$ and $Y \in \mathbb{R}^{n \times p_Y}$ are centered then $\operatorname{cov}(X, Y) = X^\top Y$ and $\operatorname{cov}(XA, YB) = A^\top \operatorname{cov}(X, Y)B = A^\top X^\top Y B$.

Method	Objective function	Constraint	Deflation constraint	Solution	Ref.
1D					
PCA	$\max_t \text{var}(Xt)$	$\|\|t\|\| = 1$	$Xt^{(t+1)} \perp Xt^{(t)}$	Xt = eigen vector of $X^\top X$	[24, Sec. 12.1]
2D					
GSVD	$X_i = U_i D_i V^\top$	$\|\|u_i\|\| = 1$, $\|\|v\|\| = 1$	$u_i^{(t+1)} \perp u_i^{(t)}$	generalized SVD	[9]
CCA	$\max_{t_i} \text{corr}(X_1 t_1, X_2 t_2)$	$\text{var}(X_i t_i) = \|\|X_i t_i\|\| = 1$	$X_i t_i^{(t+1)} \perp X_i t_i^{(t)}$	t_i = eigenvector of $\Sigma_{X_i X_i}^{-1} \Sigma_{X_i X_j} \Sigma_{X_j X_j}^{-1} \Sigma_{X_j X_i}$	[64]
PLS	$\max_{t_i} \text{cov}(X_1 t_1, X_2 t_2)$	$\|\|t_i\|\| = 1$	$X_i t_i^{(t+1)} \perp X_i t_i^{(t)}$	t_i = eigen vector of $X_i^\top X_j X_j^\top X_i$	[169]
>2D					
HOGSVD	$X_i = U_i D_i V^\top$	$\text{diag}(U_i^\top U_i) = 1$, $\|\|v\|\| = 1$	all at once	analytical, see paper	[117]
MCIA/MCoA	$\max_{t_i,u} \sum_k \text{cov}^2(X_k t_k, u)$	$\|\|t_i\|\| = \|\|u\|\| = 1$	$t_i^{(t+1)} \perp t_i^{(t)}$	u = eigen vector of $\sum_i X_i X_i^\top$, $t_i = X_i^\top u$	[99, 35]
CPCA	$\max_{t_i} \sum_k \text{cov}^2(X_k t_k, u)$	$\|\|t_i\|\| = \|\|u\|\| = 1$	$u^{(t+1)} \perp u^{(t)}$	u = eigen vector of $\sum_i X_i X_i^\top$, $t_i = X_i^\top u$	[55] [45]
rCPCA	$\max_{t_i} \sum_k \text{cov}^m(X_k t_k, u), m \geq 1$	$\text{var}(u) = 1$, $\tau_i \|\|t_i\|\|^2 + (1-\tau_i)\text{var}(X_i t_i) = 1$	to choose	Gradient alg.	[148]
gCCA	$\max_{t_i} \sum_{j \neq k} c_{jk} g(\text{corr}(X_j t_j, X_k t_k))$ where g is the identity, the absolute value or the square function	$\text{var}(X_i t_i) = 1$	Not given	Solved iteratively with Lagrangian.	[146]
rgCCA	$\max_{t_i} \sum_{j \neq k} c_{jk} g(\text{cov}(X_j t_j, X_k t_k))$	$\tau_i \|\|t_i\|\|^2 + (1-\tau_i)\text{var}(X_i t_i) = 1$	Not given	Solved iteratively with Lagrangian.	[146]
+Var. sel.		$+\|\|a_j\|\|_1 \leq s_j \quad \forall j$, s_j the amount of sparsity			[147]
MINT	$\max_{t_x,t_y} \sum_k n_k \text{cov}(X_k t_x, Y_k t_y) + \lambda \|\|t_x\|\|_1$	$\|\|t_x\|\|_2 = \|\|t_y\|\|_2 = 1$, n_k nb of samples, Y dummy variables to predict	$u^{(t+1)} \perp u^{(t)}$, $u = Xt$	Solved iteratively	[129]
intNMF	$\min_{W,H_i} \sum_i \theta_i \|\|X_i - W H_i\|\|_2$	W, H_i nonnegative, weight θ_i	all at once	nonnegative ALS/block coordinate descent	[33]
JIVE	$\min_{U_i,S,A_i} \sum_i \|\|X_i - U_i S - A_i\|\|_F^2$	$\text{rank}(U_i s) = r$, $\text{rank}(A_i) = r_i$, $S A_i^\top = 0$	all at once	ALS via SVD's	[94]

Table 3.1: Various algorithms extracting representatives of lower dimension.

3.1.2 Minimum enclosing ball problem: existing approaches

The minimum enclosing ball problem, or 1-center problem, or minimax location problem, is a well-studied problem in Euclidean space reported to date back to 1857 in the plane [144]. The objective is to find a ball $B(U, R)$ with center U and minimal radius R covering a set $S = \{X_1, \ldots, X_m\}$ of m points in $\mathbb{R}^p$:

$$R = \min_U \max d(U, X_i)$$

with $d(U, X_i) = (U - X_i)^\top (U - X_i)$ representing the Euclidean distance. The initial problem considered points in $\mathbb{R}^2$ or $\mathbb{R}^3$ only, but many versions have been studied including considering a higher p, weighted points, various distances, points lying on another space and balls instead of points. The minimum enclosing ball problem found applications in various fields like facility location [104, 113], computer graphics [83, 67] and machine learning related tasks such as clustering and classification [56, 107, 27, 14]. This problem is convex but nonsmooth: the exact solution can be computed using a combinatorial approach, but not analytically. Since these applications can require dealing with a large number of points (m) within a high dimensional space (p), many recent papers concentrate on improving computational efficiency [81, 80, 49, 31].

In our case each point is associated with a dataset and we can reasonably assume that m is quite low. We choose then to concentrate on the different general approaches to tackle the problem and not on the strategies to deal with large m. Some adaptations to the Noneuclidean case exist [11, 106], but none deals specifically with the subspace case. We present here the main ideas of the methods existing in $\mathbb{R}^p$.

In Euclidean space, the problem is equivalent to

$$\begin{aligned} &\min \tau \\ &\text{s.t.} \quad (U - X_i)^\top (U - X_i) - \tau \leq 0 \quad \forall i. \end{aligned}$$

The associated KKT conditions (see Section 3.2.3.1 for more details) gives:

$$\begin{aligned} \sum_i \gamma_i &= 1 && \text{(Stationarity)} \\ \sum_i \gamma_i X_i &= U && \text{(Stationarity)} \\ \gamma_i &\geq 0 && \text{(Dual feasibility)} \\ (U - X_i)^\top (U - X_i) - \tau &\leq 0 && \text{(Primal feasibility)} \\ \gamma_i \left((U - X_i)^\top (U - X_i) - \tau \right) &= 0 && \text{(Complementary slackness).} \end{aligned}$$

Combining stationarity and dual feasibility conditions implies that U should be a convex combination of the X_i's. Depending on which conditions are enforced at each iteration, the combinatorial approaches can be classified in three families [58]: primal, dual or primal-dual approaches.

3.1.2.1 Primal approaches

Primal methods start with a wider estimation of $B(U, R)$ and at each step decrease R. They are called primal approaches as $d(U, X_i) \leq R$, the primal feasibility, is always satisfied. Computation of γ_i's such that complementary slackness is satisfied is possible, however the dual feasibility implying that U is a convex combination of the points holds only at optimality.

In $\mathbb{R}^2$, [36] proposes a finite time approach. This approach is in fact a special case of Algorithm 5, which is a feasible direction method proposed in [73].

Algorithm 5 Primal approach to compute the minimax center in $\mathbb{R}^2$.

Require: Points X_1, ...,X_m, initial iterate $U^{(0)}$, tolerance δ
1: $t \leftarrow 0$
2: $I_f \leftarrow \left\{f | d(U^{(t)}, X_f) \geq (1-\delta) \max_i d(U^{(t)}, X_i)\right\}$
3: $\xi^{(t)} \leftarrow$ feasibleDirection(I_f)
4: **while** $\xi^{(t)}$ exists **do**
5: $\quad t \leftarrow t + 1$
6: $\quad U^{(t)} \leftarrow$ move in direction ξ until a new X_i enters I_f
7: $\quad I_f \leftarrow \left\{f | d(U^{(t)}, X_f) \geq (1-\delta) \max_i d(U^{(t)}, X_i)\right\}$
8: $\quad \xi^{(t)} \leftarrow$ feasibleDirection(I_f)
9: **end while**

The set of feasible directions corresponds to the cone defined by U and the furthest points X_f, $f \in I_f$. If the chosen direction is the cone bisector, then [36] showed that it is equivalent to the finite algorithm proposed in [58]. This algorithm was initially proposed to tackle the $\mathbb{R}^2$ problem, in which case the cone bisector is easy to find: I_f can be reduced to the two points that subtend the widest angle at U_i, see Figure 3.2. The approach is also valid for $p > 2$.

Another feasible direction method adapted to higher dimensions proposed in [49]is described in Algorithm 6. The minimax center is computed using the orthogonal projection of $U^{(t)}$ on aff(F), the affine hull of F defined as

$$\text{aff}(F) = \left\{ \sum_i \gamma_i X_i | \sum_i \gamma_i = 1, X_i \in F \right\}. \tag{3.6}$$

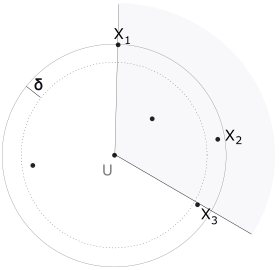


Figure 3.2: Cone of feasible directions, $I_f = \{X_1, X_2, X_3\}$

Algorithm 6 Primal approach to compute the minimax center in $\mathbb{R}^p$ (p large).

Require: Points X_1, ...,X_m
$t \leftarrow 0$
$U^{(t)} \leftarrow X_i$ for any $i \in [1, ..., m]$
$F \leftarrow \left\{X_f | d(U^{(t)}, X_f) = \max_i d(U^{(t)}, X_i)\right\}$
while $U^{(t)} \notin \text{conv}(F)$ **do**
    if $U^{(t)} \in \text{aff}(F)$ **then**
        $F \leftarrow F \backslash X_i$ for i such that $\gamma_i < 0$ in Equation (3.6)
    end if
    $t \leftarrow t + 1$
    $U^{(t)} \leftarrow$ move towards minimax center of F until a new X_i enters F
    $I_f \leftarrow \left\{f | d(U^{(t)}, X_f) \geq (1 - \delta) \max_i d(U^{(t)}, X_i)\right\}$
end while

3.1.2.2 Dual approaches

Dual methods start with a smaller estimation of $B(U, R)$ passing through a few points and at each step increase R until all points are covered. The set of points used to compute $B(U, R)$ is updated using the fact that the solution is determined by at most $p + 1$ affinely independent points. This corresponds to assigning $\gamma_i \neq 0$ to points defining the ball, and $\gamma_i = 0$ to the others: complementary slackness is satisfied. Stationarity and dual feasibility are satisfied as U is built as a convex combination of the points. The primal feasibility $d(U, X_i) \leq R$ however holds only at optimality. An approach of this type in $\mathbb{R}^2$ is proposed in [47] or [41, 31] in $\mathbb{R}^p$ and described in Algorithm 7, with

$$M(S): \quad \min_{U \in \mathbb{R}^p, z \in \mathbb{R}} z$$
$$\text{s.t.} \quad z \geq d(X_i - U) \quad \forall\ X_i \in S.$$

and the selection of X_{out} is made cleverly (see [41] for more details).

Algorithm 7 Dual approach to compute the minimax center in $\mathbb{R}^p$.

Require: Points X_1, ...,X_m
$t \leftarrow 0$
$S^{(t)} \leftarrow$ any two points X_i, X_j
$(U^{(t)}, R^{(t)}) \leftarrow$ solve $M(S^{(t)})$
while exists X_i not in $B(U^{(t)}, R^{(t)})$ **do**
    $t \leftarrow t + 1$
    $X_{in} \leftarrow X_i \notin B(U^{(t)}, R^{(t)})$
    if $X_{in} \in \text{aff}(S^{(t)})$ **then**
        $X_{out} \leftarrow$ less interesting point in $S^{(t)}$
        $S^{(t)} \leftarrow S^{(t-1)} \backslash X_{out}$
    end if
    $S^{(t)} \leftarrow S^{(t)} \cup X_{in}$
    $(U^{(t)}, R^{(t)}) \leftarrow$ solve $M(S^{(t)})$
    if $S^{(t)}$ not affinely independent **then**
        $X_{out} \leftarrow$ less interesting point in $S^{(t)}$
        $S^{(t)} \leftarrow S^{(t)} \backslash X_{out}$
    end if
end while

3.1.2.3 Dual-primal approaches

For dual-primal methods, primal feasibility ($d(U, X_i) \leq R$) and dual feasibility (U is a convex combination of the points) hold at each iteration, while computation of γ_i such that complementary slackness is satisfied is possible only at optimality. Such an algorithm, proposed in [84], is described in Algorithm 8.

Algorithm 8 Dual-primal approach to compute the minimax center in $\mathbb{R}^p$.

Require: Points X_1, ..., X_m, stopping criterion ϵ
$t \leftarrow 0$
$\gamma_i^{(t)} \leftarrow 1/m$ for all i
$U^{(t)} \leftarrow \sum_i \gamma_i^{(t)} X_i$
$\sigma \leftarrow \sum_i \gamma_i^{(t)} d^2(U^{(t)}, X_i)$
$\tau \leftarrow \max_i d^2(U^{(t)}, X_i)$
while $\tau - \sigma > \epsilon$ **do**
    $t \leftarrow t + 1$
    $\gamma_i^{(t)} \leftarrow \gamma_i^{(t)} d_i^2(U^{(t)}, X_i)$ for all i
    $\gamma^{(t)} \leftarrow \gamma^{(t)} / ||\gamma^{(t)}||_1$
    $U^{(t)} \leftarrow \sum_i \gamma_i^{(t)} X_i$
    $\sigma \leftarrow \sum_i \gamma_i^{(t)} d^2(U^{(t)}, X_i)$
    $\tau \leftarrow \max_i d^2(U^{(t)}, X_i)$
end while

3.1.2.4 Core-sets approximations

Finding the optimal solution can be computationally demanding since complexity of these and related algorithms grows with the number of dimension n and the number of points m. In many applications, approximate solutions to the problem are acceptable so many algorithms have been proposed to compute an approximate solution.

A very simple and efficient method proposed in [13], is given in Algorithm 9. Given a set of points $S \subset \mathbb{R}^p$ and value $\epsilon > 0$, an ϵ-core-set $S_\epsilon \subset S$ has the property that the smallest ball containing S_ϵ is an ϵ-approximation of the smallest ball containing S. That is, if the smallest ball containing S_ϵ is expanded by $1+\epsilon$, then it contains S.

Algorithm 9 Core-set approach to compute the minimax center in $\mathbb{R}^p$.

Require: Points X_1, ...,X_m, stopping criterion ϵ
1: $t \leftarrow 0$
2: $U^{(t)} \leftarrow X_i$
3: **while** $t \leq 1/\epsilon^2$ **do**
4: $\quad t \leftarrow t+1$
5: $\quad X_f \leftarrow$ a point at maximal distance from $U^{(t)}$
6: $\quad U^{(t+1)} \leftarrow U^{(t)} + (X_p - U^{(t)})\frac{1}{t+1}$.
7: **end while**

This algorithm was proved to give an $(1+\epsilon)$ approximation of the smallest enclosing ball after $\lceil \frac{1}{\epsilon^2} \rceil$ iterations.

3.1.3 Problem formulation

Let $X_i \in \mathbb{R}^{p \times n_i}$ be a matrix of p variables times n_i samples, for $i = 1, \ ..., m$. Our goal is to find

$$\mathcal{U}_c = \arg \min_{U \in \mathbb{R}_*^{p \times k}} \max_i d(\mathcal{U}, \mathcal{X}_i), \tag{3.7}$$

with $\mathcal{U}$ denoting the space spanned by the columns of U. That is, to find a subspace $\mathcal{U}$ of dimension k representative of all the subspaces $\mathcal{X}_i$, where $\mathcal{X}_i$ is the subspace generated by the columns of X_i. In other words we are looking for a $U \in \mathbb{R}^{p \times k}$ of full rank minimizing $d(U, X_i)$ for all i, where $d(U, X) = d(\mathcal{U}, \mathcal{X})$ is a dissimilarity measure between the range of U and the range of X.

The main difference with methods described in Section 3.1.1 is the minimization of the maximal dissimilarity and not the minimization of a (weighted) sum of all dissimilarities. The problem is also not an instance of the Euclidean minimum enclosing ball problem described in Section 3.1.2 since we are now working with subspaces $\mathcal{X}_i$ and not the X_i themselves. More precisely, instead of working with points in $\mathbb{R}^{p \times k}$, we consider points on the Grassmann manifolds $\mathcal{G}(n_i, p)$. The Grassmann manifold $\mathcal{G}(r, q)$ represents the space which parametrizes all r-dimensional linear subspaces in $\mathbb{R}^q$. Dealing with subspaces

of possibly different dimensions n_i, requires defining an appropriate dissimilarity $d(U, X_i)$.

3.1.3.1 Dissimilarity measures

Different dissimilarities can be used to quantify $d(U, X)$ and we detail some of them below. For $k = 1$, a possible choice to evaluate if a vector $u \in \mathbb{R}^p$ is close to $\mathcal{X}$ is the angle between u and its orthogonal projection onto $\mathcal{X}$, see Figure 3.3. A vector u is close to the subspace $\mathcal{X}$ if the (positive) angle between them is small. If we define ϕ as the angle between u and $\mathcal{X}$ (in $[0, \frac{\pi}{2}]$), we have

$$\left\langle u, \check{X}\check{X}^\top u \right\rangle = \cos\phi ||u||\ ||\check{X}\check{X}^\top u|| = \cos\phi ||u||\ \cos\phi ||u|| = \cos^2\phi$$

with $||u|| = 1$ and where the columns of $\check{X}$ form an orthonormal basis of $\mathcal{X}$.

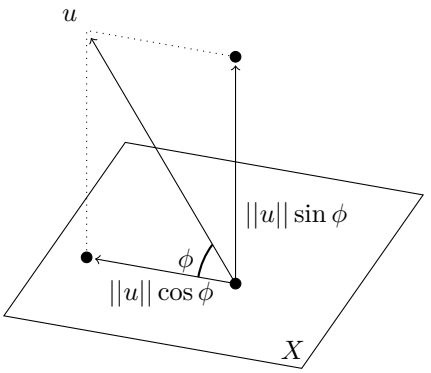


Figure 3.3: Principal angles: illustration in $\mathbb{R}^3$ for $\mathcal{U}$ of dimension 1 and $\mathcal{X}$ of dimension 2.

The term $u^\top \check{X}\check{X}^\top u$ can be seen as a similarity measure quantifying how close u is to $\mathcal{X}$, with a value of 1 when u is in the subspace $\mathcal{X}$, and 0 when they are orthogonal. We can then define a dissimilarity:

$$d(u, X) = \sqrt{1 - u^\top \check{X}\check{X}^\top u} = \sin\phi$$

with $d(u, X) = 0$ if and only if $u \in \mathcal{X}$.

This can be extended to a more general $U \in \mathbb{R}^{p\times k}$ with $p \geq n \geq k \geq 1$ (with n the dimension of $\mathcal{X}$) by summing the dissimilarities obtained for each element of an orthonormal basis of $\mathcal{U}$. If the columns of $\check{U}$ generate such an

orthonormal basis, we have:

$$d_{\sin\phi}(U,X) = \sqrt{\sum_{l=1}^{k} 1 - \check{U}_{:l}^{\top}\check{X}\check{X}^{\top}\check{U}_{:l}} = \sqrt{k - \mathrm{Tr}\left(\check{U}^{\top}\check{X}\check{X}^{\top}\check{U}\right)}$$
$$= \sqrt{\sum_{l=1}^{k} \sin^2\phi_l(U,X)}$$

with $\cos\phi_l(U,X)$ the singular values of $\check{U}^{\top}\check{X}$. Note that this quantity does not depend on the $\check{U}$ or $\check{X}$ chosen.

The ϕ_l's are also known as the principal or canonical angles between two subspaces, and can be recursively defined as [53, Section 2.4.3]:

$$\cos(\phi_l) = \max_{u\in\mathcal{U}} \max_{x\in\mathcal{X}} u^{\top}x = u_l^{\top}x_l$$

subject to

$$\begin{aligned} ||u|| = ||x|| &= 1 \\ u^{\top}u_i &= 0 \qquad\qquad i = 1:l-1 \\ v^{\top}v_i &= 0 \qquad\qquad i = 1:l-1. \end{aligned}$$

Another possible definition for $d(U,X)$ is to consider directly the principal angles ϕ_l between both subspaces:

$$d_{\phi}(U,X) = \sqrt{\sum_{l=1}^{k} \phi_l^2(U,X)}.$$

These two dissimilarities are not distances as $d_{\sin\phi}$ and d_{ϕ} are such that $\mathcal{U} \subsetneq \mathcal{X} \Rightarrow d(U,X) = 0$. They are distances only when $\dim(\mathcal{U}) = \dim(\mathcal{X})$.

To have distances we could take [46]:

$$d(U,X) = \frac{1}{\sqrt{2}}||\check{U}\check{U}^{\top} - \check{X}\check{X}^{\top}||_F = \sqrt{\frac{k+n}{2} - \mathrm{Tr}(\check{U}^{\top}\check{X}\check{X}^{\top}\check{U})}$$
$$= \sqrt{\frac{n-k}{2} + \sum_{l=1}^{k} \sin^2\phi_l(U,X)}.$$

Another possible distance is the norm of the difference between $\check{X}$ and its projection onto the common subspace $\mathcal{U}$ (termed *chordal metric* in [172, Table

3]):

$$d(U,X) = ||(I - \check{U}\check{U}^\top)\check{X}||_F = \sqrt{n - \operatorname{Tr}\left(\check{U}^\top \check{X}\check{X}^\top \check{U}\right)}$$
$$= \sqrt{n - k + \sum_{l=1}^{k} \sin^2 \phi_l(U,X)}.$$

When using those two last dissimilarities in $\min_U \max_i d(U, X_i)$, they will give more importance to datasets X_i with a higher n_i, so we will concentrate on d_ϕ and $d_{\sin\phi}$. Other dissimilarity measures are possible, see for example [172].

Dissimilarity $d_{\sin\phi}$ is easily linked to methods described in Section 3.1.1. If X is centered, then

$$\arg\min_u \max_i d_{\sin\phi}(X_i, u) = \arg\min_u \max_i d^2_{\sin\phi}(X_i, u)$$
$$= \arg\max_u \min_i \operatorname{cov}(X_i^\top u).$$

The expression for $d_{\sin\phi}$ using the trace of a matrix makes it easily differentiable. On the other hand, dissimilarity d_ϕ reduces to the canonical Grassmann distance when U and X have the same dimension, which facilitates the use of some manifolds tools.

Both $d_{\sin\phi}$ and d_ϕ have the interesting property that when computing dissimilarity between two subspaces $\mathcal{U}$ and $\mathcal{Y}$ of dimensions n_u and n_y respectively, with $n_u < n_y$, the dissimilarity between $\mathcal{U}$ and $\mathcal{Y}$ is equal to the dissimilarity between $\mathcal{U}$ and $\mathcal{X}$, the subspace of $\mathcal{Y}$ of dimension n_u closest to $\mathcal{U}$. This property is detailed in Proposition 3.1.1.

Proposition 3.1.1. *Let $\mathcal{U} \in \mathcal{G}(k,p)$ and $\mathcal{X} \in \mathcal{G}(n,p)$ where $n \geq k$, with $\check{X}$ and $\check{U}$ orthonormal basis of $\mathcal{X}$ and $\mathcal{U}$. Let $A_1 D_1 B_1^\top$ be a thin SVD of $\check{U}^\top \check{X}$, and $\phi_i = \arccos D_1(i,i)$ the ith principal angle between $\check{X}$ and $\check{U}$. Then with $d_\phi(\mathcal{U},\mathcal{X}) = \sum_i \phi_i^2$, we have*

$$\min_{\substack{\mathcal{Y}\in\mathcal{G}(k,p)\\ \mathcal{Y}\subset\mathcal{X}}} d_\phi(\mathcal{Y},\mathcal{U}) = d_\phi(\mathcal{X},\mathcal{U}) = d_\phi(\operatorname{Col}(\check{X}B_1),\mathcal{U}).$$

The equalities hold also for $d_{\sin\phi}$.

Proof. A proof of the first equality may be easily adapted from [172, Lemma 11].

Since $A_1 D_1 B_1^\top = \check{U}^\top \check{X}$ is a thin SVD, so is $A_1 D_1 I = \check{U}^\top \check{X} B_1$. Hence the singular values of $\check{U}^\top \check{X} B_1$ are those of $\check{U}^\top \check{X}$, the principal angles are the same, and the second equality follows.

□

Since $\phi \geq \sin\phi$, dissimilarity d_ϕ is an upper bound on $d_{\sin\phi}$.

Note that singular vectors of the concatenation of all X_i's can be interpreted as a mean subspace. If we consider the l_2 norm instead of the l_∞ norm with dissimilarity $d_{\sin\phi}$ and $k = 1$, we have

$$\begin{aligned}\arg\min_{u^\top u=1} \sum_{i=1}^{m} d^2_{\sin\phi}(u, X_i) &= \arg\min_{u^\top u=1} \sum_{i=1}^{m} 1 - \mathrm{Tr}(u^\top X_i X_i^\top u) \\ &= \arg\min_{u^\top u=1} m - u^\top (\sum_{i=1}^{m} X_i X_i^\top) u \\ &= \arg\max_{u^\top u=1} u^\top Y Y^\top u\end{aligned}$$

with $Y = [X_1, \ldots, X_m]$. This problem is the same as Example 3.1.1 and so has the leading singular vector of Y as solution. Computing a U of higher dimension such that $U^\top U = I_k$ comes to take the k leading singular vectors [45].

3.1.3.2 Final formulation

For $X_i \in \mathbb{R}^{p\times n_i}$, $i = 1, \ ..., m$, our goal is to find

$$\mathcal{U}_c = \arg\min_{U\in\mathbb{R}_*^{p\times k}} \max_i d(\mathcal{U}, \mathcal{X}_i), \tag{3.8}$$

with $\mathcal{U}$ the subspace generated by the columns of U for

$$d(\mathcal{U}, \mathcal{X}) = d_{\sin\phi}(X, U) = \sqrt{\sum_{l=1}^{\min(k,n)} \sin^2(\phi_l)}$$

or

$$d(\mathcal{U}, \mathcal{X}) = d_{\phi}(X, U) = \sqrt{\sum_{l=1}^{\min(k,n)} \phi_l^2}.$$

with $n = \dim(\mathcal{X})$. Recall that $\mathcal{U}$ and $\mathcal{X}_i$ denote the subspace spanned by the columns of U and X_i, respectively, and that the principal angles ϕ_l between $\mathcal{U}$ and $\mathcal{X}$ are the arccosines of the singular values of $\check{X}^\top \check{U}$.

This problem is nonconvex. A simple example can be given considering two subspaces of dimension 1, X_1 and X_2, represented by straight lines in the plane on Figure 3.4. The optimal solution is represented by U_1, however U_2, the subspace orthogonal to U_1, is also a local minimum.

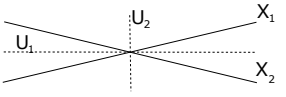


Figure 3.4: Example of local minimum for $\min_U \max_i d(U, X_i)$ when $d(U, X)$ is based on the principal angles.

3.2 Optimization tools

Problem (3.8) is nonconvex, nondifferentiable and with constraints. In this section we describe briefly the optimization tools that we will use to handle those difficulties in the algorithms proposed in Section 3.3. We start with a few generalities on optimization problems in Section 3.2.1, then describe the consequences of nonsmoothness of the objective function in Section 3.2.2, of adding constraints in Section 3.2.3 and of nonconvexity of the objective function in Section 3.2.4. The goal of this section, mainly based on [23, 26, 108], is to give a general intuition of the various methods, more details can be found for example in [23, 26, 108].

3.2.1 Basics

We consider the following general problem:

$$\min_{x \in D_x} f(x) \tag{3.9}$$

with D_x a nonempty subset of $\mathbb{R}^p$ and $f : \mathbb{R}^p \to \mathbb{R}$. In some specific cases, the optimal solution $f(x^*)$ can be found analytically. In our case, we want to solve problem

$$\min_{\mathcal{U}} \max_i d^2(\mathcal{U}, \mathcal{X}_i)$$

such that $\mathcal{U}$ is a subspace of dimension k and d represents $d_{\sin\phi}$ or d_ϕ. Applied to Problem (3.8), we have the objective function $f(x) = \max_i d^2(x, \mathcal{X}_i)$ with $x = \mathcal{U}$ and D_x is the set of subspaces of dimension k in $\mathbb{R}^p$. As this problem can not be solved analytically, optimization algorithms are needed. Such algorithms usually aim to compute a sequence $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ such that $f(x^{(t)}) \to f(x^*)$ if $t \to \infty$. The algorithm is usually stopped when we are close enough to the solution, or when $f(x^{(t)}) - f(x^*) \leq \epsilon$ with ϵ small and positive. If $f(x^*) = f^*$ is unknown, other criteria can be used, often linked to the derivatives of f.

Many algorithms were proposed to solve the general Problem (3.9), depending on the properties of f and D_x. Most of them are based on two main ideas: iterative descent and/or approximation [23].

Iterative descent computes a sequence of iterates $x^{(t)}$ which are feasible ($x^{(t)} \in D_x$) and either decrease the objective function f or become closer to the optimal set:

$$\phi(x^{(t+1)}) < \phi(x^{(t)}) \quad \text{iff } x^{(t)} \text{ is not optimal}$$

with $\phi(x)$ being $f(x)$ or a distance to X^*, the set of optimal solutions.

The approximation approach consists in solving iteratively an approximation of the initial problem:

$$x^{(t+1)} = \arg \min_{x \in D_x^{(t)}} f^{(t)}(x).$$

The iterates are not always feasible as the approximation should be simpler to facilitate the computation. The approximation should then become closer to the original problem with t increasing.

Here we focus on line search methods that use the idea of iterative descent. In a vector space, the usual structure of line search is to define a starting point $x^{(0)} \in D_x$ and to iteratively compute the next iterates as

$$x^{(t+1)} = x^{(t)} + \alpha^{(t)} d^{(t)} \tag{3.10}$$

with $d^{(t)}$ a descent direction and $\alpha^{(t)}$ a step size.

Descent direction. The descent direction is generally based on the gradient, the simplest one being the gradient iteration:

$$x^{(t+1)} = x^{(t)} - \alpha^{(t)} \nabla f(x^{(t)}).$$

The choice of the step size $\alpha^{(t)}$ is discussed below. Numerous improvements of this method exist. For example, Newton's method scale the gradient to improve the convergence rate:

$$x^{(t+1)} = x^{(t)} - \alpha^{(t)} \left(\nabla^2 f(x^{(t)}) \right)^{-1} \nabla f(x^{(t)}). \tag{3.11}$$

A simpler alternative is to consider only $\operatorname{diag} \nabla^2 f(x^{(t)})$, the diagonal of the Hessian. As the exact inverse of the Hessian can be computationally costly to obtain, some approaches such as inexact Newton methods or quasi-Newton methods, e.g., Broyden–Fletcher– Goldfarb–Shanno (BFGS) method, do not compute the action of the Hessian or its inverse but rather approximate them using information that is often from previous iterations. Information from previous iterates may also be used in methods like heavy ball or conjugate gradient [116, 22, 23].

Step size. Various approaches exist to choose the step size $\alpha^{(t)}$ in Equation (3.10). This step size should be chosen carefully: if too big the values $f(x^{(t)})$ may not decrease, and if too small it could converge too slowly or not at all. Letting $h(\alpha) = f(x^{(t)} + \alpha d^{(t)})$, possible choices of $\alpha^{(t)}$ are:

- A **constant** step size : $\alpha^{(t)} = \alpha$ for all t.
- **Line minimization**:

$$\alpha^{(t)} \in \arg\min_{\alpha \geq 0} h(\alpha),$$

  which is usually implemented approximately.

– A **decreasing** $\alpha^{(t)}$ which satisfies the conditions

$$\sum_{t=0}^{\infty} \alpha^{(t)} = \infty \qquad \sum_{t=0}^{\infty} (\alpha^{(t)})^2 < \infty.$$

These conditions are often necessary to ensure some convergence properties, see for example Theorem 3.3.1.

– **Armijo** rule (or backtracking): α is decreased until the objective function has sufficiently decreased. Choose α such that

$$h(\alpha) \leq h(0) + \sigma\alpha(d^{(t)})^\top \nabla h(0)$$

with $0 < \sigma < 1$. α is chosen as $\tau^m \beta_0$ with $\beta_0 > 0$ fixed and m the smallest integer such that the condition holds.

– **Wolfe** conditions: add a curvature condition to Armijo, to ensure that the slope has been reduced sufficiently. We take α such that

$$h(\alpha) \leq h(0) + \sigma_1\alpha(d^{(t)})^\top \nabla h(0)$$

$$-(d^{(t)})^\top \nabla h(\alpha) \geq -\sigma_2(d^{(t)})^\top \nabla h(0)$$

with $0 < \sigma_1 < \sigma_2 < 1$.

The initial step size is another parameter to choose, which can influence the behavior of the algorithm.

Stopping criterion. In the smooth unconstrained case the goal of such methods are usually to find a point x satisfying the first-order necessary optimality condition $\nabla f(x) = 0$. When the problem is convex and $f(x)$ is bounded below, this condition is necessary and sufficient to have optimality. The presence of constraints and/or nonsmoothness of the objective function makes the choice of a stopping criterion more complicated. Two adapted stopping criteria are proposed in Sections 3.3.2.2 and 3.3.4.1 for Problem (3.8).

(Block-)Coordinate descent. An approach that is frequently used in practice is to cycle through the p coordinates. At each iteration, all variables are fixed except one on which the problem is optimized. In the next iteration, the process is repeated on another coordinate. We can also consider blocks of coordinates instead of a one by one procedure.

3.2.2 Nonsmoothness

The main difficulty for a nonsmooth problem is that the objective function f is not differentiable everywhere on its domain. As most of the previous methods require a gradient, they cannot be used without adaptations.

Reformulation. It is sometimes possible to reformulate the problem so that the nonsmoothness in the objective function disappears. For example,

$$\min_x \max(f_a(x), f_b(x))$$

can be reformulated as

$$\begin{aligned} \min t \quad \text{s.t.} \quad & t \geq f_a(x) \\ & t \geq f_b(x). \end{aligned}$$

However, this requires adding constraints to the problem.

Smoothing. A usual way to deal with nonsmooth function is to approximate it with a smooth function, whose accuracy depends on a parameter [23, Sections 2.2.5 and 6.2.2]. The approximated objective function being smooth, the gradient can be computed. The approximate problem is solved for increasingly better approximations, initializing the algorithm with the minimizer previously found.

Subgradient. Subgradient methods are a generalization of gradient-based methods for nondifferentiable functions. Instead of considering the gradient as descent direction, the methods consider a subgradient (for more details see for example [23, Chap. 3],[77] or [16, Chap. 3]). A subgradient g of a function f in x is a direction such that the linear approximation of f around x with a slope g will stay smaller than f.

Definition 3.2.1. A function $f : \mathbb{R}^p \mapsto [-\infty, \infty]$ is a **proper function** if it does not attain the value $-\infty$ and there exists at least one $x \in \mathbb{R}^p$ such that $f(x) < \infty$, that is $\text{dom}(f)$ is nonempty.

Definition 3.2.2. Let $f : \mathbb{R}^p \mapsto [-\infty, \infty]$ be a proper function and let $x \in \text{dom}(f)$. A vector $g \in \mathbb{R}^p$ is a **subgradient** of f at x if

$$f(y) \geq f(x) + \langle g, y - x \rangle$$

for all $y \in \mathbb{R}^p$. If this holds only for $y \in U$ where U is a neighbourhood of x, then g is a **local subgradient** of f at x.

In case of nonconvex f, we will use the local definition of subgradient.

Definition 3.2.3. The set of all local subgradients of f at x is called the local **subdifferential** of f at x and is denoted by $\partial f(x)$:

$$\partial f(x) = \{g \in \mathbb{R}^p | f(y) \geq f(x) + \langle g, y - x \rangle \text{ for all } y \in U\}$$

with U a neighbourhood of x. The subdifferential is global if $U = \mathbb{R}^p$.

For a function of the type $f(U) = \max\{f_1(U), \ldots, f_m(U)\}$ we have

$$\partial f(U) = \operatorname{conv}\{\nabla f_i(U) | i \in I_f(U)\}, \tag{3.12}$$

with $\operatorname{conv}(X)$ denoting the convex hull of the set X and $I_f(U)$ denoting the set of indexes for which $f_i(U) = \max_j f_j(U)$.

In the convex smooth case, a point u is a global minimum of $f(u)$ if and only if its gradient is null. In nonsmooth case, this condition is adapted using subgradient and is known as Fermat's optimality condition.

Theorem 3.2.1 (Fermat's optimality condition)**.** *Let $f : \mathbb{R}^p \mapsto [-\infty, \infty]$ be a proper convex function. Then*

$$x^* \in \arg\min\{f(x) : x \in \mathbb{R}^p\}$$

if and only if $0 \in \partial f(x^*)$.

3.2.3 Constraints

When D_x is not the whole space $\mathbb{R}^p$, the problem is constrained. In such a case, rewriting the problem in its dual form may help. Lagrange duality and associated KKT conditions are reviewed in Section 3.2.3.1. We can also rewrite the problem to incorporate the constraints in the objective function, and optimize this new objective function in $\mathbb{R}^p$. This leads to approaches like penalty methods, described in Section 3.2.3.2. When D_x is a manifold we can keep the objective function unchanged but require that the iterates belong to D_x. This leads to manifold methods, introduced in Section 3.2.3.3.

Feasibility. An important point to check is the feasibility of the considered problem: the set of feasible points for which all constraints hold, D_x, should be nonempty. Moreover, when computing a descent direction d, we should ensure that this direction is feasible: for an iteration of the form $x^{(t+1)} = x^{(t)} + \alpha^{(t)} d^{(t)}$, there should exist a small enough $\alpha > 0$ such that $x^{(t+1)}$ is still in D_x. Looking for such d leads to feasible direction methods like conditional gradient or projected gradient [23, Section 2.1.2].

3.2.3.1 Lagrange duality

Most constraints can usually be represented as (in)equalities. For example, we can consider the following (primal) formulation:

$$\begin{aligned} \min_x f(x) \quad \text{s.t.} \quad & g_i(x) \leq 0 && \forall\, i = 1, \ldots, m \\ & h_j(x) = 0 && \forall\, j = 1, \ldots, l. \end{aligned} \tag{3.13}$$

We can write the associated Lagrangian function $L : \mathbb{R}^p \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}$ as:

$$L(x, \gamma, \mu) = f(x) + \sum_i \gamma_i g_i(x) + \sum_j \mu_j h_j(x).$$

The γ_i's and the μ_j's are called the dual variables or Lagrange multipliers associated with the constraints f_i and h_j respectively. We can define the dual function

$$l(\gamma, \mu) = \inf_x L(x, \gamma, \mu)$$

which is a lower bound on f^* for any $\gamma \geq 0$, μ. The best lower bound can then be computed with

$$\max l(\gamma, \mu) \quad \text{s.t.} \quad \gamma \geq 0$$

and we have $l^* \leq f^*$ (weak duality). Indeed for all $\gamma \geq 0$ and feasible x (that is $h_i(x) = 0$, $g(x) \leq 0$), we have

$$\begin{aligned} l(\gamma, \mu) &= \inf_{x \in D_x} L(x, \gamma, \mu) \\ &\leq L(x, \gamma, \mu) = f(x) + \sum_i \gamma_i g_i(x) + \sum_j \mu_j h_j(x) \\ &\leq f(x). \end{aligned}$$

Since any feasible (γ, μ) gives us a lower bound on f^*, we can use it to evaluate how far we are from optimality. If x is primal feasible, we have

$$f(x) - f^* \leq f(x) - l(\gamma, \mu) \quad \Leftrightarrow \quad f^* \in [l(\gamma, \mu), f(x)].$$

The value $f(x) - l(\gamma, \mu)$ is called the duality gap and can be used as stopping criterion: if this gap is null we know that we are at optimality.

KKT conditions. We consider now when f, h_i, g_i are differentiable. Let assume that strong duality holds, and let x^* and (γ^*, μ^*) be any primal and dual optimal points with zero duality gap. Since x^* minimizes $L(x, \gamma^*, \mu^*)$ over x, its gradient should vanish at x^*. The following conditions should then hold at optimality:

– Stationarity:

$$-\nabla f(x^*) = \sum_{i=1}^{m} \gamma_i^* \nabla g_i(x^*) + \sum_{j=1}^{l} \mu_j^* \nabla h_j(x^*)$$

– Primal feasibility:

$$\begin{aligned} g_i(x^*) &\leq 0 \qquad && \forall\ i = 1, \dots, m \\ h_j(x^*) &= 0 && \forall\ j = 1, \dots, l \end{aligned}$$

– Dual feasibility:

$$\gamma_i^* \geq 0 \quad \forall \quad i = 1, \dots, m$$

– Complementary slackness:

$$\gamma_i^* g_i(x^*) = 0 \quad \forall \quad i = 1, \dots, m.$$

Those conditions are called the Karush-Kuhn-Tucker (KKT) conditions and are necessary for the points to be optimal. As this set of equations is analytically solvable only in a few cases, many optimization algorithms are conceived to look for points such that those conditions hold.

3.2.3.2 Approximation methods

The approximation methods solve

$$x = \arg \min_{x \in D_x} f(x)$$

iteratively by solving an approximation of the initial problem:

$$x^{(t+1)} = \arg \min_{x \in D_x^{(t)}} f^{(t)}(x).$$

In **penalty method**, the problem

$$\min f(x) \quad \text{s.t.} \quad x \in D_x, \quad g_i(x) \leq 0$$

with $i = 1, \dots, m$ is approximated by

$$\min f(x) + c^{(t)} \sum_i P\left(g_i(x)\right)) \quad \text{s.t.} \quad x \in D_x$$

with $P(y) = 0$ if $y \leq 0$, $P(y) > 0$ either and $c^{(t)} > 0$. Parameter $c^{(t)}$ should increase with t to have an increasingly accurate approximation of the solution.

In the **Augmented Lagrangian** a linear term is added to the penalty function:

$$\min f(x) + c^{(t)} \sum_i P\left(g_i(x)\right) + \sum_i \lambda_i^{(t)T} g_i(x) \quad \text{s.t.} \quad x \in D_x$$

with $\lambda^{(t)} \in \mathbb{R}^m$. The minimizing $x^{(t)}$ is obtained, then the λ is updated with a

rule aiming to approximate an optimal dual solution. An example could be

$$\lambda^{(t+1)} = \lambda^{(t)} + c^{(t)} \sum_i g_i(x).$$

Interior point methods use barrier functions $P(y)$ to prevent the iterates from moving too close to the boundary of the feasible region. A typical example is $P(g_i(x)) = -\log(-g_i(x))$ or $P(g_i(x)) = -\frac{1}{g_i(x)}$. At each iteration, all inequalities are satisfied strictly.

3.2.3.3 Riemannian methods

When dealing with constrained optimization problems, a large family of methods is the optimization on manifolds. Instead of iterating in $\mathbb{R}^p$ and integrating the constraints into the objective function, such methods work directly on D_x without modifying the objective function. The previously described optimization methods can be used, provided that the usual tools like gradient lines, and norms are adapted.

The advantage of requiring that the iterates belong to a manifold $D_x = \mathcal{M}$ is that locally, a manifold is smooth and can be locally approximated by an Euclidean tangent space. At each iteration x, the space tangent to $\mathcal{M}$ at x, $T_x\mathcal{M}$, is used and all the necessary information like gradient or step size are computed in this tangent space and then mapped back to the manifold. To switch between the values on the Riemannian manifold $\mathcal{M}$ and their local representations on $T_x\mathcal{M}$, we need the tools that are briefly discussed next. Specific formulas are given for the Grassmannian manifold as they are used later in Sections 3.3.1 and 3.3.4. Definitions given here are informal and meant to give the general intuition behind the concepts rather than a formal definition. For more details on Riemannian geometry, see for example [112, 3], or [127, 50, 4, 46] for the Grassmann or the Stiefel manifold more specifically.

Definition 3.2.4. A **topological manifold** is a set that is locally Euclidean: every point has a neighbourhood homeomorphic to an open subset of Euclidean space. Those homeomorphisms are called charts. Any topological manifold can be described by a collection of charts, also known as an atlas.

Definition 3.2.5. A differentiable or smooth **manifold** is a topological manifold such that the transition from one chart to another is differentiable.

The differentiable structure of such differentiable or smooth manifolds allows calculus to be done on manifolds. From here and for the rest of the thesis all manifolds are assumed smooth.

Definition 3.2.6. Let $\mathcal{M}$ be a manifold in $\mathbb{R}^p$. The **tangent space** of a point $x \in \mathcal{M}$, noted $T_x\mathcal{M}$, is the linear subspace of $\mathbb{R}^p$ generated by the directional derivatives at x of all possible smooth curves on $\mathcal{M}$ passing through x. The dimension of $T_x\mathcal{M}$ is the same as the manifold.

Definition 3.2.7. A **Riemannian metric** is an inner product g_x defined on the tangent space $T_x\mathcal{M}$ at each point x that varies smoothly from point to point. The inner product between two points $\xi, \eta \in T_x\mathcal{M}$ is noted $\langle \xi, \eta \rangle_x$.

Definition 3.2.8. A **Riemannian manifold** is a smooth manifold with a metric.

A Riemannian metric makes it possible to define several geometric notions on a Riemannian manifold, such as angle or distances.

Definition 3.2.9. The **length** of a C^1 curve $\gamma(t) : [0, 1] \to \mathcal{M}$ is defined as

$$L(\gamma) = \int_0^1 ||\dot{\gamma}(t)||_{\gamma(t)} dt$$

with $||\dot{\gamma}(t)||_{\gamma(t)} = \sqrt{\langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle_{\dot{\gamma}(t)}}$ and $\dot{\gamma}$ the velocity vector of γ.

Definition 3.2.10. The **Riemannian distance** $d(x, y)$ between two points x and y on $\mathcal{M}$ is the infimum of the length of all C^1 curves on $\mathcal{M}$ going from x to y.

We define a **ball** $B(c, r)$ on $\mathcal{M}$ as the set of points $x \in \mathcal{M}$ such that $d(c, x) \leq r$.

Definition 3.2.11. A **geodesic** is a smooth path on $\mathcal{M}$ which locally minimizes the distance between two points. A **minimizing geodesic** is a geodesic that minimizes globally this distance.

In this thesis when not specified we consider minimizing geodesics.

Definition 3.2.12. A **geodesic hinge** γ_1, γ_2, α in $\mathcal{M}$ consists of two nonconstant geodesic segments γ_1, γ_2 with the same initial point making the angle α. A minimal connection γ_3 between the endpoints of γ_1 and γ_2 is called a closing edge of the hinge [100, Section 2].

An example is given on Figure 3.5 of a geodesic hinge made of γ_1, γ_2 the plain grey lines starting from x to x_1 and x_2 and the closing edge is the dashed grey line from x_1 to x_2.

Definition 3.2.13. If $\gamma(t)$ defined on $[0, 1]$ denotes the minimizing geodesic from x to y, then the **logarithm** operator $\operatorname{Log}_x y$ is the derivative of $\gamma(t)$ at $t = 0$. $\operatorname{Log}_x y$ maps point $y \in \mathcal{M}$ onto $T_x\mathcal{M}$ and is such that $\operatorname{dist}(x, y) = d(0, \operatorname{Log}_x y) = ||\operatorname{Log}_x y||_2$.

Definition 3.2.14. The **exponential** operator $\operatorname{Exp}_x y^e$ is equal to $\gamma(1)$ where γ is the geodesic with initial point $x \in \mathcal{M}$ and initial velocity y^e. $\operatorname{Exp}_x y^e$ maps y^e from $T_x\mathcal{M}$ to $\mathcal{M}$. On its domain, $\operatorname{Log}_x y^e$ is the inverse of Exp : $\operatorname{Exp}_x (\operatorname{Log}_x y) = y$.

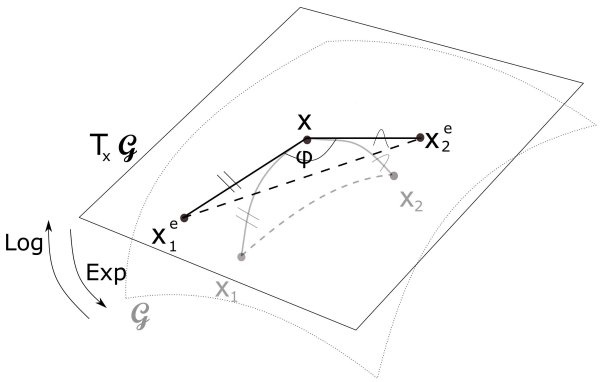


Figure 3.5: Illustration of Topogonov's theorem

Definition 3.2.15. The **injectivity radius** of a manifold $\mathcal{M}$, $\text{inj}(\mathcal{M})$, is the largest radius r such that for all $x \in \mathcal{M}$, $\text{Exp}_x : B(0,r) \to B(x,r)$ is a diffeomorphism.

This implies that $\text{Log}_x(y)$ is well-defined for any $y \in B(x, \text{inj}(\mathcal{M}))$.

Definition 3.2.16. Roughly, the **cut locus** of a point x in $\mathcal{M}$, $\text{Cut}(x)$, is the set of all points for which there exists multiple minimizing geodesics connecting them to x. $\text{Cut}(x)$ is of measure null for any x on a complete, connected C^∞-Riemannian manifold [72]

Definition 3.2.17. Given $x \in T_x\mathcal{M}$, u, v two noncolinear vectors in $T_x\mathcal{M}$, the **sectional curvature** $\text{Sect}(u,v)=K$ is a number which gives information on how the geodesics issued from x behave near x. For example, if $K > 0$, $\text{Exp}_x(\text{span}(u,v))$ is near x approximately a 2-dimensional sphere with radius $\frac{1}{\sqrt{K}}$ [11].

The Euclidean space has a zero curvature.

Theorem 3.2.2. ***Topogonov's theorem*** *[100]: Let $\mathcal{M}$ be a complete Riemannian manifold with sectional curvature $K \geq \kappa$, and let M^n_κ denote the n-dimensional hyperbolic space $\mathbb{H}^n_\kappa$ of curvature κ if $\kappa < 0$, the Euclidean space $\mathbb{R}^n$ if $\kappa = 0$ and the standard sphere $\mathbb{S}^n_\kappa$ of radius $\frac{1}{\sqrt{\kappa}}$ if $\kappa > 0$. Let γ_1, γ_2, ϕ be a hinge in $\mathcal{M}$ with γ_2 minimal and γ_3 a closing edge. Then the closing edge $\tilde{\gamma}_3$ of any hinge $\tilde{\gamma}_1$, $\tilde{\gamma}_2$, α in M^2_κ with $L(\tilde{\gamma}_1) = L(\gamma_1)$,$L(\tilde{\gamma}_2) = L(\gamma_2)$ satisfies $L(\tilde{\gamma}_3) \geq L(\gamma_3)$.*

An illustration is given on Figure 3.5 with γ_1, γ_2 the plain grey lines starting from x to x_1 and x_2 and γ_3 the dashed grey line from x_1 to x_2. Their equivalent in M^2_κ, $\tilde{\gamma}_1$, $\tilde{\gamma}_2$ and $\tilde{\gamma}_3$ are in black.

Adaptation of line search methods on manifolds is intuitively simple: the Equation (3.10) becomes

$$x^{(t+1)} = R_{x^{(t)}}(\alpha^{(t)} d^{(t)}) \tag{3.14}$$

with $d^{(t)}$ in $T_{x^{(t)}}\mathcal{M}$ and R an operator mapping back $\alpha^{(t)}d^{(t)}$ to $\mathcal{M}$ like the Exp operator [3]. Adaptations of existing methods for nondifferentiable objective function to Riemannian manifolds were proposed for example in [63, 62, 65].

In this thesis we always consider Riemannian manifolds, with their classical metric if not specified. The most simple example of a Riemannian manifold is the Euclidean space with its usual Euclidean metric. We will principally work with the Grassmann manifold, and to a lesser extent with the Stiefel manifold.

The **Stiefel manifold** $\mathrm{St}(k,p)$ is the set of all orthonormal k-frames in $\mathbb{R}^p$:

$$\mathrm{St}(k,p) = \left\{U \in \mathbb{R}_*^{p\times k} : U^\top U = I_k\right\}$$

where $\mathbb{R}_*^{p\times k}$ stands for the set of full-rank $p \times k$ matrices. The tangent space $T_X \mathrm{St}(k,p)$ to $\mathrm{St}(k,p)$ at X is [46]:

$$T_X \mathrm{St}(k,p) = \left\{Y \in \mathbb{R}^{p\times k} : X^\top Y + Y^\top X = 0\right\}.$$

For any U_1, $U_2 \in T_X \mathrm{St}(k,p)$, we consider the classical metric $\langle U_1, U_2\rangle_X = \mathrm{Tr}(U_1^\top U_2)$.

The **Grassmann manifold** $\mathcal{G}(k,p)$ is the set of k-dimensional subspaces in $\mathbb{R}^p$:

$$\mathcal{G}(k,p) = \left\{\mathcal{U} = \mathrm{span}(U) : U \in \mathbb{R}_*^{p\times k}\right\}.$$

The tangent space $T_X\mathcal{G}(k,p)$ to $\mathcal{G}(k,p)$ at X is identified with [3, Section 3.6.2]:

$$T_X\mathcal{G}(k,p) = \left\{Y \in \mathbb{R}^{p\times k} : X^\top Y = 0\right\}.$$

For any U_1, $U_2 \in T_X\mathcal{G}(k,p)$, we consider the classical metric $\langle U_1, U_2\rangle_X = \mathrm{Tr}(\check{U}_1^\top \check{U}_2)$ with $\check{U}$ an orthonormal basis of $\mathcal{U}$.

As practically we will use a specific matrix representation of the subspaces on a Grassmann manifold, we will use U and X_i to denote $\mathcal{U}$ and $\mathcal{X}_i$, the column spaces of U and of X_i, when no risk of confusion arises. From here, the notation $(\cdot)^e$ will imply that we are working in the Euclidean tangent space. We provide here some specific formula that will be useful in algorithms on Grassmann later.

On the Grassmann manifold, we have

$$\mathrm{Log}_U X = BDA^\top$$

with $A\Gamma_D C^\top$ a thin SVD of $U^\top X$, and $B\Sigma_D C^\top$ a thin SVD of $(I - UU^\top)X$. Matrices $\Gamma_D = \cos(D)$ and $\Sigma_D = \sin(D)$ are diagonal matrices of respectively the cosines and the sines of the principal angles between $\mathcal{U}$ and $\mathcal{X}$. Equivalently

we can write

$$\operatorname{Log}_U X = U_2 \operatorname{atan} D U_1^\top$$

with $U_2 D U_1$ a thin SVD of $\big((X - UU^\top X)(U^\top X)^{-1}\big)$. This formula is valid only if all principal angles are smaller than $\frac{\pi}{2}$: in this case the minimizing geodesic between U and X is not unique.

The exponential operator can be computed as

$$\operatorname{Exp}_U X^e = UB \cos \Sigma B^\top + A \sin \Sigma B^\top$$

with $A\Sigma B^\top$ a compact SVD of X^e.

Note that the objects on the Grassmann manifold represented by $\operatorname{Log}_U X$ and $\operatorname{Exp}_U X^e$ do not depend on the specific matrix representation of $\mathcal{U}$ and $\mathcal{X}$.

Finally on the Grassman manifold, $\mathcal{G}$, we have

$$\operatorname{Cut}(\mathcal{U}) = \big\{\mathcal{X} \in \mathcal{G} | \dim(\mathcal{X} \cap \mathcal{U}^\perp) \geq 1\big\}$$

with $\mathcal{U}^\perp$ the orthogonal complement of $\mathcal{U}$ in $\mathcal{G}$ [170, Theorem 8]. That is, $\mathcal{X} \notin \operatorname{Cut}(\mathcal{U})$ if the largest principal angle between $\mathcal{U}$ and $\mathcal{X}$ is smaller than $\frac{\pi}{2}$.

3.2.4 Nonconvexity

Globality of the solution. An illustration in $\mathcal{G}(3,1)$ of Problem (3.8) with $d_{\sin\phi}$ and $m = 4$ is given on Figures 3.6 and 3.7. Each subspace of dimension 1 is represented by a direction in $\mathbb{R}^3$ or a point on the three dimensional sphere. Each (θ, ϕ) represents a point on $\mathcal{G}(3,1)$ using its spherical coordinates. These figures show that Problem (3.8) can have local nonglobal minimizers. In particular, it is not convex.

Nonconvex problems are generally difficult to solve exactly as they can have many local minima, saddle points, very flat regions, widely varying curvature. Consequently, the theoretical guarantees are often weak or do not exist. Some specific cases can be solved exactly, like the principal component analysis problem, but most of them cannot. Algorithms for convex problems can be applied to nonconvex ones, but generally without convergence guarantees. In many cases, the best that can be done is convergence to a local minimum or a stationary point. Convergence to a global minimum can sometimes be ensured if the initial iterate is close enough to such a global minimum, i.e., the initialization step is then very important.

Initialization. In nonconvex problems, the initialization can have a large effect on the final solution. To avoid depending on the initial iterate, a possibility is to run the chosen algorithm a few times with various initial iterates. The final solution is taken as the one with the minimum cost function value. Another approach is to carefully chose the first iterate.

For the Riemannian minimax problem, in Theorem 3.2.3 it is proved that if points X_i belong to a ball $B(c, R)$ with R small enough, then the minimax

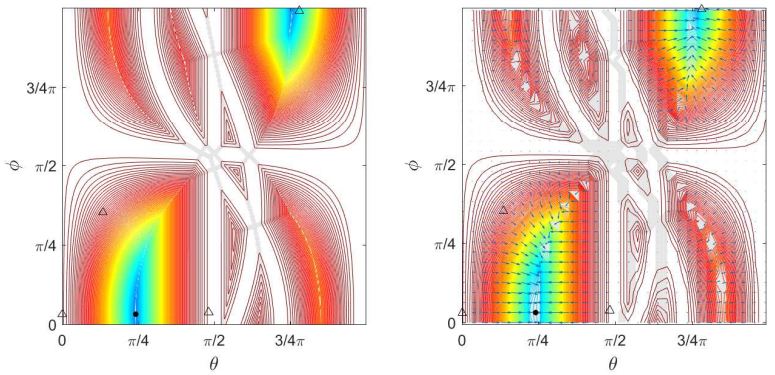


Figure 3.6: Example of objective function with four points in $\mathbb{R}^3$. Black dot: global minimum - black triangles: data points - gray lines: nondifferentiable points (at least two data points at same distance) - blue arrows: anti-gradient.

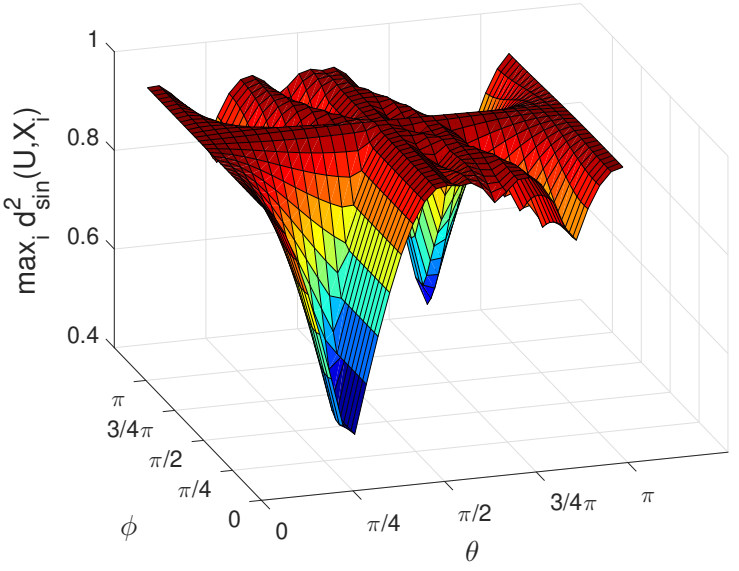


Figure 3.7: Example of objective function with four points in $\mathbb{R}^3$.

center is unique and lies inside B. Hence we choose to initialize our algorithms with a point within the convex set generated by the points X_i.

Theorem 3.2.3. *[5] Let $(\mathcal{M}, d)$ be a complete Riemannian manifold with sectional curvatures upper bound of Δ and injectivity radius of* $\text{inj}(\mathcal{M})$. *Define*

$$\rho_\Delta = \frac{1}{2}\min\{\text{inj}(M), \frac{\pi}{\sqrt{\Delta}}\}.$$

Let ν be a probability measure on M such that $supp(\nu) \in B(o, \rho) \in M$. Assume $\rho < \rho_\Delta$. Then the l_∞ center of mass with respect to ν is a unique point and lies inside B. The l_∞ center of mass is defined as any minimizer of

$$f(x) = \max_{s \in \overline{supp}(\nu)} d(x, s)$$

with d a distance function on $\mathcal{M}$.

Stopping criterion. As it is difficult to evaluate if a U is the global minimum, we would like to at least stop close to a critical point. By critical point, we mean a point such that the first-order necessary conditions to have a minimum hold. Moreover, at optimality U should have at least two X_i's realizing the maximum dissimilarity. Otherwise, we could move U a little bit toward the furthest point X_f until the second furthest point is at the same dissimilarity value. The objective function is thus not differentiable at optimality.

In the nonconvex case, the necessary and sufficient condition in Theorem 3.2.1 that 0 should belong to the subdifferential becomes a necessary condition [77, Lemma 2.14], using local subdifferential. Ideally, the algorithms should stop at least close to a point with 0 in its local subdifferential.

Duality gap. As explained in Section 3.2.3.1, any solution of the dual formulation gives a lower bound on the optimal solution. If a point is such that the dual objective function value is equal to the primal one, then this point is optimal.

3.3 Algorithms

We present here different approaches to solve Problem (3.8). The first algorithm is an adapation to the Grassmann manifold of the greedy algorithm proposed in [11]. The second one is directly inspired from the interpretation of the KKT conditions, the third is a projected subgradient, the fourth is a feasible direction approach on the Grassmann manifold and the last one uses a smoothing approach on the Stiefel manifold.

Principal characteristics of these algorithms are summarized in Table 3.2. The "Constr." column details which type of constraints are considered: St or $\mathcal{G}$ means that the corresponding manifold approach is used. The type of

dissimilarity is given in the "Diss." column and the "Nonsmoothness" column details how the nonsmoothness of the objective function is dealt with. If a specific stopping criterion is used it is given in the corresponding column, and the type of algorithm in reference to Section 3.1.2 is given in the "Approach" column. The section of the thesis where the algorithm is described is given in last column. Detailed comparisons of those algorithms on synthetic and real datasets are given in Section 3.4.

Algorithm	Constr.	Diss.	Nonsmoothness	Stopping criterion	Approach	Section
Greedy approach	$\mathcal{G}$	d_ϕ	Subgradient	(2 X_i's at max. diss.)	Primal	3.3.1
KKT	$U^TU = I$	$d_{\sin\phi}$	Reformulation	Duality gap	Primal-dual	3.3.2
Subgradient	$U^TU = I$	$d_{\sin\phi}$	Subgradient	(2 X_i's at max. diss.)	Primal	3.3.3
Descent direction	$\mathcal{G}$	d_ϕ	Descent direction	Critical point	Primal	3.3.4
LogExp smoothing	St	$d_{\sin\phi}$	Smoothing	Smoothing param. μ	Primal	3.3.5

Table 3.2: Proposed algorithms

3.3.1 Greedy

In [13], a fast and simple procedure is proposed to find the center of a collection of points in a finite-dimensional Euclidean space (see Section 3.1.2) and extended to Riemannian manifold in [11]. We describe the Grassmannian version of the algorithm in Section 3.3.1.1 and briefly discuss its convergence in Section 3.3.1.2. This section in mainly based on results published in [123].

3.3.1.1 Algorithm

The procedure extended to arbitrary Riemannian manifolds in [11] is straightforward:

- Initialize the candidate solution $U^{(t)}$ with a point in the set
- Iteratively update as:

$$U^{(t+1)} = \text{Geodesic}\left(U^{(t)}, X_f^{(t)}, \frac{1}{t+1}\right)$$

  where $X_f^{(t)}$ is the farthest point to $U^{(t)}$, and $\text{Geodesic}(p, q, t)$ represents the intermediate point m on the geodesic passing through p and q such that $\text{dist}(p, m) = t\,\text{dist}(p, q)$.

This approach was used in [10] to compute the Riemannian l_∞ center of mass of structure tensor images in order to denoise those images.

Unlike the problem in [11], in Problem (3.8) points represent subspaces of different dimensions n_i and therefore belong to different manifolds $\mathcal{G}(n_i, p)$. The first consequence is that the usual Grassmannian distance cannot be used to determine the farthest point $X_f^{(t)}$: the adapted dissimilarity d_ϕ is used to

address this difficulty. The second consequence is that to update the current iterate $U^{(t)}$ using a geodesic, $X_f^{(t)}$ should be first projected on $\mathcal{G}(k,p)$. However, since the dissimilarity used is based on the principal angles, by Proposition 3.1.1 we can compute a projection of $\mathcal{X}_f \in \mathcal{G}(n_f,p)$ on $\mathcal{G}(k,p)$ that minimizes the distance with $\mathcal{U}$ on $\mathcal{G}(k,p)$. We can then update U using the corresponding geodesic.

An adaptation is proposed in Algorithm 10. We initialize using a k-truncated SVD of $Y = [\check{X}_1, \check{X}_2, ..., \check{X}_m]$, which can be interpreted as initializing with a mean (with respect to $d^2_{\sin\phi}$) (lines 5-6), and stop when the two farthest subspaces have close dissimilarity values (line 16). As explained in Section 3.3.2.1, this is a necessary, but not sufficient, condition for optimality. The farthest X_i from current $U^{(t)}$ is determined using the chosen dissimilarity based on the principal angles (lines 9 to 12). If more than one point X_i is at maximal dissimilarity of $U^{(t)}$, one is picked at random. $U^{(t)}$ is then updated in the direction of X_f with a step $\frac{1}{t+1}$ along the Grassmannian geodesic [50] (line 14). The projection of $X_f^{(t)}$ on $\mathcal{G}(k,p)$ is included in the geodesic computation given in Algorithm 11.

Algorithm 10 Greedy approach to compute the minimax center in $\mathcal{G}(k,p)$.

Require: Stopping criterion tol, dimension k, data X_1, ...,X_m

1: $t \leftarrow 1$
2: **for all** X_i **do**
3: $\quad \check{X}_i \leftarrow \text{orthonormalBasis}(X_i)$
4: **end for**
5: $Y \leftarrow [\check{X}_1 ... \check{X}_m]$
6: $U^{(0)} \leftarrow k$ first left singular vectors of Y
7: $\text{err}^{(0)} \leftarrow \text{tol} + 1$
8: **while** $\text{err}^{(t)} \geq \text{tol}$ **do**
9: $\quad$ **for all** $\check{X}_j$ **do**
10: $\quad\quad d_j^{(t)} \leftarrow \text{d}_\phi(U^{(t)}, \check{X}_j)$
11: $\quad$ **end for**
12: $\quad f \leftarrow \arg\max_i d_i^{(t)}$
13: $\quad A_f D_f B_f^{(t)} \leftarrow \text{SVD}(U^{(t)T} \check{X}_f)$
14: $\quad U^{(t+1)} \leftarrow \text{Geodesic}\left(U^{(t)}, X_f^{(t)} B_f, \frac{1}{t+1}\right)$
15: $\quad d_{\text{sorted}} \leftarrow \text{sort}_{\text{decreasing}}(d^{(t)})$
16: $\quad \text{err}^{(t)} \leftarrow d_{\text{sorted}}(1) - d_{\text{sorted}}(2)$
17: $\quad t \leftarrow t + 1$
18: **end while**

Algorithm 11 Geodesic (A, B, t) on $\mathcal{G}(k, p)$.

Require: A and B orthonormal basis of $\mathcal{A}$ and $\mathcal{B}$ on $Gr(k, p)$, $t \in [0, 1]$
1: $U_c D_c V_c^\top \leftarrow \text{SVD}(A^\top B, k)$
2: $S_0 \leftarrow AU_c$
3: $S_1 \leftarrow BV_c$
4: $\Theta \leftarrow \arccos \text{diag}\, D_c$
5: $\Gamma_\alpha \leftarrow \text{diag} \cos \alpha\Theta$
6: $\Sigma_\alpha \leftarrow \text{diag} \sin \alpha\Theta$
7: $P \leftarrow S_0\Gamma_t + (S_1 - S_0\Gamma_1)\,\Sigma_1^{-1}\Sigma_t$

3.3.1.2 Convergence

When the set of X_i's realizing the maximal distance (line 12) is only one X_i, we can then link the update (line 14) to a gradient step. Indeed, we have that

$$\nabla d^2(U, X_i) = -2 \operatorname{Log}_U X_i$$

[12, Eq. 5], and doing a step of size $\frac{1}{t}$ along the geodesic from U to $X_f^{(t)}$ is equivalent to going in the direction $\operatorname{Log}_U X_i$, followed by a mapping onto $\mathcal{G}$.

In [11], a convergence analysis of the procedure described at the beginning of this subsection is made on Riemannian manifold based on Theorem 3.2.3. The convergence of the procedure to the minimax center is guaranteed for manifolds with (κ, Δ) finite lower and upper bounds on the sectional curvature, if all points X_i are within a ball $B(c, R)$ with

$$R < \frac{1}{2} \min \left\{ \text{inj}(\mathcal{M}), \frac{\pi}{\sqrt{\Delta}} \right\}.$$

As the injectivity radius $\text{inj}(\mathcal{M})$ is equal to $\frac{\pi}{2}$ for the Grassmann manifold and the sectional curvature upper bound Δ is equal to 2 [154], we have $R = \frac{\pi}{4}$: all $d_\phi(X_i, X_j)$ should be less than $\frac{\pi}{2}$.

This convergence proof is valid only if all the $\check{X}_i$ and U belong to the same Grassmannian $\mathcal{G}(k, p)$. To apply this convergence analysis, the $\check{X}_i$ should be first reduced to a subspace of dimension k. However, many possibilities exist to reduce dimension of the X_i's. Choosing one or another specific reduced version of X_i and then applying the approach developped in [11] can lead to different solutions.

Block coordinate descent interpretation. We can easily adapt Algorithm 10 as a block coordinate descent on

$$\min_{U, B_1, \ldots, B_m} f(U, B_1, \ldots, B_m) = \min_{U, B_1, \ldots, B_m} \max_i d_\phi\left(U, X_i B_i\right).$$

We can compute the best B_i's for fixed U, and then the best U for fixed B_i's.

Computing $B_i^{(t)} = \arg\min f$ for a fixed $U^{(t)}$ is easy since the optimal $B_i^{(t)}$ does not depend on the other B_j's, $j \neq i$:

$$B_i^{(t)} = \arg\min_{B_i} f(U^{(t)}, B_1, \ldots, B_m) = \arg\min_{B_i} d_\phi(U^{(t)}, X_i B_i). \tag{3.15}$$

This problem is solved using SVD by Proposition 3.1.1.

With the $B_i^{(t)}$ fixed, we look for

$$U^{(t+1)} = \arg\min_{U} \max_{i} d_\phi(U, X_i B_i^{(t)}). \tag{3.16}$$

This problem is equivalent to finding the Grassmannian 1-center of the set of points $\left\{X_1 B_1^{(t)}, \ldots, X_m B_m^{(t)}\right\}$, which can be solved using the algorithm of [11] if the $X_i B_i$ respect some conditions (i.e., all $X_i B_i$ belongs to a ball $B(o, R)$ with $R < \frac{\pi}{4}$).

This two-step approach is implemented in Algorithm 12: exact minimization on B_i is done in lines 9-12, while computation of a new U decreasing the objective function value is done in lines 13-25. By Theorem 3.2.3, if all $X_i B_i$ belong to a ball $B(o, R)$ with $R < \frac{\pi}{4}$, such a decrease is always possible, except if the current iterate is already the minimax center of the $X_i B_i$'s. An illustration of the behavior of this implementation (with $l_0^{(t)} = t$) is given in Figure 3.8. Each red circled point corresponds to the cost function value after computation of the optimal $B_i^{(t)}$ while the intermediate points correspond to the objective evolution using a few steps of algorithm from [11].

An example of the behavior of Algorithms 10 and 12 on data generated using model (3.3) is shown in Figure 3.8. Algorithm 10 does not guarantee a decrease at each iteration: while the objective function decreases fast at the beginning, some small increases in the objective function can be observed later. Adaptation as a block-coordinates descent in Algorithm 12 removes those small increases (see the squares on Figure 3.8). We use Algorithm 10 in more extended tests in Section 3.4.

Algorithm 12 Block coordinate adaptation of the greedy approach to compute the minimax center in $\mathcal{G}(k,p)$.

Require: Stopping criterion tol, dimension k, data X_1, ...,X_m
$t \leftarrow 1$
for all X_i **do**
    $\check{X}_i \leftarrow \text{orthonormalBasis}(X_i)$
end for
$Y \leftarrow [\check{X}_1 ... \check{X}_m]$
$U^{(0)} \leftarrow k$ first left singular vectors of Y
$\text{err}^{(0)} \leftarrow \text{tol} + 1$
while $\text{err}^{(t)} \geq \text{tol}$ **do**
    for all $\check{X}_j$ **do**
        $A_j D_j B_j^{(t)} \leftarrow \text{SVD}(U^{(t)T} \check{X}_j)$
        $d_j^{(t)} \leftarrow d_\phi(U^{(t)T}, \check{X}_j B_j^{(t)})$
    end for
    $l \leftarrow l_0^{(t)}$
    $d_0 \leftarrow \max_i d_i^{(t)}$
    $U_{tmp} \leftarrow U^{(t)}$
    while $\max_i d_j^{(t)} \geq d_0$ **do**
        $I_f \leftarrow \arg\max_i d_i^{(t)}$
        $f \leftarrow i \in I_f$
        $\delta \leftarrow \frac{1}{l+1}$
        $U_{tmp} \leftarrow \text{Geodesic}\left(U_{tmp}, X_f^{(t)} B_f^{(t)}, \frac{1}{t+1}\right)$
        for all $\check{X}_j$ **do**
            $d_j^{(t)} \leftarrow d_\phi(U_{tmp}, \check{X}_j B_j^{(t)})$
        end for
        $l \leftarrow l + 1$
    end while
    $U^{(t+1)} \leftarrow U_{tmp}$
    $d_{\text{sorted}} \leftarrow \text{sort}_{\text{decreasing}}(d^{(t)})$
    $\text{err}^{(t)} \leftarrow d_{\text{sorted}}(1) - d_{\text{sorted}}(2)$
    $t \leftarrow t + 1$
end while

3.3.2 KKT-based

If we consider Problem (3.8) with dissimilarity $d = d_{\sin\phi}$, after a reformulation in the spirit of Section 3.2.2 we can easily state the KKT conditions that should hold at optimality. We first derive those conditions in Section 3.3.2.1 and propose an algorithm to compute a solution in Section 3.3.2.2. A brief discussion on convergence of this algorithm is given in Section 3.3.2.3. This

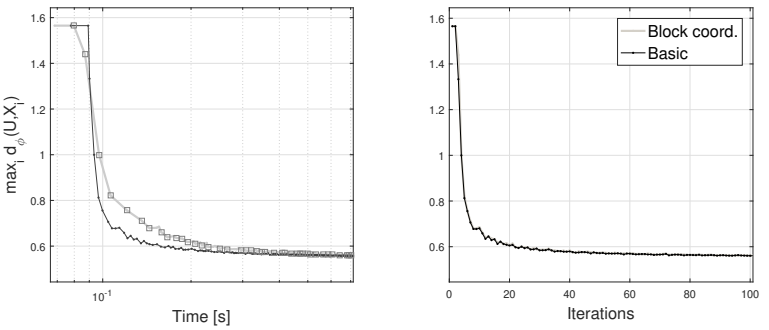


Figure 3.8: Example of the evolution of the objective function using Algorithms 10 and 12 on data generated using model (3.3). Squares correspond to one 'complete' iteration, that is each time t is incremented at line 29 of Algorithm 12.

section in mainly based on results published in [120].

3.3.2.1 KKT conditions

With dissimilarity $d_{\sin\phi}$, Problem (3.8) becomes:

$$\min_{U\in\mathbb{R}^{p\times k}} \max_i k - \mathrm{Tr}(\check{U}^\top \check{X}_i \check{X}_i^\top \check{U}).$$

Since k is fixed and $\check{U}$ verifies $\check{U}^\top\check{U} = I_k$, this is equivalent to

$$\max_{U^\top U=I} \min_i \mathrm{Tr}(U^\top \check{X}_i \check{X}_i^\top U).$$

Since $\max_U \min_i f_i(U)$ is equivalent to $\max_{U,\tau} \tau$ subject to $\tau \leq f_i(U)$ for all i, this is equivalent to:

$$\max_{U,\tau} \tau \tag{3.17}$$

$$\text{s.t. } \tau - \sum_{l=1}^{k} u_l^\top \check{X}_i \check{X}_i^\top u_l \leq 0 \qquad \forall i = 1,...,m \tag{3.17a}$$

$$u_j^\top u_j - 1 = 0 \qquad \forall j = 1,...,k \tag{3.17b}$$

$$u_j^\top u_l = 0 \qquad \forall l \neq j,\ j = 1,...,k\ ;\ l = 1,...,k \tag{3.17c}$$

with u_i the ith column of U. Observe that (3.17) is an optimization problem with a linear objective function and quadratic (in)equality constraints.

We can derive the first order necessary conditions of optimality for problem (3.17) (KKT conditions, see e.g., [108] or Section 3.2.3.1). Associating Lagrange multipliers γ_i's with constraints (3.17a), M_{jj}'s with constraints (3.17b)

and M_{jl}'s with constraints (3.17c), the KKT conditions can be written as:

$$\sum_i \gamma_i = 1 \tag{3.18a}$$

$$\left(\sum_i \gamma_i \check{X}_i \check{X}_i^\top\right) U = UM \tag{3.18b}$$

$$U^\top U = I \tag{3.18c}$$

$$\tau - \operatorname{Tr}\left(U^\top \check{X}_i \check{X}_i^\top U\right) \leq 0 \quad \forall i = 1, ..., m \tag{3.18d}$$

$$\gamma_i \geq 0 \quad \forall i = 1, ..., m \tag{3.18e}$$

$$\gamma_i \left(\tau - \operatorname{Tr}\left(U^\top \check{X}_i \check{X}_i^\top U\right)\right) = 0 \quad \forall i = 1, ..., m. \tag{3.18f}$$

The M_{ij}'s correspond to the Lagrange multipliers associated with constraints $u_i^\top u_j = 0$ and the M_{ii}'s to $u_i^\top u_i - 1 = 0$, so M is symmetric. Therefore there exists a diagonal matrix D and an orthogonal matrix Q such that $M = QDQ^\top$. We have then $\left(\sum_i \gamma_i \check{X}_i \check{X}_i^\top\right) UQ = UQD$ which means that UQ is a matrix of eigenvectors of $\sum_i \gamma_i \check{X}_i \check{X}_i^\top$.

We can interpret the conditions as following:

- Conditions (3.18b) and (3.18c) are equivalent to finding eigenvectors of the matrix $\sum_i \gamma_i \check{X}_i \check{X}_i^\top$.

- Condition (3.18a) combined with (3.18b) and (3.18e) can be interpreted as finding an optimal weighting $\boldsymbol{\gamma}$ of the different subspaces, where $\boldsymbol{\gamma}$ is the vector of all γ_i. U should then be an eigenvector of a convex combination of the $\check{X}_i \check{X}_i^\top$'s.

- Condition (3.18f) implies that for each $\check{X}_i$, either $\gamma_i = 0$ which means that the corresponding dataset is not used, either $\tau = \operatorname{Tr} U^\top \check{X}_i \check{X}_i^\top U$ which tightens the corresponding constraint (3.18d).

Let $U_Y D_Y V_Y^\top$ be the singular value decomposition of

$$Y = [\sqrt{\gamma_1}\check{X}_1, \sqrt{\gamma_2}\check{X}_2, ..., \sqrt{\gamma_m}\check{X}_m] \in \mathbb{R}^{p \times N}.$$

Observe that $U_Y D_Y^2 U_Y^\top$ is then an eigendecomposition of $YY^\top$. A candidate solution of Problem (3.17) would then be, for fixed γ_i respecting condition (3.18f):

$$M_{ij} = 0 \quad \forall i \neq j \qquad\qquad U = U_Y$$
$$M_{ii} = D_Y^2(i,i) \qquad\qquad \tau = \operatorname{Tr}\left(U^\top Y Y^\top U\right).$$

The last equality results from the combination of conditions (3.18a) and (3.18f):

$$\tau = \sum_i \gamma_i \tau = \sum_i \gamma_i \operatorname{Tr}\left(U^\top X_i X_i^\top U\right).$$

To maximize τ, we should consider the k first singular values of Y. This tells us also that if $\sum_i \gamma_i \operatorname{Tr}\left(U^\top X_i X_i^\top U\right) \neq \max_i \operatorname{Tr}\left(U^\top X_i X_i^\top U\right)$, then U can not be optimal. The difficulty is then to find γ_i such that condition (3.18f) holds.

3.3.2.2 Naive algorithm

Based on the interpretation of KKT conditions, we can build an algorithm to compute a candidate U^*. The main difficulty is to find good γ_i's. Since $\sum_i \gamma_i = 1$, we have that

$$\sum_i \gamma_i d^2(U, X_i) \leq \sum_i \gamma_i \max_j d^2(U, X_j) = \max_j d^2(U, X_j) \tag{3.19}$$

with equality at optimality since condition (3.18f) must hold. We propose an iterative procedure where all conditions except (3.18f) hold at each step, and that aims to enforce the last condition. If we stop when

$$\delta_f = \max_i d(U, X_i) - \sqrt{\sum_i \gamma_i d^2(U, X_i)} \leq \epsilon \tag{3.20}$$

with ϵ small and positive, condition (3.18f) should nearly hold.

Ideally we want both to increase $\sum_i \gamma_i d^2(U, X_i)$ and decrease $\max_i d^2(U, X_i)$ at each iteration t. With $U^{(t)} = U(\gamma^{(t)})$ fixed for iteration t, maximizing the first term is setting to 0 all γ_i's except for those associated with the X_i's at maximal dissimilarity to U. However, taking directly $\gamma_i^{(t+1)} = \alpha$ if $\max_j d(U^{(t)}, X_j) - d(U^{(t)}, X_i) \leq \epsilon$, and 0 otherwise, would eventually ignore the other X_i's and probably not decrease $\max_i d^2(U(\gamma^{(t+1)}), X_i)$. To smooth the γ_i's evolution, we propose to take:

$$\gamma_i^{(t+1)} = \left(\gamma_i^{(t)} + \frac{1}{t+1} r_i^{(t)}\right) \beta \tag{3.21}$$

with $r_i^{(t)} = 1$ if $\max_j d(U^{(t)}, X_j) - d(U^{(t)}, X_i) \leq \epsilon$, $r_i^{(t)} = 0$ otherwise, and β a normalizing constant ensuring that $\sum_i \gamma_i = 1$.

Algorithm 13 enforces conditions (3.18a) to (3.18e) at each iteration and tries to achieve (3.18f) in the limit. The approach is reminiscent of the one proposed in [84] for the minimax problem in $\mathbb{R}^n$, detailed in Section 3.1.2.

We first start by giving all the subspaces the same weight (line 5). The corresponding matrix Y is built, and its k first left singular vectors are computed (lines 7-8). Dissimilarities between U and each subspace are computed to select the points (close to) realizing the maximum (line 11). The weights $\boldsymbol{\gamma}$

are then adapted: the weights γ_i of the farthest points from U are increased. If the error is too large, we iterate the process until the tolerance is met. We took as estimated error the duality gap δ_f in Equation (3.20).

Algorithm 13 Naive KKT approach to compute the minimax center in $\mathcal{G}(k,p)$.

Require: Data X_1, ...,X_m, dimension k, tolerance to maximal dissimilarity ϵ_d, stopping criterion tolerance ϵ

$t \leftarrow 1$
for all X_i **do**
  $\check{X}_i \leftarrow \text{orthonormalBasis}(X_i)$
end for
$\gamma_i^{(t)} \leftarrow \frac{1}{m}$ for all i
$\delta_f \leftarrow 2\epsilon$
while $\delta_f \geq \epsilon$ **do**
  $Y \leftarrow [\sqrt{\gamma_1}\check{X}_1, \sqrt{\gamma_2}\check{X}_2, ..., \sqrt{\gamma_m}\check{X}_m]$
  $U^{(t)} \leftarrow k$ first left singular vectors of Y
  $\delta_f \leftarrow \max_i d(U^{(t)}, X_i) - \sqrt{\sum_i \gamma_i d^2(U, X_i)}$
  $r_i^{(t)} \leftarrow 1$ if $\max_j d(U^{(t)}, X_j) - d(U^{(t)}, X_i) \leq \epsilon_d$, $r_i^{(t)} \leftarrow 0$ otherwise,
  $\gamma_i^{(t+1)} \leftarrow \left(\gamma_i^{(t)} + \frac{1}{t+1} r_i^{(t)}\right) \Big/ \sum_j \left(\gamma_j^{(t)} + \frac{1}{t+1} r_j^{(t)}\right)$
  $t \leftarrow t + 1$
end while

3.3.2.3 Dual method and its convergence

In view of Equation (3.19), we can interpret Algorithm 13 as a dual one. Indeed, we can write the Lagrange dual of Problem (3.17) as [97]:

$$\begin{aligned} &\max_{\gamma} g(\gamma) \\ \text{s.t.} \quad &g(\gamma) = \min_{U^\top U = I} \sum_i \gamma_i d^2(U, X_i) \\ &\gamma_i \leq 0 \quad \forall i = 1, \ldots, m \\ &\sum_i \gamma_i = 1. \end{aligned}$$

If U_γ minimizes $\sum_i \gamma_i d^2(U, X_i)$ for γ fixed, we have that

$$\nabla g(\gamma) = \begin{bmatrix} d^2(U, X_1) & \ldots & d^2(U, X_m) \end{bmatrix} \tag{3.22}$$

is a gradient of $g(\gamma)$. We can then use this gradient to update the γ values.

We know that the optimal value f^* is lower bounded by its dual:

$$\max_{\gamma} \overbrace{\min_{U^\top U = I} \sum_i \gamma_i d^2(U, X_i)}^{f_l(U,\gamma)} \leq f^* \leq \min_{U} \overbrace{\max_i d^2(U, X_i)}^{f_u(U,\gamma)}$$

$$\text{s.t. } \sum_i \gamma_i = 1 \qquad \text{s.t. } U^\top U = I$$

$$\gamma_i \geq 0 \qquad \left(\sum_i \gamma_i \check{X}_i \check{X}_i^\top\right) U = UM.$$

Line 12 increases $f_l(U, \gamma)$ for U fixed (dual step) while lines 8-9 decrease $f_u(U, \gamma)$ for γ fixed (primal step). The difficulty is to update the γ without increasing the corresponding $f_u(U, \gamma)$. This could be fixed by adapting the γ update as proposed in [97], detailed in Algorithm 14. The udpate direction $r^{(t)}$ can be taken as $\nabla g(\lambda)$ as defined in Equation (3.22), or as defined in Equation (3.21).

An example of the behavior of both Algorithms 13 and 14 on data generated using model (3.3) is shown on Figure 3.9. Both approaches converge to the same value, with quite large oscillations in the objective function. However, the interpretation as a dual approach implies associated lower bounds with much smoother evolution. With a smarter choice of step size, Algorithm 14 ensures an increase in the lower bound at each iteration, and a smoother evolution of the associated objective function. We use Algorithm 14 in more extended tests in Section 3.4.

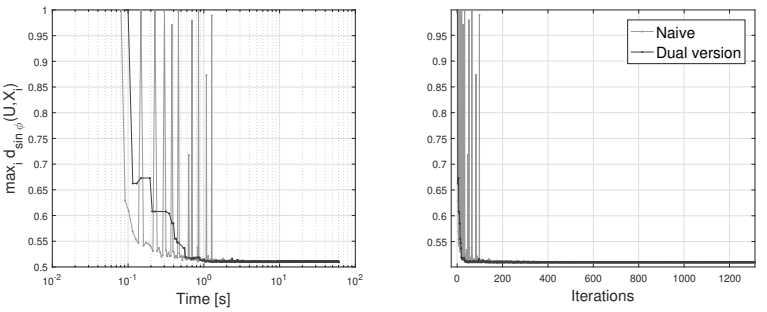


Figure 3.9: Example of the evolution of the objective function using Algorithms 13 and 14 on data generated using model (3.3).

Algorithm 14 Dual adapted KKT approach to compute the minimax center in $\mathcal{G}(k,p)$.

Require: Data X_1, ...,X_m, dimension k, initial step size α, growing parameter β, minimal step size α_{min}, stopping criterion tolerance ϵ

$t \leftarrow 1$
for all X_i **do**
    $\check{X}_i \leftarrow \text{orthonormalBasis}(X_i)$
end for
$\gamma_i^{(t)} \leftarrow \frac{1}{m}$ for all i
$Y \leftarrow [\sqrt{\gamma_1}\check{X}_1, \sqrt{\gamma_2}\check{X}_2, ..., \sqrt{\gamma_m}\check{X}_m]$
$U^{(t)} \leftarrow k$ first left singular vectors of Y
$d \leftarrow \left[d^2(U^{(t)}, X_1), \ldots, d^2(U^{(t)}, X_m)\right]$
$r^{(t)} \leftarrow \text{updateDirection}(d)$
$\delta_f \leftarrow 2\epsilon$
while $\delta_f \geq \epsilon$ **do**
    $t \leftarrow t+1$
    $\gamma^{(t)} \leftarrow \left(\gamma^{(t-1)} + \alpha r^{(t-1)}\right) / ||\gamma^{(t-1)} + \alpha r^{(t-1)}||_1$
    $Y \leftarrow [\sqrt{\gamma_1}\check{X}_1, \sqrt{\gamma_2}\check{X}_2, ..., \sqrt{\gamma_m}\check{X}_m]$
    $U^{(t)} \leftarrow k$ first left singular vectors of Y
    $d \leftarrow \left[d^2(U^{(t)}, X_1), \ldots, d^2(U^{(t)}, X_m)\right]$
    $\delta_f \leftarrow \max_i d(i) - \sqrt{\sum_i \gamma_i d(i)}$
    $r^{(t)} \leftarrow \text{updateDirection}(d)$
    $f_l^{(t)} \leftarrow \sum_i \gamma_i^{(t)d_i}$
    while $f_l^{(t)} < f_l^{(t-1)}$ and $\alpha > \alpha_{min}$ **do**
        $\alpha \leftarrow \alpha/2$
        $\gamma^{(t)} \leftarrow \left(\gamma^{(t-1)} + \alpha r^{(t-1)}\right) / ||\gamma^{(t-1)} + \alpha r^{(t-1)}||_1$
        $Y \leftarrow [\sqrt{\gamma_1}\check{X}_1, \sqrt{\gamma_2}\check{X}_2, ..., \sqrt{\gamma_m}\check{X}_m]$
        $U^{(t)} \leftarrow k$ first left singular vectors of Y
        $d \leftarrow \left[d^2(U^{(t)}, X_1), \ldots, d^2(U^{(t)}, X_m)\right]$
        $r^{(t)} \leftarrow \text{updateDirection}(d)$
        $f_l^{(t)} \leftarrow \sum_i \gamma_i^{(t)d_i}$
    end while
    $\alpha \leftarrow \beta\alpha$
end while

3.3.3 Projected subgradient

We consider here the formulation using $d = d_{\sin\phi}$:

$$\min_U \max_i \left(f_i(U) = k - \operatorname{Tr} U^\top \check{X}_i \check{X}_i^\top U \right) \quad \text{s.t. } U^\top U = I. \tag{3.23}$$

This formulation is not differentiable everywhere, but the points where the objective function gradient cannot be computed are easily identifiable; they are the ones where at least two X_i's realize the maximum dissimilarity from U. The subdifferential is easy to compute at such points. By Equation (3.12) we have

$$\partial f(U) = \operatorname{conv}\left\{\nabla f_i(U) | i \in I_f(U)\right\} = \operatorname{conv}\left\{-2X_i X_i^\top U | i \in I_f(U)\right\},$$

with $I_f(U)$ representing the set of indexes for which $f_i(U) = \max_j f_j(U)$.

To take into account the $U^\top U = I$ constraint, we can use Theorem 3.3.1 that gives sufficient conditions for convergence of a subgradient method in a convex constrained case.

Theorem 3.3.1 (Subgradient method in convex case)**.** *[23, Section 3.2] Let us consider the problem*

$$\min_{x \in D_x} f(x)$$

with D_x a closed convex set, $f : \mathbb{R}^p \mapsto \mathbb{R}$ convex, and updates of the form

$$x^{(t+1)} \leftarrow P_{D_x}\left(x^{(t)} - \alpha^{(t)} g^{(t)}\right)$$

with $g^{(t)}$ any subgradient of f at $x^{(t)}$, $\alpha^{(t)}$ a positive step size and $P_{D_x}(\cdot)$ the projection on the set D_x (with respect to the standard Euclidean norm). If it exists some scalar c such that

$$c \geq \sup\left\{||g^{(t)}|| \ | t = 0, 1, \ldots\right\},$$

and the $\alpha^{(t)}$ satisfies

$$\lim_{t\to\infty} \alpha^{(t)} = 0, \qquad \sum_{t=0}^{\infty} \alpha^{(t)} = \infty, \qquad \sum_{t=0}^{\infty} (\alpha^{(t)})^2 < \infty,$$

then, for D_x nonempty,

$$\lim_{t\to\infty} \inf f(x^{(t)}) = \inf_{x \in D_x} f(x).$$

To apply Theorem 3.3.1, we need to verify the following assumptions:

– D_x is a closed convex set,

- $P_{D_x}(.)$ is a projection onto the set D_x,
- there exists some scalar c such that $c \geq \sup\left\{||g^{(t)}|| \ |t = 0, 1, \dots\right\}$,
- $f(U)$ is convex,

with $D_x = \left\{U|U^\top U = I\right\}$, $g = \sum_i -2w_i X_i X_i^\top U$ for some $w_i \geq 1$ such that $\sum_i w_i = 1$, and $f(U) = \max_i k - \operatorname{Tr} U^\top X_i X_i^\top U$.

Optimization on a closed convex set. We can rewrite Problem (3.23) as an optimization problem on a closed convex set. This is easily done by relaxing the constraint $Y^\top Y = I$ to the closed convex set $Y^\top Y \preceq I$ will not change the optimal solution:

$$\arg\min_{Y^\top Y \preceq I} -\operatorname{Tr}(Y^\top X X^\top Y) = \arg\min_{Y^\top Y = I} -\operatorname{Tr}(Y^\top X X^\top Y).$$

.

Indeed, if $U_Y \Sigma_Y V_Y^\top$ is a k-truncated SVD decomposition Y, we have:

$$\begin{aligned} I - Y^\top Y &\succeq 0 \\ V_Y V_Y^\top - V_Y \Sigma_Y^2 V_Y^\top &\succeq 0 \\ V_Y (I - \Sigma_Y^2) V_Y^\top &\succeq 0 \\ \Sigma_Y^2(i,i) &\leq 1 \ \forall \ i. \end{aligned}$$

Rewriting the objective function gives

$$\begin{aligned} \operatorname{Tr}(Y^\top X X^\top Y) &= \operatorname{Tr}(V_Y \Sigma_Y U_Y^\top X X^\top U_Y \Sigma_Y V_Y^\top) \\ &= \operatorname{Tr}(\Sigma_Y^2 U_Y^\top X X^\top U_Y) \\ &= \sum_l \Sigma_Y^2(l,l) \underbrace{\left(U_Y^\top(:,l) X X^\top U_Y(:,l)\right)}_{\geq 0} \end{aligned}$$

which is maximal if $\Sigma_Y(l,l) = 1$ for all l, that is $Y^\top Y = I$. Observe that the set $\left\{U \in \mathbb{R}^{p \times k} | U^\top U \preceq I_k\right\}$ is the (closed) unit spectral ball, and corresponds to the convex hull of the Stiefel manifold [2].

Projection. The projection of $Z = U - \alpha g$ onto the set

$$D_x = \left\{Y | I - Y^\top Y = 0\right\}$$

can be simply done by putting all singular values of Z to one. This is equivalent to taking an orthonormal basis Q of $\operatorname{span}(U - \alpha g)$:

$$Q^* = \arg\min_{I \succeq Q^\top Q} ||Z - Q|| = P_{D_x}(U - \alpha g)$$

Indeed, let $U_Z D_Z V_Z^\top$ be the SVD of Z. We first show that $D_Z(i,i) \geq 1$ for all i. Then we show that $P_{D_x}(Z) = U_Z \tilde{D}_Z V_Z^\top$ with $\tilde{D}_Z(i,i) = \min(D_Z(i,i), 1)$.

We can write

$$\begin{aligned} Z &= \left(I + 2 \sum_i w_i X_i X_i^\top \right) U = \left(I + AA^\top \right) U \\ Z^\top Z &= U^\top \left(I + AA^\top \right) \left(I + AA^\top \right) U \\ &= I + \underbrace{U^\top AA^\top AA^\top U + 2U^\top AA^\top U}_{B} \end{aligned}$$

with $A = \sqrt{2} \left[\sqrt{w_1} X_1 \quad \dots \quad \sqrt{w_m} X_m \right]$. By Weyl's Theorem [21, Theorem 8.4.11], as I and B are symmetric we have that

$$\lambda_l(I) + \lambda_{min}(B) \leq \lambda_l(I + B) \leq \lambda_l(I) + \lambda_{max}(B)$$

where $\lambda_l(X)$ represents the lth higher eigenvalue of X. As B is semidefinite positive, we have

$$1 \leq 1 + \lambda_{min}(B) \leq \lambda_i(Z^\top Z)$$

and $D_Z[i,i] = \sqrt{\lambda_i(Z^\top Z)} \geq 1$.

With $Q = U_Q D_Q V_Q^\top$ the SVD of Q, $D_Q[l,l] = d_l$ in $]0\ 1]$, v_l and u_l the lth column of V_Q and U_Q:

$$\begin{aligned} \arg \min_{Q^\top Q \preceq I} ||Z - Q||^2 &= \arg \min_{Q^\top Q \preceq I} \operatorname{Tr} Z^\top Z + \operatorname{Tr} Q^\top Q - 2 \operatorname{Tr} Z^\top Q \\ &= \arg \min_{U_Q^\top U_Q = I, V_Q^\top V_Q = I, d_i \in]0\ 1]} \operatorname{Tr} D_Q^2 - 2 \operatorname{Tr} V_Q^\top Z^\top U_Q D_Q \\ &= \arg \max_{U_Q^\top U = I, V_Q^\top V_Q = I, d_i \in]0\ 1]} \sum_l d_l \left(v_l^\top Z^\top u_l - \frac{d_l}{2} \right). \end{aligned}$$

As $d_l \geq 0$, we want to maximize $v_l^\top Z^\top u_l$, and so u_l is a left singular vector of Z, v_l a right singular vector, and $v_l^\top Z^\top u_l = \sigma_l$ the corresponding singular value. The problem becomes

$$\arg \max_{d_i \in]0\ 1]} \sum_l d_l \left(\sigma_l - \frac{d_l}{2} \right)$$

which is maximized with $d_l = \min(\sigma_l, 1)$. So

$$P_{D_x}(Z) = U_Z \tilde{D}_Z V_Z^\top$$

with $U_Z D_Z V_Z^\top$ the SVD of Z, and $\tilde{D}_Z[i,i] = \min(D_Z[i,i], 1)$.

Bounded subgradient. The bound on subgradient is easily checked. We have that

$$\begin{aligned}
||g^{(t)}|| &= || \sum_{i \in I_f(U^{(t)})} -2w_i X_i X_i^\top U^{(t)} || \\
&\leq \sum_{i \in I_f(U^{(t)})} 2w_i ||X_i X_i^\top U^{(t)}|| \qquad = \sum_{i \in I_f(U^{(t)})} 2w_i \sqrt{\sum_{kl} (X_i X_i^\top U^{(t)})_{kl}^2} \\
&\leq \sum_{i \in I_f(U^{(t)})} 2w_i \sqrt{\sum_{kl} (pn_i)^2} \\
&\leq 2pn_i\sqrt{pk}
\end{aligned}$$

as $0 \leq W_i \leq 1$ for all i and $X_i^\top X_i = I_{n_i}$, $U^{(t)T} U^{(t)} = I_k$, so all elements in the X_i's and $U^{(t)}$ are in $[-1, 1]$.

Convexity. Unfortunately, f is not convex. We can however say from Theorem 3.3.2 that using a subgradient approach with appropriate step size within a locally convex region will converge to a local minimum, if the iterates stay in this locally convex region.

Theorem 3.3.2. *[15] Let f be an almost differentiable function with subgradient $g_f(x)$ and let x^* be its local minimum point such that*

$$f(x^*) = \min_{x \in B_r} f(x)$$

with $B_r = \{x : ||x - x^*|| \leq r\}$. Suppose that for any ϵ satisfying $0 < \epsilon < r$ one has

$$\inf_{x \in B_r \setminus B_\epsilon} (g_f(x), x - x^*) > 0.$$

If $||x_0 - x^*|| \leq r$ then the sequence $\left\{x^{(t)}\right\}_{t=0}^{\infty}$ generated by

$$x^{(t+1)} = \begin{cases} \bar{x}^{(t+1)} & \text{if} \quad \bar{x}^{(t+1)} \in B_r, \\ x_0 & \text{if} \quad \bar{x}^{(t+1)} \notin B_r \end{cases}$$

where

$$\bar{x}^{(t+1)} = x^{(t)} - \alpha^{(t)} \frac{g_f(x^{(t)})}{||g_f(x^{(t)})||},$$

$$\alpha^{(t)} > 0, \quad \lim_{t \to \infty} \alpha^{(t)} = 0, \quad \sum_{t=0}^{\infty} \alpha^{(t)} = +\infty,$$

converges to x^* .

3.3.3.1 Algorithm

While the objective function is not convex, we can still propose a subgradient approach based on Theorem 3.3.1, described in Algorithm 15. We initialize with a point in the convex set of $\{X_1, ..., X_m\}$ (lines 5-6). While the error is too large, we determine which points are at maximal dissimilarity from the current iterate, with some tolerance ϵ_d (line 14). From those points we compute a subgradient (line 15), compute the next iterate (line 16) and project it onto the set of orthogonal matrices (lines 17-18). As convex combination to compute the subgradient in line 15 we propose to take the mean of all $\nabla f_i(U)$ for $i \in I_f(U)$:

$$g^{(t)} \leftarrow -\frac{2}{|I_f|} \sum_{i \in I_f} X_i X_i^\top U^{(t)}.$$

Lines 17-18 of Algorithm 15 can be interpreted as a projection on a closed convex set that is the convex hull of Stiefel. When only one X_i is at maximal dissimilarity from the current iterate, the idea is similar to Algorithm 10: the current iterate is updated in the direction of the furthest X_i.

To apply Theorem 3.3.2, we have to ensure that the iterates stay in a locally convex region to avoid a possible switch from one convex region to another. Practically, using Algorithm 15 we observe that the objective function is always decreasing, but slowly (see Section 3.4). An example of the behavior of Algorithm 15 on data generated using model (3.3) is shown on Figure 3.10 for various values of (a, b). We can see that in the case values $(a, b) = (2, 1)$, which give more weight to the subgradient, improve the convergence but also help the apparition of local increases in the objective function (like around $t = 10^0 s$). In more extended tests in Section 3.4 we use Algorithm 15 with $(a, b) = (1, 1)$ to mimic Algorithm 10.

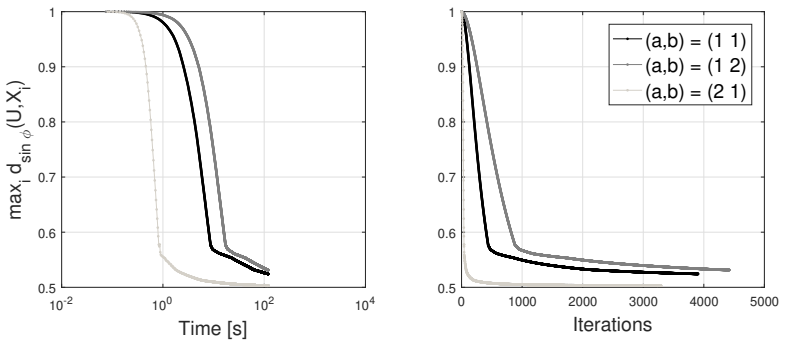


Figure 3.10: Example of the evolution of the objective function using Algorithm 15 on data generated using model (3.3) for various values of (a, b).

Algorithm 15 Subgradient approach to compute the minimax center in $\mathcal{G}(k,p)$.

Require: data X_1, ...,X_m, step size parameters (a,b), tolerance to maximal dissimilarity ϵ_d, stopping criterion tol

for all X_i **do**
  $t \leftarrow 1$
  $\check{X}_i \leftarrow \text{orthonormalBasis}(X_i)$
end for
$Y \leftarrow [\check{X}_1 ... \check{X}_m]$
$U^{(0)} \leftarrow k$ first left singular vectors of Y
for all $\check{X}_j$ **do**
  $d_j^{(0)} \leftarrow \mathrm{d}_{\sin\phi}(U^{(t)}, \check{X}_j)$
end for
$d_{\text{sorted}} \leftarrow \text{sort}_{\text{decreasing}}(d^{(0)})$
$\text{err}^{(0)} \leftarrow d_{\text{sorted}}(1) - d_{\text{sorted}}(2)$
$t \leftarrow 1$
while $\text{err}^{(t-1)} \geq \text{tol}$ **do**
  $I_{\text{f}} \leftarrow \left\{ i | \max_j d_j^{(t)} - d_i^{(t)} \leq \epsilon_d \right\}$
  $g^{(t)} \leftarrow -\text{ConvexCombination}\left(X_i X_i^\top U^{(t)}, i \in I_f \right)$
  $U_{tmp} \leftarrow U^{(t-1)} - \frac{a}{b+t} g^{(t)}$
  $A\Sigma B^\top \leftarrow \text{SVD}(U_{tmp})$
  $U^{(t)} \leftarrow AB^\top$
  for all $\check{X}_j$ **do**
    $d_j^{(t)} \leftarrow \mathrm{d}_{\sin\phi}(U^{(t)}, \check{X}_j)$
  end for
  $d_{\text{sorted}} \leftarrow \text{sort}_{\text{decreasing}}(d^{(t)})$
  $\text{err}^{(t)} \leftarrow d_{\text{sorted}}(1) - d_{\text{sorted}}(2)$
  $t \leftarrow t + 1$
end while

Stationary points. Using Algorithm 15, the iterates do not change anymore if we have $U^{(t+1)} = U^{(t)}$, that is if

$$g^{(t)} = \sum_i -2w_i X_i X_i^\top U^{(t)} = 0$$

$$\Leftrightarrow U^{(t)} \perp X_i \quad \forall i,$$

which corresponds to a maximum, or

$$\text{span}(U^{(t)}) = \text{span}(U^{(t+1)})$$

that is, there exists an invertible matrix B such that

$$U^{(t)}B = U^{(t+1)} = (I - \sum_i -2w_i X_i X_i^\top)U^{(t)}.$$

B is then symmetric. This corresponds to the KKT conditions (3.18) and we are at a critical point.

3.3.4 Descent direction

To avoid the possible intermediate increases of the objective function seen in Figure 3.8, we propose here a feasible direction method using a manifold approach.

Recall from Section 3.1.3.2 that we want to find a subspace $\mathcal{U}$ belonging to $\mathcal{G}(k,p)$, the set of subspaces of dimension k in $\mathbb{R}^p$. This $\mathcal{U}$ should be representative of all the subspaces $\mathcal{X}_i = \text{span}(X_i) \in \mathcal{G}(n_i, p)$, that is we are looking for $\mathcal{U}^*$ such that

$$\mathcal{U}^* = \arg\min_{\mathcal{U}} \underbrace{\max_i d^2(\mathcal{U}, \mathcal{X}_i)}_{:=f(\mathcal{U})} \tag{3.24}$$

where $d(\mathcal{U}, \mathcal{X})$ is a dissimilarity measure between $\mathcal{U}$ and $\mathcal{X}$. We consider the dissimilarity $d_\phi(\mathcal{X}, \mathcal{U}) = \sqrt{\sum \phi_l^2 (\mathcal{U}, \mathcal{X})}$. This dissimilarity reduces to the canonical distance on the Grassmann manifold when $\mathcal{U}$ and $\mathcal{X}$ have the same dimensions.

3.3.4.1 General idea

Let U be the current iterate and $\tilde{U}$ the next one. We want to find $\tilde{U}$ such that, if U is not a minimum, the objective function $f(U) = \max_i d_\phi^2(U, X_i)$ decreases:

$$f(U) - f(\tilde{U}) > 0$$

with $\tilde{U} = \tilde{U}(U, \xi, \alpha)$, where ξ is a descent direction and α an associated step size. To find such (α, ξ) we will use an upper bound of the objective function in the tangent space using Topogonov's theorem (see Theorem 3.2.2). Hence if the

upper bound in tangent space decreases enough, the initial objective function on the manifold decreases too.

As the Grassmann manifold has a lower bound $\kappa = 0$ [154] for sectional curvature, we can use Topogonov's Theorem with $\mathcal{M}_\kappa^2 = \mathcal{M}_0^2 = \mathbb{R}^2$ (Euclidean space) on the hinge made by the geodesics from U to X_i, and from U to $\tilde{U}$ (see Figure 3.11).

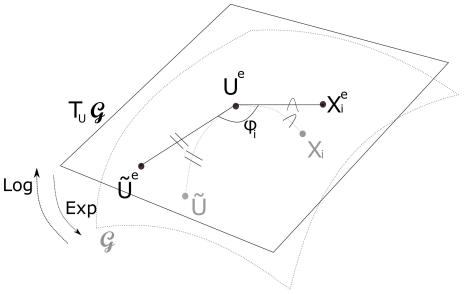


Figure 3.11: Illustration of Topogonov's theorem

We will use the following notation for various objects in the tangent space to the Grassmann manifold at U (which is a Euclidean space):

- $U^e = 0$ the origin of the tangent space in U
- $\tilde{U}^e = \operatorname{Log}_U \tilde{U}$
- $X_i^e = \operatorname{Log}_U X_i$
- $\phi_i = \measuredangle(X_i^e U^e \tilde{U}^e)$ the angle between vectors $U^e X_i^e$ and $U^e \tilde{U}^e$.

By definition of Log, we have $d(U^e, \tilde{U}^e) = d_\phi(U, \tilde{U})$, $d(U^e, X_i^e) = d_\phi(U, X_i)$ and $\measuredangle(X_i^e U^e \tilde{U}^e) = \measuredangle(X_i U \tilde{U}) = \phi_i$. Note that $d(A, B)$ with A, B in Euclidean space represents the usual Euclidean distance $d(A, B) = \sqrt{\operatorname{Tr}(A^\top B)}$. By Topogonov's theorem, we then have $d_\phi(\tilde{U}, X_i) \leq d(\tilde{U}^e, X_i^e)$. So if $d(\tilde{U}^e, X_i^e) \leq d(U^e, X_i^e)$ then $d_\phi(\tilde{U}, X_i) \leq d_\phi(U, X_i)$.

Given an update direction $\xi \in \mathbb{R}^{p \times k}$, $||\xi|| = 1$, such that $\tilde{U} = \operatorname{Exp}_U \alpha\xi$, by Topogonov's theorem and the cosine law, we have the following upper bound:

$$\begin{aligned} d_\phi^2(\tilde{U}, X_i) &\leq d^2(\tilde{U}^e, X_i^e) \\ &= d^2(U^e, \tilde{U}^e) + d^2(U^e, X_i^e) - 2d(U^e, \tilde{U}^e) d(U^e, X_i^e) \cos\phi_i \\ &= d_\phi^2(U, \tilde{U}) + d_\phi^2(U, X_i) - 2d_\phi(U, \tilde{U}) d_\phi(U, X_i) \cos\phi_i \\ &= \alpha^2 ||\xi||_U^2 + ||\operatorname{Log}_U X_i||_U^2 - 2\alpha \langle \operatorname{Log}_U X_i, \xi \rangle_U \end{aligned} \tag{3.25}$$

with d the Euclidean distance, and equality for $\alpha = 0$. If we can find a pair (α, ξ) such that the upper bound (3.25) is strictly less than $d_\phi^2(U, X_i)$, then we

have $d^2_\phi(\tilde{U}, X_i) < d^2_\phi(U, X_i)$. So if

$$(\alpha^*, \xi^*) = \arg \min_{\alpha, ||\xi||=1} \max_i \underbrace{\alpha^2 ||\xi||^2_U + ||\operatorname{Log}_U X_i||^2_U - 2\alpha \langle \operatorname{Log}_U X_i, \xi \rangle_U}_{=h_i(\alpha,\xi)} \quad (3.26)$$

is such that $\alpha^2 ||\xi^*||^2_U + ||\operatorname{Log}_U X_i||^2_U - 2\alpha^* \langle \operatorname{Log}_U X_i, \xi^* \rangle_U < d^2_\phi(U, X_i)$, we can decrease the objective function.

Find step size α. Once ξ is fixed, finding optimal α is quite easy as (3.26) is convex in α. Such a $\alpha > 0$ exists if for all X_f realizing the maximum the tangent in $\alpha = 0$ is negative , i.e. $\frac{\partial h_f}{\partial \alpha}(\alpha = 0, \xi) < 0$. So we need

$$\langle \operatorname{Log}_U X_f, \xi \rangle_U = \operatorname{Tr}\left(\operatorname{Log}_U X_f^\top \xi\right) > 0$$

for all X_f's. If this condition does not hold, we cannot decrease the objective function in direction ξ.

Find descent direction ξ. First, note that since we can move arbitrarily a bit away from the X_j's not realizing the maximum without affecting $f(\tilde{U})$, we do not have to take them into account when computing ξ. So we consider only the points X_f for $f \in I_f = \{j | d_\phi(U, X_j) = \max_i d_\phi(U, X_i)\}$.

From Equation (3.26), since we consider $\alpha \geq 0$, we should have

$$\langle \operatorname{Log}_U X_f, \xi \rangle_U > 0$$

for all $f \in I_f$ to be able to decrease the objective function. This is equivalent to $\cos \phi_f > 0$ for all $f \in I_f$ as $||\xi|| = 1$ and $||\operatorname{Log}_U X_i|| = ||\operatorname{Log}_U X_j||$ for all $i, j \in I_f$. So if we find a hyperplane H passing through U such that all points $\operatorname{Log}_U X_f$ are in the same half-space (excluding H), then taking ξ as the normal to H we have $\cos(\phi_f) > 0$, that is, the normal to H will be a descent direction. Ideally, we want to maximize this quantity, that is

$$\begin{aligned} \xi &= \arg \max_{||\xi||=1} \min_{f \in I_f} \cos \phi_f \; ||\xi|| \; ||\operatorname{Log}_U X_f|| \\ &= \arg \max_H \min_{f \in I_f} d(\operatorname{Log}_U X_f, H) \end{aligned} \quad (3.27)$$

with H the hyperplane passing trough the origin U and with normal ξ (i.e. such that $\langle x, \xi \rangle_U = 0$). By $d(\operatorname{Log}_U X_f, H)$ we mean the distance between point $\operatorname{Log}_U X_f$ and its projection on the hyperplane H. This problem is equivalent to computing the separating hyperplane H^* that maximizes the margin between the origin and the X_f's (see Figure 3.12): this is a special case of the well-known SVM problem (for more details on SVM, see for example [24, Chapter 7]).

Observe that if a separating hyperplane H exists, by construction ξ will be in the cone defined by all $\operatorname{Log}_U X_f$ and can be seen as a (scaled) subgradient

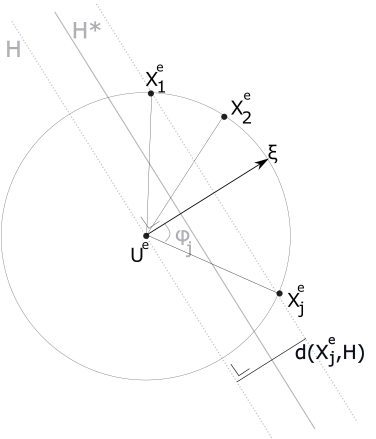


Figure 3.12: Illustration of the equivalence between solving Equation (3.27) and SVM

of f.

Stopping criterion. The nonconvexity of Problem (3.24) makes it harder to solve it globally, so a reasonable objective would be to stop at least close to a critical point. In propositions 3.3.3 and 3.3.4 we give a criterion for a critical point. We then propose in definition 3.3.2 a criterion for a "safely" noncritical point. A point $\mathcal{U}$ is called critical if 0 is a subgradient of f in $\mathcal{U}$.

Proposition 3.3.3. *$\mathcal{U}$ is a critical point of Problem* (3.24) *if*

$$0 \in \operatorname{conv}\left\{X_f^e : d_\phi(U, X_f) = \max_j d_\phi(U, X_j)\right\}$$

with $X_i^e = \operatorname{Log}_U X_i$.

Proof. U is a critical point of f if $0 \in \partial f(U)$ [128, Theorem 10.1]. As $f(U) = \max_i f_i(U)$ with $f_i(U) = d_\phi^2(U, X_i)$, we have [43]

$$\partial f(U) = \operatorname{conv}\{\nabla f_i(U) : f(U) = f_i(U)\}$$

with $\nabla f_i(U) = -2 \operatorname{Log}_U X_i$ [6]. And $0 \in \operatorname{conv}\{-2 \operatorname{Log}_U X_i : f(U) = f_i(U)\}$ is equivalent to $0 \in \operatorname{conv}\left\{X_f^e : d_\phi(U, X_f) = \max_j d_\phi(U, X_j)\right\}$. □

Definition 3.3.1. A hyperplane H with normal h is separating two sets of points A and B if it exists $c > 0$ such that such that $\langle a, h\rangle > c$ and $\langle b, h\rangle < c$ for all $a \in A$ and all $b \in B$.

Proposition 3.3.4. *The point* 0 *belongs to the convex hull of the* X_f^e*'s, that is* $0 \in \text{conv}\left\{X_f^e : d_\phi(U, X_f) = \max_j d_\phi(U, X_j)\right\}$ *iff there does not exist a separating hyperplane between* 0 *and those* X_f^e*.*

Proof. Saying that there exists a separating hyperplane between 0 and the X_f^e's is equivalent to saying that (using the usual SVM formulation) there exist $\xi \in \mathbb{R}^{kp}$, $b \in \mathbb{R}$ such that

$$\xi^\top \text{vec}(X_f^e) - b \geq 1 \ \forall f \tag{3.28}$$

$$\xi^\top \text{vec}(U^e) - b \leq -1. \tag{3.29}$$

where $\text{vec}(X)$ is the vectorization of X. Since we are working in the tangent space $T_U\mathcal{M}$ to U, we have $U^e = 0$. Hence, we readily obtain that (3.28)-(3.29) is equivalent to

$$\xi^\top X^e \geq \begin{pmatrix} 2 \\ \vdots \\ 2 \end{pmatrix} = \text{vec}(2)$$

with $X^e = [\text{vec}(X_{f_1}^e) ... \text{vec}(X_{f_F}^e)] \in \mathbb{R}^{kp \times F}$. We also have that 0 belongs to the convex hull of the X_f^e's iff there exists $\beta \in \mathbb{R}^F$ such that $\beta_j \geq 0$, $\sum \beta_j = 1$ and $X^e\beta = 0$. So proving the proposition is equivalent to proving that

$$\exists \beta \in \mathbb{R}^F \text{ s.t. } \beta_j \geq 0, \sum \beta_j = 1, X^e\beta = 0 \Leftrightarrow \nexists \xi \in \mathbb{R}^{kp} \text{ s.t. } \xi^\top X^e \geq \text{vec}(2)$$

which is easily proved using Gordan's Theorem [21, Fact 4.11.5].

□

We will make use of the following robust version of noncritical points.

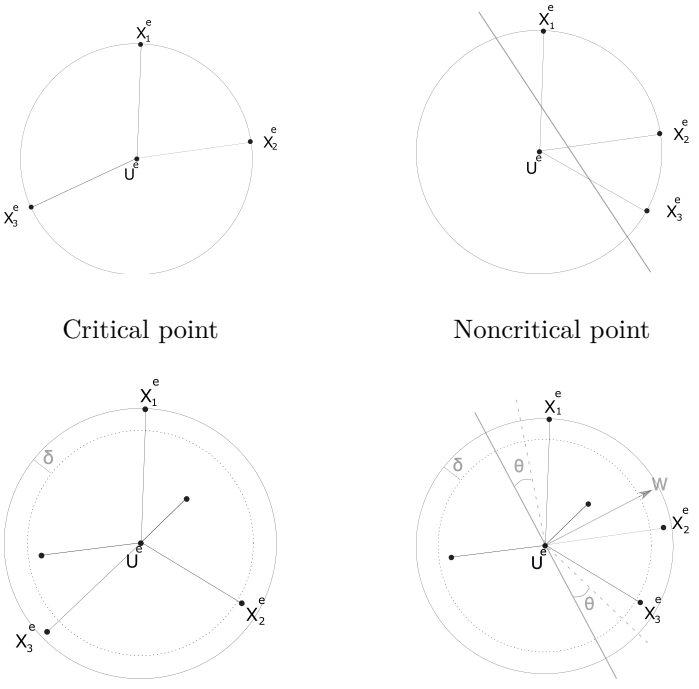
Definition 3.3.2. For some $1 > \delta > 0$ and $\frac{\pi}{2} > \theta > 0$, $\mathcal{U}$ is a (δ, θ) noncritical point if it exists $W \in T_\mathcal{U}\mathcal{G}$ such that all the $\text{Log}_U X_i = X_i^e$ are outside the closed region

$$\mathcal{S}_U = \{\eta \in T_U\mathcal{G} : (1-\delta)R_U \leq ||\eta||_U \leq R_U \text{ and } \measuredangle(W, \eta) \geq \frac{\pi}{2} - \theta\},$$

with $R_u = \max_i d_\phi(U, X_i)$ and $\cos \measuredangle(W, \eta) = \langle W, \eta \rangle \, ||W|| \, ||\eta||$ using the usual Euclidean scalar product.

Illustrations of (non) critical points and their (δ, θ) approximations are shown on Figure 3.13.

Proposition 3.3.5. *Every* (δ, θ) *noncritical point of* (3.24) *is a noncritical point of* (3.24)*. Conversely, every noncritical point of* (3.24) *is a* (δ, θ) *noncritical point of* (3.24) *for some* (δ, θ) *small enough.*



Critical point Noncritical point

(δ, θ) critical point (δ, θ) noncritical point

Figure 3.13: Illustrations of (non) critical points and their (δ, θ) approximations

Proof. The first claim is direct. For the second, let

$$\delta = 1 - \frac{\max_i\{d(U, X_i) < \max_j d(U, X_j)\}}{\max_j d(U, X_j)},$$

$$\theta = \frac{\pi}{2} - \min_{\xi \in T_U\mathcal{G}} \max_i \{\measuredangle(\xi, X_i) : d(U, X_i) = \max_j d(U, X_j)\}.$$

□

Algorithm. We detail in Algorithm 16 the proposed procedure to solve problem (3.24). We first initialize U with an SVD of the concatenation of orthonormal basis of the $\mathcal{X}_i$ (lines 4-5). We then consider the X_f's far enough from U, and compute a hyperplane separating them from $U = 0$ (lines 7-9). This hyperplane, and its normal ξ, can be found using any SVM implementation. We can either use a hard-margin SVM, or a soft-margin and check the θ conditions. If ξ is such that $\measuredangle(\xi, X_i) < \frac{\pi}{2} - \theta$ for all i such that $d(U, X_i) \geq (1-\delta)\max_j d(U, X_j)$, then we update U in the direction ξ. The step size is set as the minimum of the corresponding convex upper bound (lines 11-12).

Algorithm 16 Descent direction approach to compute the minimax center in $\mathcal{G}(k, p)$ as a (δ, θ) critical point of Problem (3.24).

Require: Dimension k, stopping criterion (δ, θ), data $X_1, \ldots, X_m$

1: **for all** X_i **do**
2: $\quad \check{X}_i \leftarrow \text{orthonormalBasis}(X_i)$
3: **end for**
4: $Y \leftarrow \left[\check{X}_1, \ldots, \check{X}_m\right]$
5: $U \leftarrow k$ first left singular vectors of Y
6: **while** U a (δ, θ) noncritical point **do**
7: $\quad I_f \leftarrow \{f | d_\phi(U, X_f) \geq (1-\delta)\max_i d_\phi(U, X_i)\}$
8: $\quad X_i^e \leftarrow \text{Log}_U X_i$ for all i
9: $\quad \xi \leftarrow \text{normalOfSeparatingHyperplane}\left(U, \left\{X_f^e\right\}_{f \in I_f}\right)$
10: $\quad$ **if** $\frac{\langle X_f^e, \xi\rangle}{||X_f^e||\ ||\xi||} > \sin\theta\ \forall\ f \in I_f$ **then**
11: $\quad\quad \alpha^* \leftarrow \arg\min_\alpha \max_i \alpha^2 ||\xi||_U^2 + ||X_i^e||_U^2 - 2\alpha \langle X_i^e, \xi\rangle_U$
12: $\quad\quad U \leftarrow \text{Exp}_U(\alpha^* \xi)$
13: $\quad$ **else**
14: $\quad\quad U^* \leftarrow U$
15: $\quad\quad$ Break
16: $\quad$ **end if**
17: **end while**

In the two-dimensional Euclidean case, our approach reduces to the feasible direction approach of [73], which is equivalent to the Chrystal-Peirce al-

gorithm [36] with a specific choice of direction and which is guaranteed to converge in a finite number of steps. The difficulty in higher dimension is to compute a feasible direction, that we solve here by interpreting the problem as an SVM. In Euclidean space of higher dimension, our approach is analogous to that in [49], where the set of points on the boundary of the ball are our X_i's such that $d(X_i, U) \geq (1-\delta)\max_j d(X_j, U)$. The points to drop are the ones not considered as support vectors. The update direction is then the same as the separating hyperplane normal.

3.3.4.2 Convergence analysis

We now show that our proposed algorithm converges to a (δ, θ) critical point with probability one. For this we first prove in Proposition 3.3.6 that if U is a (δ, θ) noncritical point, Algorithm 16 can construct $\tilde{U}(\alpha, \xi)$ with Algorithm 16 such that $f(U) - f(\tilde{U}(\alpha, \xi)) \geq \epsilon > 0$ where *epsilon* depends on (δ, θ) and linearly on $R^2 = f(U)$. It is shown in Proposition 3.3.8 that any point in a sufficiently small neighborhood of a (δ, θ) noncritical point is a (δ, θ) noncritical point. We can then show in Proposition 3.3.9 that for any (δ, θ) noncritical point there exists a neighborhood where the decrease of the objective function is bounded away from zero. We finally use a proof based on Polak's theorem [115, Section 1.3, Th. 3] to show in Theorem 3.3.10 that with probability one every accumulation point constructed by Algorithm 16 is a (δ, θ) critical point.

Proposition 3.3.6. *Let consider Problem* (3.24)*. For all U not a (δ, θ) critical point, there exist a descent direction ξ and a step size $\alpha > 0$ such that $f(U) - f(\tilde{U}(\alpha, \xi)) \geq \epsilon > 0$ with*

$$\epsilon = R^2 \sin^2\theta \min\left\{(1-\delta)^2, \frac{\delta(2-\delta)}{(\sin\theta+1)^2}\left(1 + \frac{1}{\sin\theta} - \frac{\delta(2-\delta)}{4(1-\delta)^2\sin^2\theta}\right)\right\}$$

for $R = \max_i d_\phi(X_i, U)$. Algorithm 16 realizes such a (ξ, α) pair.

Proof. Let's note $R^2 = \max_j d_\phi^2(U, X_j) = f(U)$. As U is not a (δ, θ) critical point, it exists $W \in T_{\mathcal{U}}\mathcal{G}$ such that $\measuredangle(W, X_i) < \frac{\pi}{2} - \theta$ for all i such that $d(U, X_i) > (1-\delta)R$. We will show that $\xi = W$ is a descent direction: let's take $\mathrm{Exp}_U W$ as the update direction on the Grassmann manifold and define $\tilde{U}(\alpha) = \mathrm{Exp}_U(\alpha W)$ as the new iterate.

For any X_i, let's consider the tangent plane generated by W and $X_i^e = \mathrm{Log}_U X_i$, with the origin in U. As the sectional curvature of the Grassmann manifold is lower bounded by $\kappa = 0$ [154], by Topogonov's theorem we know that

$$d_\phi(X_i, \tilde{U}(t)) \leq d(X_i^e, \tilde{U}^e(\alpha)) = d(X_i^e, \alpha W).$$

By the cosine law, we have

$$d^2(X_i^e, \alpha W) = \alpha^2 ||W||^2 + ||X_i^e||^2 - 2\alpha\cos\phi_i ||W||\ ||X_i^e||$$

with ϕ_i the (acute) angle between W and X_i^e. We want to find α such that

$$f(U) - f(\tilde{U}(\alpha)) = R^2 - \max_i d_\phi^2(X_i, \tilde{U}(\alpha)) \geq \epsilon > 0.$$

That is we want for all i

$$R^2 - d_\phi^2(X_i, \tilde{U}(\alpha)) \geq R^2 - d^2(X_i^e, \alpha W) \geq \epsilon > 0 \tag{3.30}$$

or

$$d^2(X_i^e, \alpha W) = \alpha^2 ||W||^2 + ||X_i^e||^2 - 2\alpha \cos\phi_i ||W||\ ||X_i^e|| \leq R^2 - \epsilon$$

with $\epsilon > 0$.

Without loss of generality let us consider $||W|| = 1$. Depending on X_i we have three possible cases:

A. $||X_i^e|| \geq (1-\delta)R$, and $\cos\phi_i > \sin\theta > 0$ as U is not a (δ, θ) critical point:

$$d^2(X_i^e, \alpha W) < g_a(\alpha) = \alpha^2 + R^2 - 2\alpha \sin\theta (1-\delta)R,$$

B. $||X_i^e|| < (1-\delta)R$ and $\cos\phi_i \geq 0$:

$$d^2(X_i^e, \alpha W) < g_b(\alpha) = \alpha^2 + (1-\delta)^2 R^2,$$

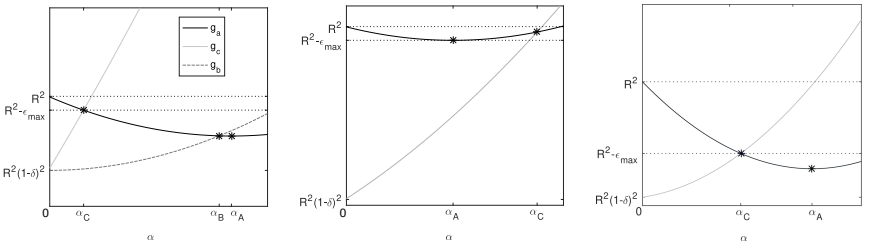
C. $||X_i^e|| < (1-\delta)R$ and $\cos\phi_i < 0$:

$$\begin{aligned} d^2(X_i^e, \alpha W) &< \alpha^2 + (1-\delta)^2 R^2 - 2\alpha \cos\phi_i (1-\delta)R \\ &\leq g_c(\alpha) = \alpha^2 + (1-\delta)^2 R^2 + 2\alpha(1-\delta)R. \end{aligned}$$

All those upper bounds have the same curvature with respectively a negative, null and positive slope in $t = 0$. A possible plot of the three upper bounds is shown on Figure 3.14 (left plot). We now show that it exists $\epsilon > 0$ such that we can find a $\alpha > 0$ such that $d^2(X_i^e, \tilde{U}(\alpha)) < R^2 - \epsilon$.

Maximum possible value for ϵ. To maximize the possible value for such an ϵ, we should find

$$\bar{\alpha} = \arg\min_\alpha \max\left(g_a(\alpha), g_b(\alpha), g_c(\alpha)\right).$$



General behavior of g_a, g_b and g_c | Case $\alpha_A < \alpha_C$ | Case $\alpha_C < \alpha_A$

Figure 3.14: Illustration of the possible cases when computing optimal step size α.

Such a minimum is either at the intersection of g_a with g_c or g_b, either at the minimum of g_a. Let's define those points as

$$\begin{aligned}
\alpha_A &= \min_{\alpha} g_a(\alpha) = R\sin\theta(1-\delta), \\
\alpha_B &= \arg\left(g_a(\alpha) = g_b(\alpha)\right) = \frac{R\delta(2-\delta)}{2\sin\theta(1-\delta)}, \\
\alpha_C &= \arg\left(g_a(\alpha) = g_c(\alpha)\right) = \frac{R\delta(2-\delta)}{2(\sin\theta+1)(1-\delta)}.
\end{aligned}$$

We can easily see that those three points are strictly positive and that $\bar{\alpha} = \min\left(\alpha_A, \alpha_C\right)$ as $\alpha_B \geq \alpha_C$.

If $\alpha_A \leq \alpha_C$ (see Figure 3.14, middle plot), we have

$$\epsilon_{max} = R^2 - g_a(\alpha_A) = \sin^2\theta(1-\delta)^2R^2.$$

If $\alpha_A > \alpha_C$ (see Figure 3.14, right plot), we have

$$\epsilon_{max} = R^2 - g_a(\alpha_C) = \frac{R^2\delta(2-\delta)}{(\sin\theta+1)^2}\left(\sin^2\theta + \sin\theta - \frac{\delta(2-\delta)}{4(1-\delta)^2}\right).$$

This value is strictly positive if $\sin^2\theta + \sin\theta > \frac{1}{4}\left(\frac{\delta(2-\delta)}{(1-\delta)^2}\right)$, which holds as $\alpha_A > \alpha_C$ implies $\sin^2\theta + \sin\theta > \frac{1}{2}\left(\frac{\delta(2-\delta)}{(1-\delta)^2}\right)$.

So taking

$$\bar{\alpha} = \min\left\{\alpha_A, \alpha_C\right\}$$

implies that

$$f(U) - f(\tilde{U}(\bar{\alpha})) \geq R^2 - \max_i d^2(X_i^e, \bar{\alpha}W) = \min_i R^2 - d^2(X_i^e, \bar{\alpha}W) \geq \epsilon_{max}$$

with

$$\epsilon_{max} = R^2 \min\left\{(1-\delta)^2 \sin^2\theta, \frac{\delta(2-\delta)}{(\sin\theta+1)^2}\left(\sin^2\theta + \sin\theta - \frac{\delta(2-\delta)}{4(1-\delta)^2}\right)\right\}$$

which is strictly positive. Finally, Algorithm 16 choses

$$\alpha^* = \arg\min_\alpha \max_i d^2(X_i^e, \alpha W),$$

hence it finds U^* such that

$$\begin{aligned} f(U) - f(U^*) &\geq R^2 - \min_\alpha \max_i d^2(X_i^e, \alpha W) \\ &\geq R^2 - \max_i d^2(X_i^e, \bar{\alpha}W) \\ &\geq \epsilon_{max} > 0. \end{aligned}$$

□

We will now prove in Proposition 3.3.8 that any point close enough to a (δ, θ) noncritical point is also a (δ, θ) noncritical point using a continuity argument: all points outside the closed set $C_{\bar{U}}(\delta, \theta)$ will stay outside the closed set $C_U(\delta, \theta)$ on Figure 3.15. For that we will need Proposition 3.3.7 that gives the necessary conditions to have continuity of $\mathrm{Log}_U X$ at U.

Proposition 3.3.7. *Let $\mathcal{M}$ be a complete (connected) Riemannian manifold. Then $\mathrm{Log}_U X$ is continuous at U as long as X is not in the cut locus of U.*

Proof. As $f : x \mapsto \frac{1}{2}d^2(x, y)$ is $\mathcal{C}^\infty$ on $\mathcal{M} \setminus \mathrm{Cut}(y)$ [6] and $\nabla f = -\mathrm{Log}_x y$ [6] (or [133, Chapter III, Proposition 4.8]), $\mathrm{Log}_U X$ is continuous at U in X as long as X is not in the cut locus of U. □

Proposition 3.3.8. *For all $\bar{U}$ a (δ, θ) noncritical point and such that $X_i \notin \mathrm{Cut}(\bar{U})$ for all i, there exists $0 < \rho$ such that all U in $B_\rho(\bar{U})$ is a (δ, θ) noncritical point, where $B_\rho(\bar{U}) = \left\{U : d_\phi(U, \bar{U}) \leq \rho\right\}$.*

Proof. Let denote $R_A = \max_i d_\phi(X_i, A)$. Since $\bar{U}$ is a (δ, θ) noncritical point, there exists a closed region

$$\mathcal{S}_{\bar{U}} = \{\eta \in T_{\bar{U}}\mathcal{G} : (1-\delta)R_{\bar{U}} \leq ||\eta||_{\bar{U}} \leq R_{\bar{U}} \text{ and } \measuredangle(W_{\bar{U}}, \eta) \geq \frac{\pi}{2} - \theta\}$$

such that $X_i^e = \mathrm{Log}_{\bar{U}} X_i \notin \mathcal{S}_{\bar{U}}$ for all i. We will prove by contradiction that there exists a neighbourhood $B_\rho(\bar{U}) \in \mathcal{G}$ such that, for all $U \in B_\rho(\bar{U})$, $\mathrm{Log}_U X_i \notin P^{U \leftarrow \bar{U}} \frac{R_U}{R_{\bar{U}}} \mathcal{S}_{\bar{U}}$ for all i, where $P^{U \leftarrow \bar{U}}$ denotes the parallel transport

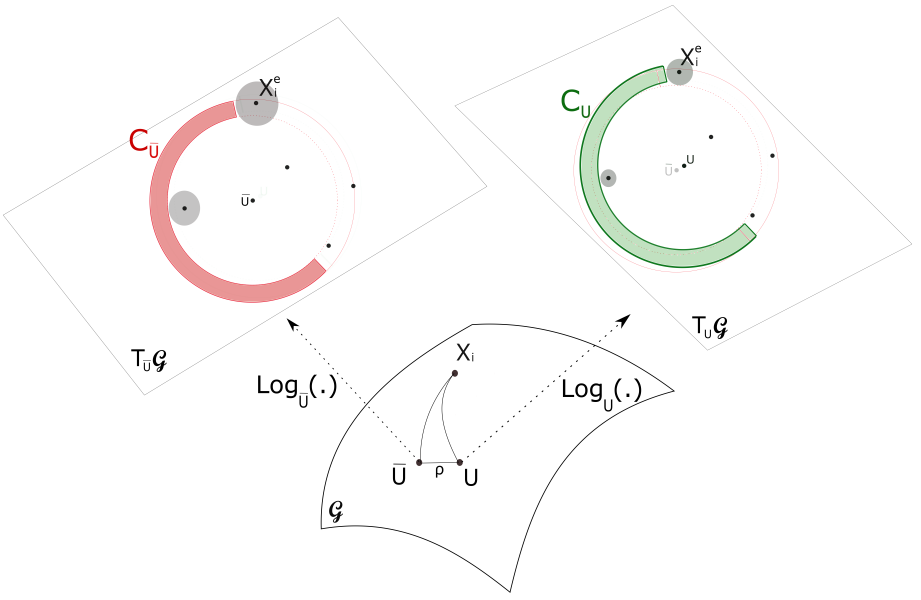


Figure 3.15: Continuity of the closed set.

along the geodesic from $\bar{U}$ to U. Since $P^{U\leftarrow\bar{U}}$ is an isometry, it follows that $P^{U\leftarrow\bar{U}}\frac{R_U}{R_{\bar{U}}}\mathcal{S}_{\bar{U}}$ is a suitable regions for Definition 3.3.2 and then that U is a (δ,θ) noncritical point.

Now, by contradiction, we suppose that there is a sequence (U_j) such that

- $\lim_{j\to\infty} U_j = \bar{U}$ and
- It exists X_{i_j} such that $\mathrm{Log}_{U_j} X_{i_j} \in P^{U_j\leftarrow\bar{U}}\frac{R_{U_j}}{R_{\bar{U}}}\mathcal{S}_{\bar{U}}$.

Restricting to a subsequence if necessary, we have $\mathrm{Log}_{U_j} X_i \in P^{U_j\leftarrow\bar{U}}\frac{R_{U_j}}{R_{\bar{U}}}\mathcal{S}_{\bar{U}}$ for a fixed i and for all j. Since $\mathrm{Log}_{U_j} X_i \in P^{U_j\leftarrow\bar{U}}\frac{R_{U_j}}{R_{\bar{U}}}\mathcal{S}_{\bar{U}}$, there is $\eta_j \in \frac{R_{U_j}}{R_{\bar{U}}}\mathcal{S}_{\bar{U}}$ such that $\mathrm{Log}_{U_j} X_i = P^{U_j\leftarrow\bar{U}}\eta_j$. Further restricting to a subsequence if necessary, we have that η_j converges to some η_*, which belongs to $\mathcal{S}_{\bar{U}}$ by compactness of $\mathcal{S}_{\bar{U}}$. Hence $\lim_{j\to\infty}\mathrm{Log}_{U_j} X_i = P^{\bar{U}\leftarrow\bar{U}}\eta_* = \eta_* \in \mathcal{S}_{\bar{U}}$. On the other hand, by continuity of $U \mapsto \mathrm{Log}_U X_i$ at $\bar{U}$, we have that

$$\lim_{j\to\infty}\mathrm{Log}_{U_j} X_i = \mathrm{Log}_{\bar{U}} X_i \notin \mathcal{S}_{\bar{U}},$$

a contradiction.

□

Proposition 3.3.9. *For all $\bar{U}$ a (δ, θ) noncritical point and such that $X_i \notin \mathrm{Cut}(\bar{U})$ for all i, there exist $\rho > 0$ and $\epsilon > 0$ such that*

$$f(U) - f(\tilde{U}) \geq \epsilon > 0$$

for all U in $B_\rho(\bar{U})$ and all $\tilde{U}$ generated from U by lines 7-12 from Algorithm 16, where $B_\rho(\bar{U}) = \{U : ||U - \bar{U}|| \leq \rho\}$.

Proof. It suffices to combine Propositions 3.3.6 and 3.3.8, and the fact that the cut locus on the Grassmann manifold is closed (this follows from Definition 3.2.16). □

Theorem 3.3.10. *For a set of points $S = \{X_1, \ldots, X_m\}$, every accumulation point of the sequence $\{U_q\}_{q\geq 1}$ constructed by Algorithm 16 is a (δ, θ) critical point of f, or it belongs to the zero measure set*
$\{U \in \mathcal{G}(k,p) : \ \exists i : U \in \mathrm{Cut}(X_i)\}$.

Proof. The proof is an adaptation of Theorem 5.4 in [71], based on Polak's theorem [115, Section 1.3, Th. 3].

Suppose $\bar{U}$ is an accumulation point of Algorithm 16. Then there exist a subsequence of $\{U_q\}_{q\geq 1}$ converging to $\bar{U}$, i.e., $\{U_q\}_{q\in R}$, where R is the index set of the subsequence. Suppose that $\bar{U}$ is not a (δ, θ) critical point of f and that, for all i, $\bar{U} \notin \mathrm{Cut}(X_i)$. By Proposition 3.3.9 there exist $0 < \rho$ and $0 < \epsilon$ such that

$$f(U_r) - f(U_{r+1}) \geq \epsilon > 0,$$

for all r such that $||\bar{U} - U_r|| \leq \rho$. Since $\{U_q\}_{q\in R}$ converges to $\bar{U}$, there exists a $t \in R$ such that for all $q \geq t$, $q \in R$,

$$||\bar{U} - U_q|| \leq \rho.$$

Thus for any two consecutive points U_q, U_{q+s} of the subsequence, with q, $q + s \in R$; $q \geq t$ we have

$$f(U_q) - f(U_{q+s}) \geq f(U_q) - f(U_{q+1}) \geq \epsilon > 0.$$

Thus, the sequence $\{f(U_q)\}_{q\in R}$ is not a Cauchy sequence so it does not converge. On the other hand, since f is continuous and $\{U_q\}_{q\in R}$ converges to $\bar{U}$, $\{f(U_q)\}_{q\in R}$ should converge to $f(\bar{U})$. This is a contradiction, which proves the theorem. The fact that $\{U \in \mathcal{G}(k,p) : \ \exists i : U \in \mathrm{Cut}(X_i)\}$ has zero measure follows from Definition 3.2.16. □

Note that the previous convergence analysis holds for any complete Riemannian manifold with a lower bound of 0 for its sectional curvature and with a closed cut locus.

3.3.4.3 Extension to X_i's of different dimensions

In the Grassmannian case, if the X_i are of dimension greater than k, the whole analysis can be easily extended using projections of X_i's on $\mathcal{G}(k,p)$ instead of the X_i's themselves, using Proposition 3.1.1.

In Propositions 3.3.6 and 3.3.8, consider instead

$$X_i^e = \operatorname{Log}_U X_i B_i \tag{3.31}$$

with $A_i D_i B_i^\top$ a thin SVD of $U^\top X_i$: by Proposition 3.1.1 we have

$$d_\phi(X_i, \tilde{U}(\alpha)) \leq d_\phi(X_i B_i, \tilde{U}(\alpha)) \qquad \text{and} \qquad d_\phi(X_i B_i, U) = d_\phi(X_i, U).$$

Applying Topogonov's theorem we have that $d_\phi(X_i B_i, \tilde{U}(\alpha)) \leq d(X_i^e, \alpha W)$.

To adapt Proposition 3.3.8, we need continuity of $U \mapsto \operatorname{Log}_U X_i B_i$ with $B_i = B_i(U)$ at $\bar{U}$. That is, we need continuity of $\operatorname{span}(X_i B_i)$ in U as the Log operator does not depend on the specific representation of the subspace. It can be seen as $P_{X_i}(U)$, the projection of $[U\ 0_{p\times(n_i-k)}]$ on $\mathcal{X}_i$, which can be computed as $P_{X_i}(U) = X_i X_i^\top U$. Indeed we can easily verify that $\operatorname{span}(X_i B_i) = \operatorname{span}(X_i X_i^\top U)$ that is, it exists $C \in \mathbb{R}^{k\times k}$ invertible such that

$$\begin{aligned} X_i B_i &= X_i X_i^\top U C \\ X_i B_i &= X_i B_i D_i A_i^\top C \\ C &= A_i D_i^{-1}. \end{aligned}$$

Matrix C is invertible by definition of A_i and D_i as long as $U^\top X_i$ is of full rank, which implies that the kth largest principal angle between $\mathcal{U}$ and $\mathcal{X}$ should be smaller than $\frac{\pi}{2}$. As $X_i X_i^\top U$ varies continuously with U, we have continuity of $\operatorname{span}(X_i B_i)$.

The U computed using a modified version of Algorithm 16, where line 8 is replaced by $X_i^e \leftarrow \operatorname{Log}_U X_i B_i$ as defined in Equation (3.31), is then also a (δ, θ) critical point.

3.3.4.4 Variants for the descent direction and step size computation

Descent direction. Computing the $X_i^e = \operatorname{Log}_U X_i$'s for all the X_i's on the border is necessary to compute a descent direction ξ. In Algorithm 16, we consider as 'on the border' all X_i's at dissimilarity greater or equal than $(1-\delta)\max_i d(U, X_i)$. Considering this set of points ensures to find a descent direction if we are not at (δ, θ) critical point. However, incorporating more information coming from other points, that are not on the border yet but close to and so will influence the descent direction ξ in future iterates, could help to find a better ξ. This can be done using I_f defined as

$$I_f = \left\{ f | d_\phi(U, X_f) \geq (1-\delta_0) \max_i d_\phi(U, X_i) \right\}$$

with $\delta_0 \geq \delta$. If no descent direction can be found with such a δ_0, its value is decreased until a descent direction exists or $\delta_0 = \delta$.

Step size. In the step size computation $\alpha^* = \arg\min_\alpha \max_i d^2(X_i^e, \alpha W)$, we can avoid computing all the logarithms for the points not on the border by using a looser upper bound on α obtained from the triangle inequality. For X_i such that $d(U, X_i) < (1 - \delta_i)R$ we have:

$$\begin{aligned} d(\alpha W, X_i^e) &\leq d(\alpha W, U) + d(U, X_i^e) \\ &\leq \alpha + (1 - \delta_i)R. \end{aligned}$$

Decreasing ϵ as in Equation (3.30) leads to the bound

$$\alpha \leq \sqrt{R^2 - \epsilon} - R + R\delta_i \tag{3.32}$$

with $\epsilon < R^2\delta_i(2 - \delta_i)$, which does not require the computation of the $\mathrm{Log}_U X_i$.

So if we consider two sets of points I_1 and I_2 defined carefully, we can take $\alpha^* = \min\{\alpha_f, \min_{j \in I_2} \alpha_j\}$ with $\alpha_f = \arg\min_\alpha \max_{i \in I_1} d^2(X_i^e, \alpha W)$ and α_j as in (3.32), the algorithm will converge to critical point. For example, we can choose $I_2 = \{i | d(U, X_i) < c \max_j d(U, X_j)\}$ with $c = 0.9$ and $I_1 = \{1, \ldots, m\} \setminus I_2$. Of course we should have $c \leq (1 - \delta)$. The trade-off is then to choose between a large I_1 which requires to compute a large number of logarithms but gives a good t, and a large I_2 which is cheaper to compute but can imply a smaller α and so a smaller decrease of the objective function.

Algorithm. An implementation using these variants is described in Algorithm 17. Lines 12 to 15 allow starting with $\delta_0 > \delta$ to take into account more points in the computation of the update direction ξ. If such an update does not exist then the current δ_i is decreased until reaching the limit value δ. The optimal step size is computed using the limited number of points in I_{f2} (line 18) and then corrected if necessary (line 19).

An example of the behavior of Algorithm 17 on data generated using model (3.3) is shown in Figure 3.16 when varying δ_0 and in Figure 3.17 when varying c. All approaches converge to the same value. Increasing δ_0 appears to accelerate slightly the decrease in the objective function. Choosing a larger c implies a faster decrease. The expense of taking into account more X_i's in the computation of the step size is compensated by a greater decrease in the objective function. We use Algorithm 17 with $\delta_0 = 10^{-2}$ and $c = 0$ in more extended tests in Section 3.4.

Algorithm 17 Improved descent direction approach to compute the minimax center in $\mathcal{G}(k,p)$ as a (δ,θ) critical point of Problem (3.24).

Require: Dimension k, stopping criterion $(\delta,\ \theta)$, data $X_1,\ldots,X_m$, parameter $0 \leq c < 1$ fixing the number of X_i used to compute the step size, initial δ_0 with $\delta \leq \delta_0 \leq 1-c$,

for all X_i **do**
  $\check{X}_i \leftarrow \text{orthonormalBasis}(X_i)$
end for
$Y \leftarrow \left[\check{X}_1,\ldots,\check{X}_m\right]$
$U \leftarrow k$ first left singular vectors of Y
while U a (δ,θ) noncritical point **do**
  $\tilde{\delta} \leftarrow \delta_0$
  $I_f \leftarrow \left\{f | d_\phi(U,X_f) \geq (1-\tilde{\delta})\max_i d_\phi(U,X_i)\right\}$
  $I_{f2} \leftarrow \{f | d_\phi(U,X_f) \geq c\max_i d_\phi(U,X_i)\}$ (on a donc $I_f \subset I_{f2}$)
  $X^e_{f2} \leftarrow \text{Log}_U X_f$ for all $f \in I_{f_2}$
  $\xi \leftarrow \text{normalOfSeparatingHyperplane}\left(U, \left\{X^e_f\right\}_{f\in I_f}\right)$
  while exists X_f, $f \in I_f$ s.t. $\left\langle X^e_f, \xi\right\rangle < 0$ and $\tilde{\delta} > \delta$ **do**
    $\tilde{\delta} \leftarrow \tilde{\delta}/2$
    $I_f \leftarrow \left\{f | d_\phi(U,X_f) \geq (1-\tilde{\delta})\max_i d_\phi(U,X_i)\right\}$
    $\xi \leftarrow \text{normalOfSeparatingHyperplane}\left(U, \left\{X^e_f\right\}_{f\in I_f}\right)$
  end while
  if $\frac{\left\langle X^e_f,\xi\right\rangle}{||X^e_f||\ ||\xi||} > \sin\theta\ \forall\ f \in I_f$ **then**
    $\alpha_{tmp} \leftarrow \arg\min_\alpha \max_{i\in I_{f2}} t^2||\xi||^2_U + ||X^e_i||^2_U - 2\alpha\left\langle X^e_i,\xi\right\rangle_U$
    $\alpha \leftarrow \min(\alpha_{tmp}, (1-c)\max_i d_\phi(U,X_i))$
    $U \leftarrow \text{Exp}_U(\alpha\xi)$
  else
    $U^* \leftarrow U$
    Break
  end if
end while

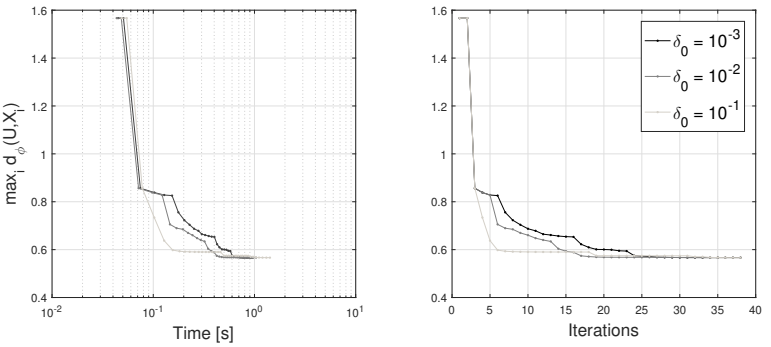


Figure 3.16: Example of the evolution of the objective function using Algorithm 17 for $c = 1$ and varying δ_0 on data generated using model (3.3).

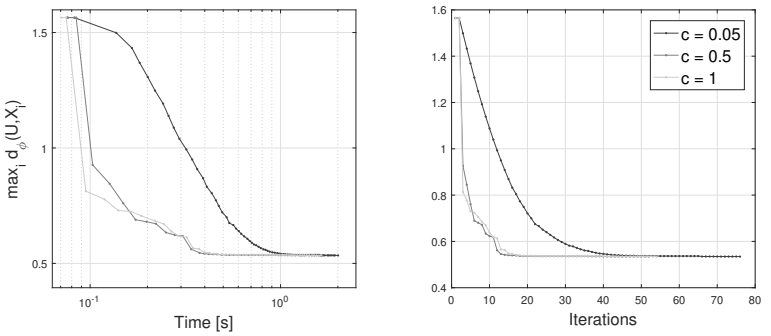


Figure 3.17: Example of the evolution of the objective function using Algorithm 17 for $\delta_0 = 10^{-2}$ and varying c on data generated using model (3.3).

3.3.5 LogExp smoothing

An usual way to deal with nonsmooth function is to approximate it with a smooth function. For a max function, we can use the LogSumExp (LSE) function:

$$LSE(x_1, ..., x_n) = \log\left(\sum_{i=1}^{n} e^{x_i}\right).$$

A smooth approximation of $f(U, X_1, ..., X_m) = \max_i d^2(U, X_i)$ is then given by

$$f_\mu(U, X_1, ..., X_m) = \mu \log\left(\sum_i e^{\frac{d^2(U,X_i)}{\mu}}\right) \tag{3.33}$$

with parameter $\mu > 0$. We have

$$f_\mu(U, X_1, ..., X_m) - \mu \log m \leq \max_i d^2(U, X_i) \leq f_\mu(U, X_1, ..., X_m)$$

as $\mu \log\left(\sum_i e^{\frac{d^2(U,X_i)}{\mu}}\right) \leq \mu \log\left(m \max_i e^{\frac{d^2(U,X_i)}{\mu}}\right)$: a smaller μ gives a better approximation of f.

The corresponding Euclidean gradient is

$$\frac{\partial f_\mu}{\partial U} = -\frac{2}{\sum_b e^{\frac{d^2(U,X_b)}{\mu}}}\left(\sum_a e^{\frac{d^2(U,X_a)}{\mu}} X_a X_a^\top\right) U. \tag{3.34}$$

Any Riemannian method applicable to smooth objective functions can then be applied, such as the ones proposed in the ROPTLIB [66] or Manopt [25] toolboxes. We used codes from Manopt, which proposes "a large library of manifolds and ready-to-use Riemannian optimization algorithms" but only for smooth cost functions.

We provided the cost function and the associated gradient to Manopt with a choice of solver. If the μ value is small enough, the obtained solution should be close enough to a critical point. We took the default trust region implementation of Manopt, starting with $\mu = 10e - 1$. We then re-run the trust region with decreasing $\mu = 10e[-2, \ldots, -7]$ each time initializing U with the solution of the previous approximation. The corresponding algorithm is described in Algorithm 18.

Algorithm 18 LogExp smoothing on Stiefel approach to compute the minimax center in $\mathcal{G}(k,p)$.

Require: Dimension k, initial μ_0, stopping criterion μ, data $X_1, \ldots, X_m$
for all X_i **do**
  $\check{X}_i \leftarrow$ orthonormalBasis(X_i)
end for
$Y \leftarrow \left[\check{X}_1, \ldots, \check{X}_m\right]$
$U \leftarrow k$ first left singular vectors of Y
$\mu_i \leftarrow \mu_0$
while $\mu_i \geq \mu$ **do**
  $U \leftarrow$ ManoptTrustRegion $\left(\min_{U \in \mathrm{St}(k,p)} f_\mu(U, X_1, ..., X_m), U_0 = U\right)$
  $\mu_i \leftarrow \mu_i/10$
end while

The use of exponential function in the cost function and gradient can lead to computer overflow. We then use the trick of multiplying the exponential at denominator and numerator in (3.34) by $e^{-\frac{d^2_{max}}{\mu}}$ with $d_{max} = \max_a d(U, X_a)$. The cost function (3.33) can also be rewritten as

$$f_\mu(U, X_1, ..., X_m) = \mu \log\left(\sum_i e^{\frac{d^2(U,X_i) - d^2_{max}}{\mu}}\right) + d^2_{max}.$$

An example of the behavior of Algorithm 18 on data generated using model (3.3) is shown on Figure 3.18. One star $*$ corresponds to an iteration of the algorithm while the dots correspond to the iterations of the trust region. An example of the evolution of the radius for the corresponding trust region is shown on Figure 3.19. Quite logically, initial larger values of the radius come with greater decreases of the objective function. The radius globally decreases with iterations: getting closer to a critical point and using an improving accuracy of the approximation implies that the local model has to represent a sharper transition between two X_i's at maximal dissimilarity. We use Algorithm 18 with $\mu_0 = 10^{-1}$, $\mu = 10^{-7}$ in more extended tests in Section 3.4.

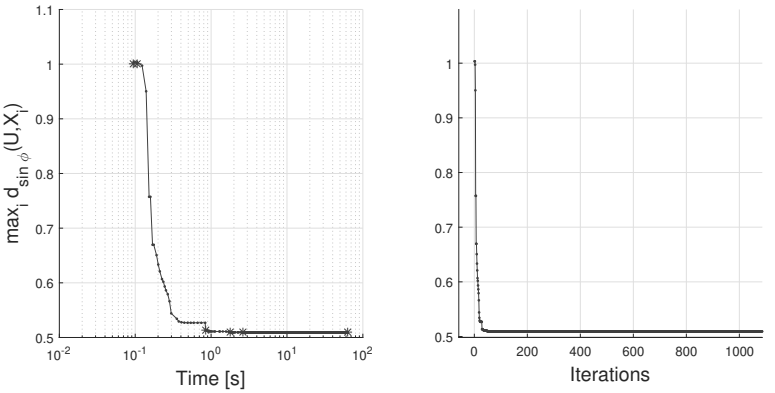


Figure 3.18: Example of the evolution of the objective function using Algorithm 18 on data generated using model (3.3). A star $*$ corresponds to an iteration of the algorithm (new μ value) while the dots correspond to the iterations of the trust region (TR).

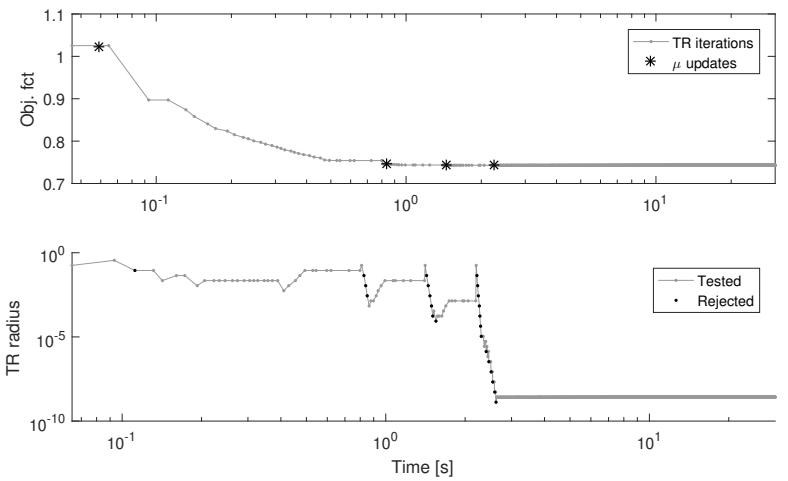


Figure 3.19: Example of the evolution of the objective function using Algorithm 18 on data generated using model (3.3), and the corresponding evolution of the radius of the trust region. The black dots indicate a radius rejected and decreased at the next iteration.

3.4 Experiments

In this section we compare the performances of the proposed algorithms:

- *Greedy*: at each iteration, move in the direction of the furthest point with a decreasing step size (Section 3.3.1, Algorithm 10).

- *KKT*: inspired from KKT conditions, try to find a convex combination γ of the points such that $\sum_i \gamma_i d^2(U, X_i) = \max_j d^2(U, X_j)$ (Section 3.3.2, Algorithm 14 with $\alpha = 1$, $\beta = 1.5$, $\alpha_{min} = 10^{-10}$).

- *Proj.SG*: at each iteration, take as update direction the mean of all subgradient associated to a point at maximal dissimilarity, then project back on Stiefel (Section 3.3.3, Algorithm 15 with $(a, b) = (1, 1)$).

- *Descent*: at each iteration, compute the update direction and the step size that will decrease the most an upper bound of the objective function (Section 3.3.4, Algorithm 17 with $(\delta, \theta) = (10^{-3}, 10^{-3})$, $c = 0$, $\delta_0 = 10^{-2}$).

- *LogExp*: minimize a smooth version of the objective function on Stiefel using a trust region method in Manopt (Section 3.3.5, Algorithm 18 with $\mu_0 = 10^{-1}$, $\mu = 10^{-7}$).

We first test the algorithms on synthetic data in Section 3.4.1. We then apply our minimax model on gene expression datasets in Section 3.4.2 and on a parametric model order reduction problem in Section 3.4.3.

3.4.1 Synthetic data

To study the behavior of the algorithms, we generated datasets using the following two-step procedure. We first generated a reference subspace U_{cA} in $\mathcal{G}(k_A, p)$. From U_{cA} we generated three other subspaces U_1, U_2 and U_3 each at a distance $R < R_{max} = \sqrt{k_A}\frac{\pi}{2}$ of U_{cA}. Then from each of those the U_j's we generated m_j subspaces Y_i as noisy versions of U_j (left side of Figure 3.20). The minimax center of those Y_i's should then be close to U_{cA} by construction. We then added new dimensions to those Y_i's. For this we generated another subspace U_{cB} in $\mathcal{G}(k_B, p)$, orthogonal to U_{cA}. From U_{cB} we generated $m_1 + m_2$ noisy versions W_i's, and the remaining m_3 subspaces are generated to be close to orthogonality to U_{cB}, i.e. at a distance close to $R_{max} = \sqrt{k_B}\frac{\pi}{2}$ of U_{cB} (right side of Figure 3.20). Final subspaces are generated as $X_i = \text{span}(Y_i, W_i)$. We took $p = 1000$, $k_A = 2$, $k_B = 2$ and $[m_1, m_2, m_3] = [6, 2, 2]$.

We compared the algorithms presented in this chapter, and stopped them after maximum 30 seconds or sooner if the stopping criterion was reached. We applied all algorithms to both dissimilarities and if an algorithm was designed for one of the dissimilarities it is suitably modified when applied to the other (see Table 3.3 for more details). We also compared to the subspace U obtained

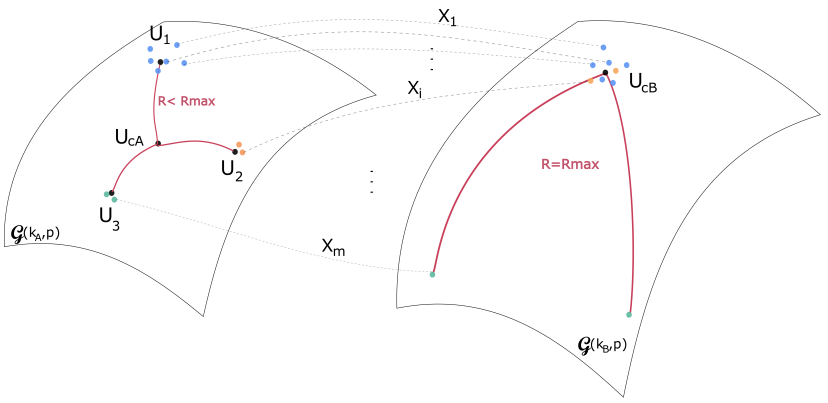


Figure 3.20: Illustration of the process generating the test data.

Algorithm	Alg. nr.	Diss.	Stop. crit.	Adaptation to other diss.
Greedy	10	d_ϕ	maxtime	Use of $d_{\sin\phi}$ to compute obj. fct
KKT	14	$d_{\sin\phi}$	$\delta_f \leq 10^{-3}$	Use of d_ϕ to compute obj. fct + update direction
Proj.SG	15	$d_{\sin\phi}$	maxtime	Use of d_ϕ to compute obj. fct + I_f
Descent	17	d_ϕ	$\delta \leq 10^{-3}$, $\theta \leq 10^{-3}$	Use of $d_{\sin\phi}$ to compute obj. fct + I_f
LogExp	18	$d_{\sin\phi}$	$\mu \leq 10^{-7}$	Use of d_ϕ to compute obj. fct

Table 3.3: Algorithms compared.

when applying an SVD on all orthonormal basis concatenated:

$$U \leftarrow \text{k first left singular vectors}([\check{X}_1, \ldots, \check{X}_m]) \tag{3.35}$$

with $\check{Y}$ an orthonormal basis of span(Y). This approach, denoted SVD_o, will select a subspace U such that [45]

$$U = \arg\min_U \sum_i d^2_{\sin\phi}(U, X_i).$$

SVD_o can then be seen as computing the Grassmannian l_2 center of mass, while minimax computes the l_∞ center.

Case 1: $R < \frac{\text{inj}(\mathcal{M})}{2}$**.** We took R uniformly in $\frac{\pi}{4}[0.7\ 0.95]$, and the noisy versions at distances $r_1 \sim U_{[0\ 0.2]*R}$ and $r_2 \sim U_{[0\ 0.5]*R}$ of their clean counterpart. This gives an estimate of the optimal objective value in $R(1 \pm 0.2)$, so (nearly) all X_i's are in the ball $B(U_{cA}, \frac{\pi}{4})$.

Figure 3.21 shows the mean evolution of the objective functions for the different methods, for both dissimilarities. The objective function is scaled by R, an estimate of the optimal objective value. The final values should then

be close to one to be optimal. We can see that in this easy configuration, all methods converges to the same objective value. None of the implementations of the algorithms were optimized for computational efficiency, however we can see that the *Greedy* method decreases faster (in terms of time) than the others. The flat part at the beginning shared by all methods corresponds to the initialization step. An example of the evolution of the objective functions for the various approaches is shown in Figure 3.22. We can see that the *Greedy* approach tends to oscillate and the KKT approach does not always find a step size decreasing the objective function at each iteration. In this example, in the d_ϕ case, the KKT method does not converge which explains the large standard deviation in Figure 3.21.

In Figure 3.23 we can see the mean dissimilarities between the final solutions provided by the various algorithms and the reference subspace U_{cA} used to generate the datasets. Values are scaled to be in $[0,\ k_A]$. We also compared to the subspace generated by the SVD_o, that is taken as the initial iterate in all methods. Nearly all methods recover U_{cA} or are very close. SVD_o gives slightly worse results for both dissimilarities, and as expected from results of Figures 3.21 and 3.22, KKT is on average quite far from U_{cA} for d_ϕ.

Case 2: $R > \frac{\text{inj}(\mathcal{M})}{2}$**.** Here we investigate a harder case. We took R uniformly in $\frac{\pi}{4}[1\ 0.8\sqrt{k_A}]$. Evolutions of the objective functions are shown in Figure 3.24. In the $d_{\sin\phi}$ case, all methods give similar results, excepted *Proj.SG*. When using d_ϕ, the best results are obtained by the *Greedy* and *Descent* approaches, followed by the *Proj.SG* and the *LogExp* and then KKT. Logically, the various methods give better results when used to optimize the dissimilarity for which they are designed. *Greedy* and *Descent* behave quite well in both cases, with *Greedy* a bit faster. An explanation is that those two methods are designed to minimize d_ϕ, which is an upper bound of $d_{\sin\phi}$. The *Proj.SG* approach is not so good in both cases and requires more iterations. This could probably be improved by chosing a better subgradient in the subdifferential set as seen in Figure 3.10.

An example of the evolution of the objective functions is shown in Figure 3.25. For both dissimilarities the *Proj.SG* appears to need more time to achieve the same decrease as the other algorithms. The large standard deviation of KKT with d_ϕ has the same explanation as in the previous case. For d_ϕ, *LogExp* often converges to the same result as *Greedy* and *Descent* but not in all cases, which explains the wider standard deviation.

In Figure 3.26 we can see the mean dissimilarities between the final solutions provided by the various algorithms and the reference subspaces U_{cA} and U_{cB}. Since we have $\sum_i d^2_{\sin\phi}(U_{cB}, W_i) < \sum_i d^2_{\sin\phi}(U_{cA}, Y_i)$, SVD_o recovers U_{cB}. The best solutions are close to U_{cA}.

Case 3: $r_1 = r_2 = 0$**.** We examine the case with $r_1 = r_2 = 0$, $R = R_{max}$ and R uniformly in $\frac{\pi}{4}[1\ 0.8\sqrt{k_A}]$, which implies that we know that the optimal

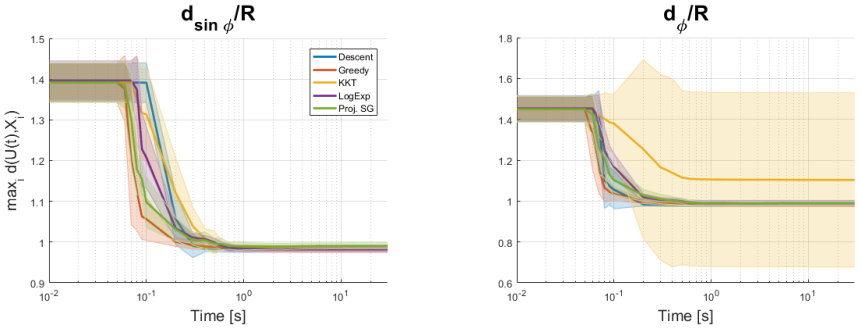


Figure 3.21: Mean on 50 tests of the evolution of the objective functions for the various methods for d_ϕ and $d_{\sin\phi}$, case 1. Shaded areas represent the standard deviations.

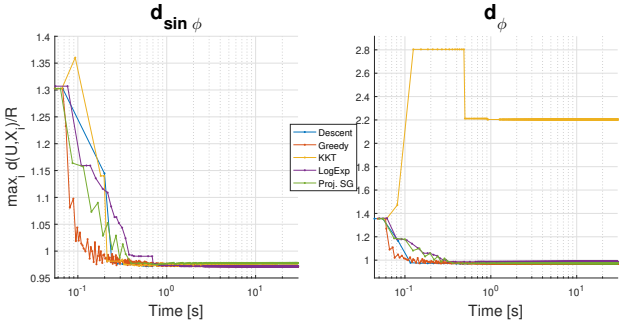


Figure 3.22: Example of evolution of the objective functions for the various methods for d_ϕ and $d_{\sin\phi}$, case 1.

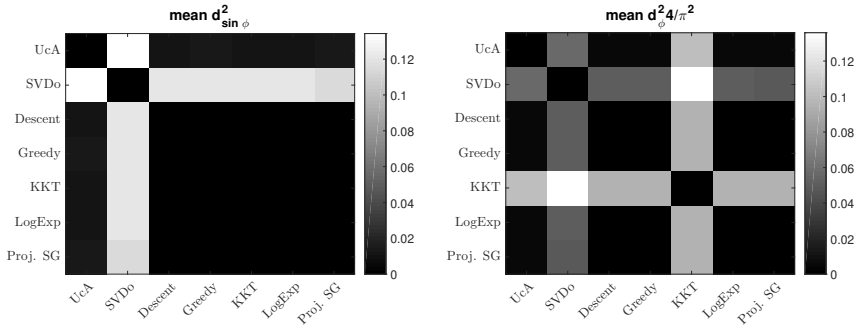


Figure 3.23: Mean of dissimilarities between the final U's for d_ϕ and $d_{\sin\phi}$, case 1.

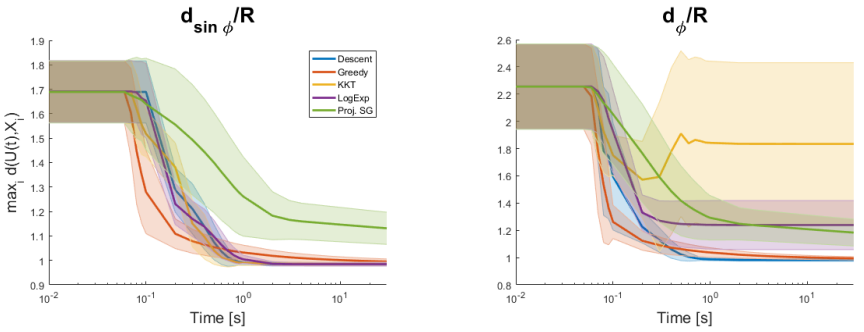


Figure 3.24: Mean on 50 tests of the evolution of the objective functions for the various methods for d_ϕ and $d_{\sin\phi}$, case 2. Shaded areas represent the standard deviations.

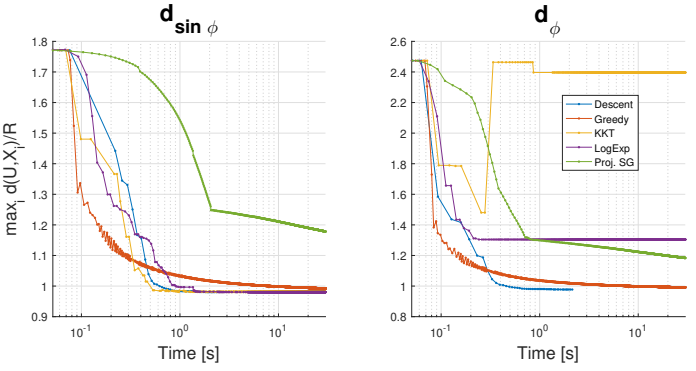


Figure 3.25: Example of evolution of the objective functions for the various methods for d_ϕ and $d_{\sin\phi}$, case 2.

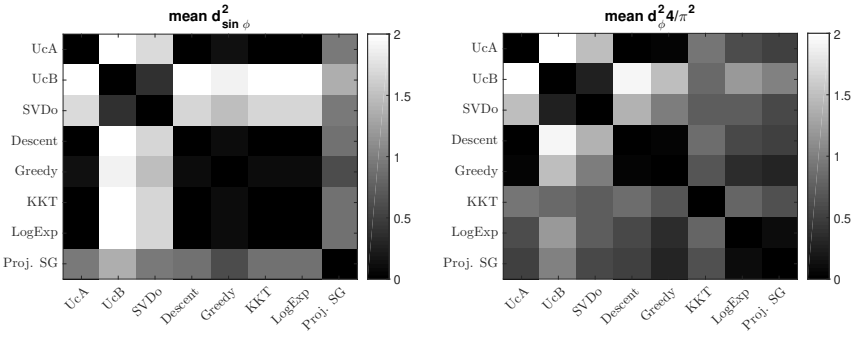


Figure 3.26: Mean of dissimilarities between the final U's for d_ϕ and $d_{\sin\phi}$, case 2.

value is R. Results in this case are quite different.

Evolutions of the objective functions are shown in Figure 3.27. We can see that for both dissimilarities, *Greedy* seems to be trapped at some stationary point. *Descent*, *LogExp* and KKT give clearly the best final solutions for $d_{\sin\phi}$; *Descent* and KKT for d_ϕ. *Greedy* still provides the fastest decrease in the beginning. An example of the evolution of the objective functions for the various approaches is shown in Figure 3.28. In Figure 3.29 we can see that the best solutions recover U_{cA} as expected, and that methods with similar objective function value recover similar subspaces.

The behavior of *Greedy* could be explained by the fact that using the SVD_o subspace as initial iterate, here $U^{(1)}$ is a linear combination of the W_i subspaces that are orthogonal to the subspaces $U^* = U_{cA}, U_1, U_2$ and U_3. By construction $U^{(1)}$ is orthogonal to all those subspaces. At each iteration, the projections of the X_i's on $\mathcal{G}(k,p)$ using Proposition 3.1.1 are the corresponding W_i's, which are by construction orthogonal to U^*. $U^{(t+1)}$ is computed as a point on the geodesic between $U^{(t)}$ and one W_i. Since any point on the Grassmannian geodesic between two points orthogonal to U^* stays orthogonal to U^*, the iterates will never escape from the orthogonal complement to U^*.

This reasoning is still valid for *Descent* since the update direction is computed as a linear combination of the W_i's, and remains orthogonal to U^*. We can speculate that the better behavior of *Descent* is probably due to the presence of noise in the computation: if a descent direction is computed even really close to orthogonality to U^*, but not exactly, then the iterate will be able to move away from this orthogonal complement of U^*. This can be seen in Figure 3.30. The first plateau of *Descent* corresponds to the plateau of *Greedy*. The second steep decrease corresponds to bigger step sizes. Escaping this orthogonal subspace can be facilitated for *Greedy* simply by initializing randomly the first iterate, or adding noise to the update direction, as shown in Figure 3.30.

We also investigated the importance of the choice of the update direction (simple as in *Greedy* or locally optimal as in *Descent*) and of the step size ($\frac{1}{iter+1}$ as in *Greedy* or optimal as in *Descent*). In Figure 3.31 all four possible combinations of update direction and step size for case 2 are compared. Using greedy direction with optimal step size does not always converge, as opposed to the three others possibilities. *Greedy* is still the fastest method.

Varying k. We also briefly investigated the influence of the dimension k chosen on the results for case 2. Results are shown in Figure 3.32 for k from 1 to 4, with an optimal k^* of 2 by construction. The final objective value is scaled in order to not depend on k. We can see that while $k \leq k^*$, the final objective value of the best methods stays flat. Once the chosen k is too high, as expected the objective value increases. Since it is not possible to find a (k^*+1)th direction close to all subspaces at once the cost function increases.

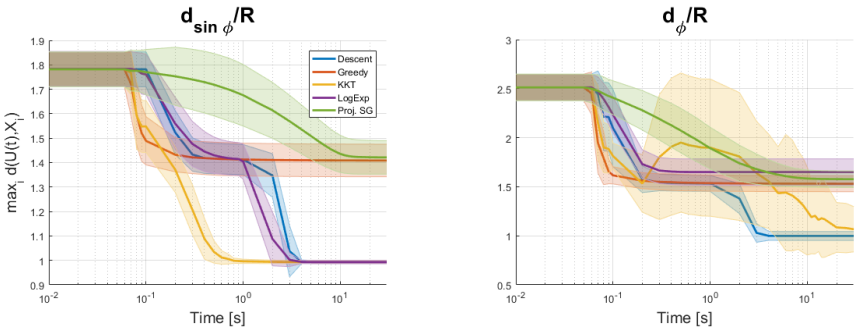


Figure 3.27: Mean on 50 tests of the evolution of the objective functions for the various methods for d_ϕ and $d_{\sin\phi}$, case 3. Shaded areas represent the standard deviations.

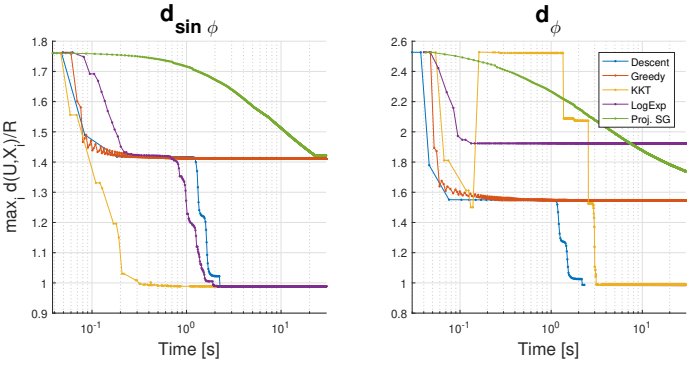


Figure 3.28: Example of evolution of the objective functions for the various methods for d_ϕ and $d_{\sin\phi}$, case 3.

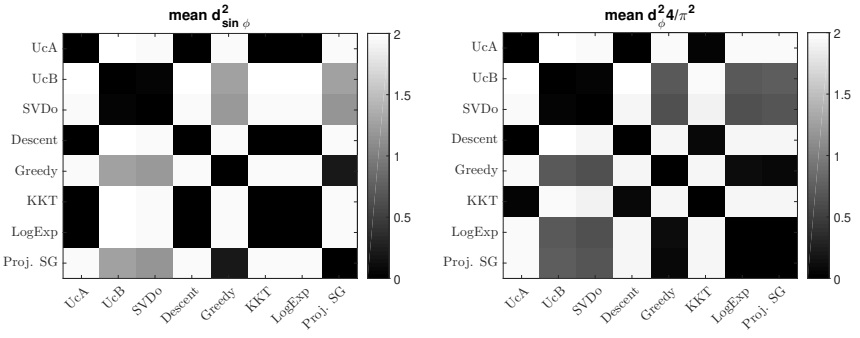


Figure 3.29: Mean of dissimilarities between the final U's for d_ϕ and $d_{\sin\phi}$, case 3.

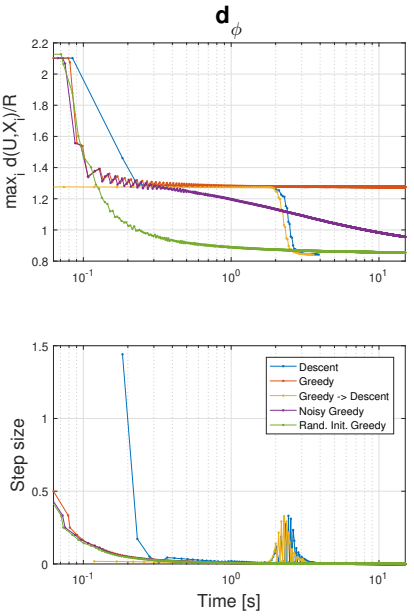


Figure 3.30: For d_ϕ, case 3, comparison of evolution of the objective functions and the associated step sizes for the methods *Greedy*, *Descent*, *Descent* initialized with the output of *Greedy* (denoted *Greedy* $\rightarrow$ *Descent*), a noisy version of *Greedy* where some noise is added to the direction (*Noisy Greedy*), and *Greedy* with a random initialization (*Rand. Init. Greedy*).

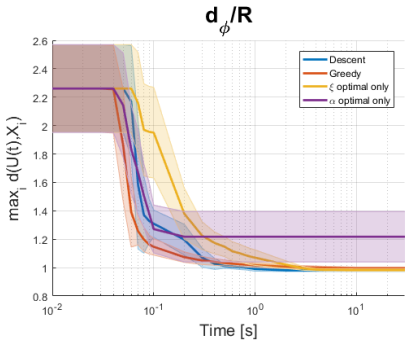


Figure 3.31: For d_ϕ, case 2, comparison of evolution of the objective functions for the methods *Greedy*, *Descent*, *Greedy* with optimal step size and *Descent* with fixed step size.

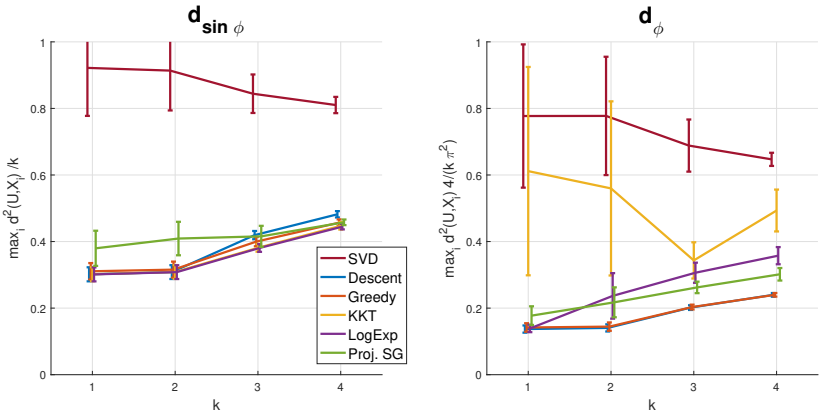


Figure 3.32: Influence of parameter k on the final objective value for the different methods. Mean and standard deviations on 50 runs.

3.4.2 Example on gene expression datasets

We applied the Grassmannian minimum enclosing ball approach to the gene expression datasets of Table 2.2. We used the same procedure as described in Section 2.3.3.1, taking the basis U of Equation (2.9) as the minimax center (computed using the descent direction approach).

We compared this approach, denoted $Minimax$ to two other methods and the case without batch effect removal method ($None$ approach described in Section 2.3.3). The first approach, denoted SVD, uses an SVD on the concatenation of all training batches to compute U:

$$U \leftarrow \text{k first left singular vectors}([X_1, \ldots, X_m])$$

where each X_i represents a batch in the training set. The second approach is the SVD_o as detailed previously in Equation (3.35). In Algorithm 4, the batch effect free training and testing sets are then computed as their projections on U using Equation (2.9). For the $Minimax$, SVD and SVD_o approaches we computed U of dimension $k = [3, 5, 10]$.

We first computed a new basis U from all the batches of Table 2.2. Associations between batches or ER and genes represented in this new basis are visible in Figure 3.33. Compared to the initial values (see plot 'None' in Figure 2.6), all methods increase the maximum R^2 value associated to the ER factor, while the association with batches decreases for some genes, but not all. Genes with higher ER association tend to be less associated to batches, and vice-versa. Taking higher values for k impacts less the R^2 values.

We plotted the first two principal components of the datasets projected on U in Figure 3.34. Those components still present a global clustering by batch,

but less clearly than in the original case (see plot 'None' in Figure 2.7). The clustering by batch becomes clearer when increasing k.

Results using the same procedure as the one described in Section 2.3.3.1 are shown in Figure 3.36 for each test dataset. Globally, datasets with only ER- samples give bad results, probably due to the majority of ER+ samples in the training sets, but slightly better than the methods tested in Section 2.3.4 (see Figure 2.11). Datasets with only ER+ samples give slightly worse results than the ones with both negative and positive ER samples. In the SVD method, results for case $k = 3$ worsen the results as the first main components are strongly linked to the batches. Increasing k improves the results of SVD, but for a same k value the performances of SVD_o and $Minimax$ are better. $Minimax$ and SVD_o give similar results: increasing k improves the performances for datasets with a majority of ER+ samples, but decreases for the ones with only ER- samples if k is too high.

More details of the effects of the various classifiers are illustrated in Figure 3.35. Each plot is representative of a the behavior of a group of datasets: GSE17705_{96} for the datasets with only ER+, GSE5327_{96} for the datasets with only ER-, and GSE3494_{96} for the others. When both classes are present, the classifiers present similar performances, with the knn a bit worse for SVD and None. When only one class is present in the test set, influence of the classifier is bigger. If we rank those classifiers based on their performances the trends flip from the ER+ only case to the ER- only one. In particular, RF appears to be more sensitive to the change of class repartition.

Results using the subsampled datasets described in Table 2.3 are shown in Figure 3.37. Using training sets with a higher proportion of ER- samples worsens the performances on ER+ only datasets but improves significantly the results for ER- only datasets. $Minimax$ and SVD_o behave similarly in that k does not influence their BCR values, which are higher than those for SVD.

Working with associated subspaces helps to find first components less linked to the batches than working with the different datasets themselves. The similarity of the performances of SVD_o and $Minimax$ could be explained by the fact that with those datasets, the l_∞- and l_2-centers of mass are close.

3.4.3 Example on parametric model order reduction

Extracting common information from merged datasets is not the only possible use of the Grassmannian minimax algorithms introduced in Section 3.3. Here we briefly present an application to model order reduction of parametric dynamical systems, where the goal is that the reduction technique performs well over a given range of parameter values.

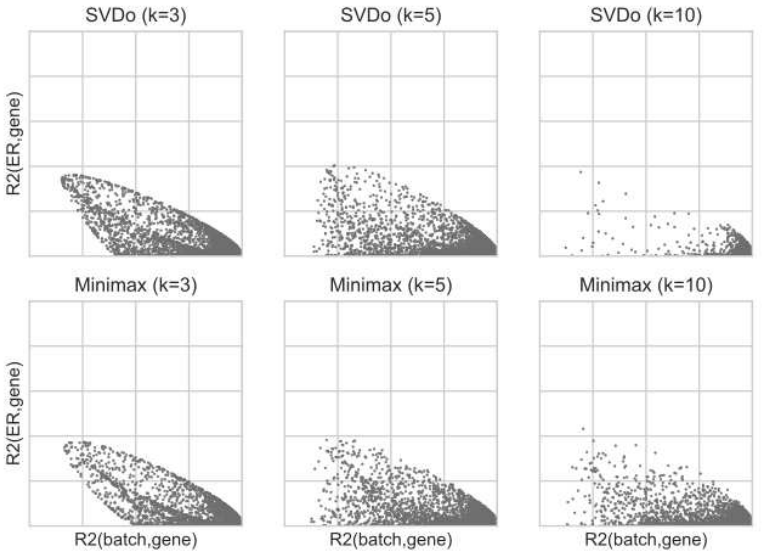


Figure 3.33: R^2 values between the gene expression values and the batches versus the ER factor, for SVD_o and $Minimax$ approaches applied on all aggregated dataset.

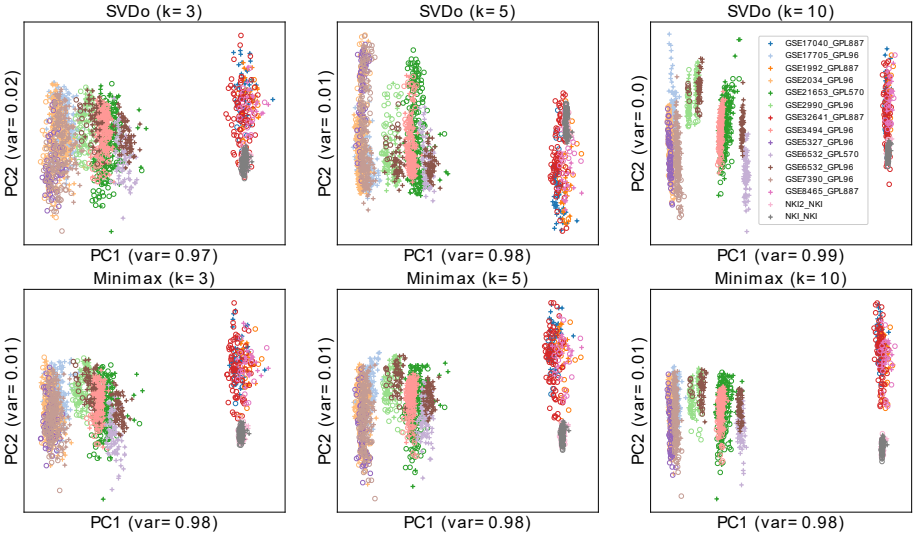


Figure 3.34: Global evaluation of batch effects: plots of the first two principal components before and after representation in the new basis U for SVD_o and $Minimax$ approaches. Colors represent the batch membership, $o/+$ corresponds to ER-/ER+ status. The percentage of variance explained by each component is given in parenthesis.

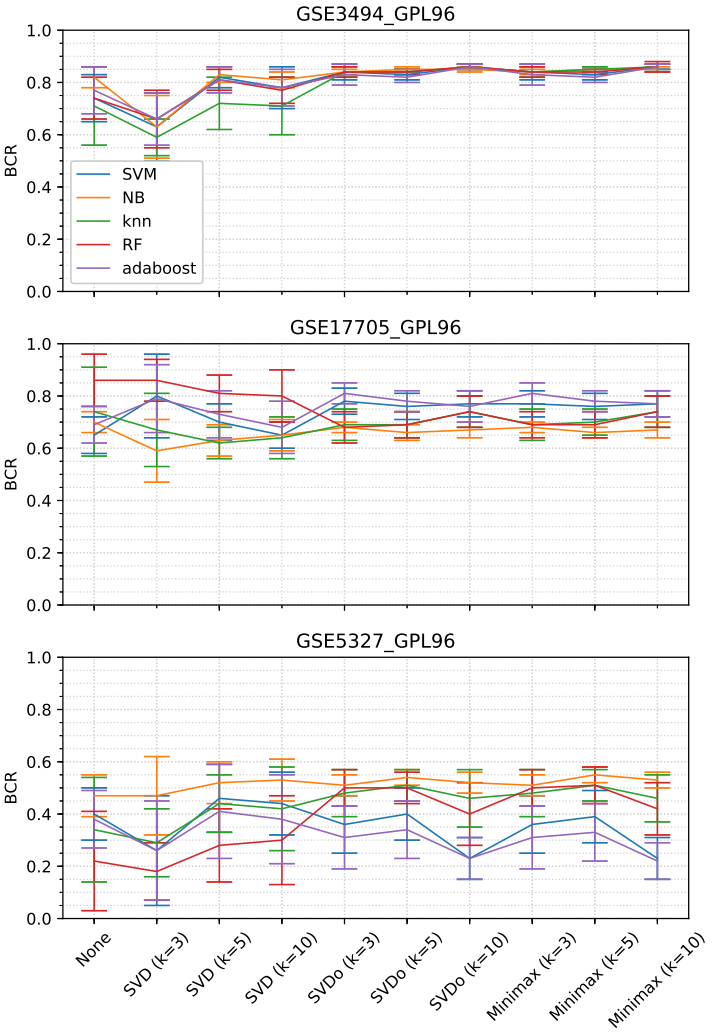


Figure 3.35: Influence of using SVD, SVD_o and $Minimax$ as batch effect removal methods on BCR for the GSE3494_{96} (ER+ and ER-), GSE17705_{96} (only ER+) and GSE5327_{96} (only ER-) test datasets. Mean on all training datasets and on all numbers of selected features.

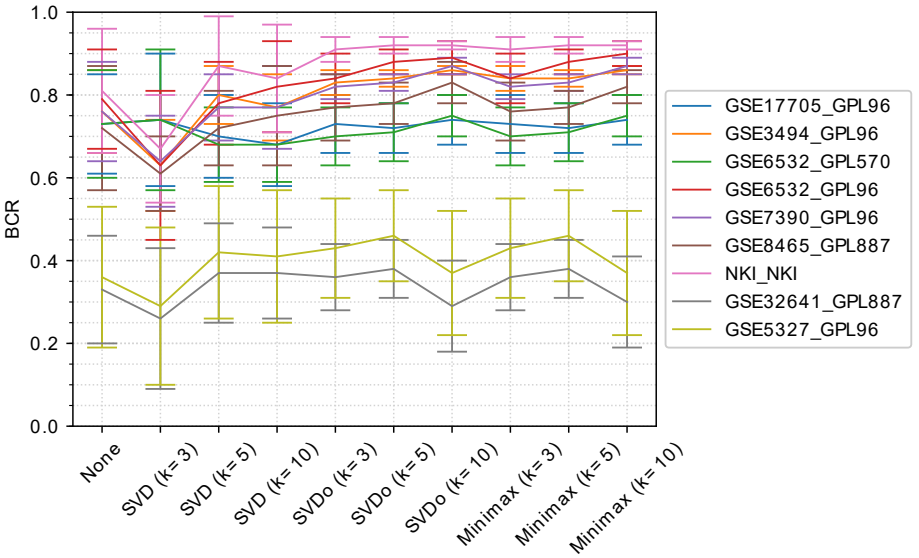


Figure 3.36: Influence of using SVD, SVD_o and $Minimax$ as batch effect removal methods on BCR for the different testing datasets. Mean on all training datasets, on all classifiers and on all numbers of selected features.

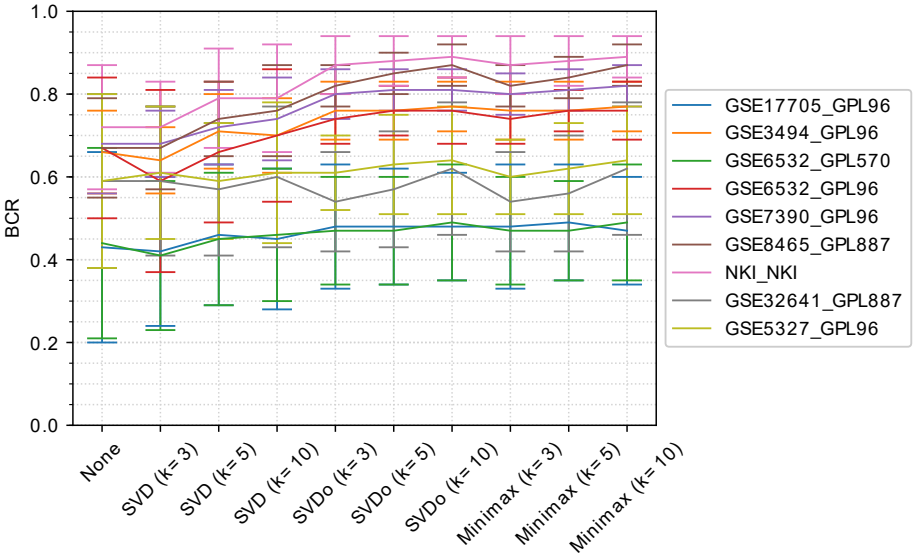


Figure 3.37: Influence of using SVD, SVD_o and $Minimax$ as batch effect removal methods on BCR for the different testing datasets when training set is modified to have a higher proportion of ER- samples. Mean on all training datasets, on all classifiers and on all numbers of selected features.

Many physical systems can be represented as a system of differential equations:

$$\begin{aligned} \dot{x} &= Ax + Bu \qquad\qquad x \in \mathbb{R}^N, N \gg 1. \\ y &= Cx + Du \end{aligned} \tag{3.36}$$

or the corresponding representation in frequency domain by its transfert function $H(s) = C(sI - A)^{-1}B + D$. This kind of system is usually studied using numerical simulations, which can require large-scale models. To decrease the associated computational burden, a reduced model can be used:

$$\begin{aligned} \dot{\hat{x}} &= \hat{A}\hat{x} + \hat{B}u \qquad\qquad \hat{x} \in \mathbb{R}^{\hat{N}}, N > \hat{N} > 1. \\ \hat{y} &= \hat{C}\hat{x} + \hat{D}u. \end{aligned}$$

Ideally, the reduced model should preserve some important system properties like stability or passivity, and the approximation error measured by $||y - \hat{y}||$ or $||H - \hat{H}||$ should be small. Such models approximate the initial high resolution simulations but are low-dimensional and so faster to solve.

One well-known method is balanced truncation that looks for matrices (V, W) such that the reduced order model

$$\begin{aligned} W^\top V\ \dot{\hat{x}} &= W^\top AV\ \hat{x} + W^\top Bu \\ \hat{y} &= CV\ \hat{x} + Du \end{aligned}$$

is a good approximation of system (3.36). Roughly, the system is first transformed to a basis where the states which are difficult to reach are simultaneously difficult to observe. This uses the reachability and the observability gramians, which are solutions to the reachability and the observability Lyapunov equations. Those states are then truncated. For more details on balanced truncation, see for example [54, 20] and references within.

Equations (3.36) can depend on a set of parameters p representing for example material properties, initial conditions or the system geometry:

$$\begin{aligned} \dot{x} &= A_p x + B_p u \\ y &= C_p x + D_p u. \end{aligned} \tag{3.37}$$

As those parameters can vary, the reduced model should approximate the original high resolution system with high fidelity over a range of parameters [20]. Here, we consider a set of parameter values $P = \{p_1, ..., p_m\}$ for which we know the associated projection subspaces V_1,...,V_m, W_1,...,$W_m \in \mathcal{G}(n, N)$. We consider the case where we do not know the exact value of p_{new}, and so want to compute unique reference values (V_{ref}, W_{ref}) that will approximate system (3.37) well enough for any p_{new} in range $[\min_i p_i, \max_i p_i]$. We compare three approaches:

– Pick one value: choose a $p = p_i \in P$ and use projections

$$V_{ref} = V_{p_i}$$
$$W_{ref} = W_{p_i},$$

– Minimax: take

$$V_{ref} = \min_{V} \max_{p_i \in P} d^2(V, V_{p_i})$$
$$W_{ref} = \min_{W} \max_{p_i \in P} d^2(W, W_{p_i}),$$

– SVD$_o$: take

$$V_{ref} = \min_{V} \sum_{p_i \in P} d^2_{\sin\phi}(V, V_{p_i})$$
$$W_{ref} = \min_{W} \sum_{p_i \in P} d^2_{\sin\phi}(W, W_{p_i}).$$

This is equivalent to selecting the first left singular vectors of the concatenation of all associated orthonormal basis [45], a method proposed in [163] and [103, Section 5.2].

We also computed the standard reduced model by taking the exact best projections $V_{ref} = V_{p_{new}}$ and $W_{ref} = W_{p_{new}}$. From the estimated V_{ref}, W_{ref}, we then compute the corresponding reduced transfer function $H_r(V_{ref}, W_{ref})$ and estimate the error by comparing it to the initial model using $||H(p_{new}) - H_r||_\infty$ with $||H(j\omega)||_\infty = \sup_\omega \sigma_1 (H(j\omega))$ and σ_1 is the maximum singular value.

We consider the system of stationary heat equation [137, 79]

$$\begin{aligned} \frac{\partial u}{\partial t} - \nabla (\sigma(\xi) \nabla u) &= f && \text{in } \Omega \times (0,T) \\ u &= 0 && \text{on } \partial\Omega \times (0,T) \end{aligned} \tag{3.38}$$

with the heat conductivity coefficient

$$\sigma(\xi) = \begin{cases} 1 + p_i & \text{for } \xi \in D_i, i = 1, ..., 4 \\ 1 & \text{for } \xi \in \Omega \backslash \left(\cup_{i=1}^4 D_i \right) \end{cases}$$

with f a source term, $\Omega = [0,4]^2$ and the D_i's are four discs of radius 0.5 and centered in $(1,1)$, $(1,3)$, $(3,1)$ and $(3,3)$ as in Figure 3.38. Parameters p_i vary in $[0.1, 10]^4$. This system is discretised for $f = 1$ using finite element method,

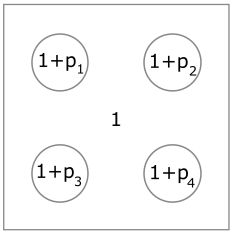


Figure 3.38: Illustration of domain of system (3.38).

leading to the system

$$E\dot{x} = A_\mu x + Bu$$
$$y = Cx.$$

with $A_\mu = A_0 + \sum_{i=1}^{4} \mu_i A_i$. The order of the reduced model, k, is fixed as 25. We took $P = \{0.1, 2, 4, 6, 8, 10\}$, $p = p_1 = p_2 = p_3 = p_4$ and tested on $p_{new} = [0.1, 0.5 : 0.5 : 10]$.

Figure 3.39 shows the evolution of the objective function for the methods tested and Figure 3.40 shows the dissimilarities between the computed representatives and the various (V_i, W_i)'s. SVD_o gives a different result due to $(V_{0.1}, W_{0.1})$ that is further away from the others (V_i, W_i)'s. *Greedy* and *Descent* behave well in all cases. We know that the *Descent* method converges to at least a stationary point so for the next experiments we use the *Descent* implementation.

Results are shown in Figure 3.41 in logarithmic and linear scales for the three considered methods and the standard approach. When picking one parameter value as reference the error logically decreases around this value. The SVD_o error is minimal around $p = 4$ while the *minimax* error is minimal around $p = 2.5$. Most choices give good approximation on a large set of parameter values, but the error generally increases for p close to the bounds of interval values, especially for p close to 0.1. The maximal errors on H for each method are shown in dotted lines. We can see that the *minimax* approach behaves quite well on the whole range of parameter values tested, only the basis corresponding to $p = 2$ gives better results. Without knowing a priori which basis to choose as a reference, the *minimax* approach gives us a good idea on which parameter value could minimize the worst case.

If we modify the training set P to p values as in Figure 3.42, we obtain Figure 3.43, where we can see that as expected the *minimax* approach is less sensitive to the set P than the SVD_o one.

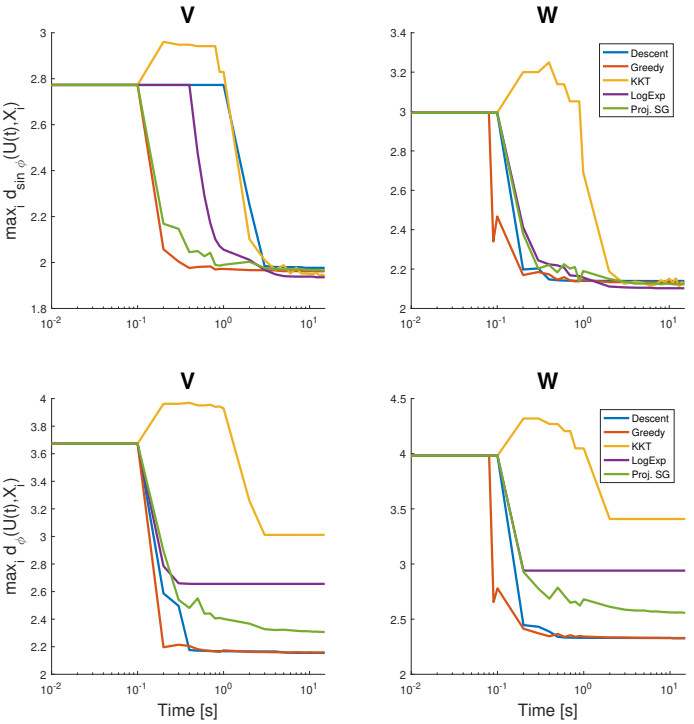


Figure 3.39: Evolution of the objective functions for the various methods for d_ϕ and $d_{\sin\phi}$.

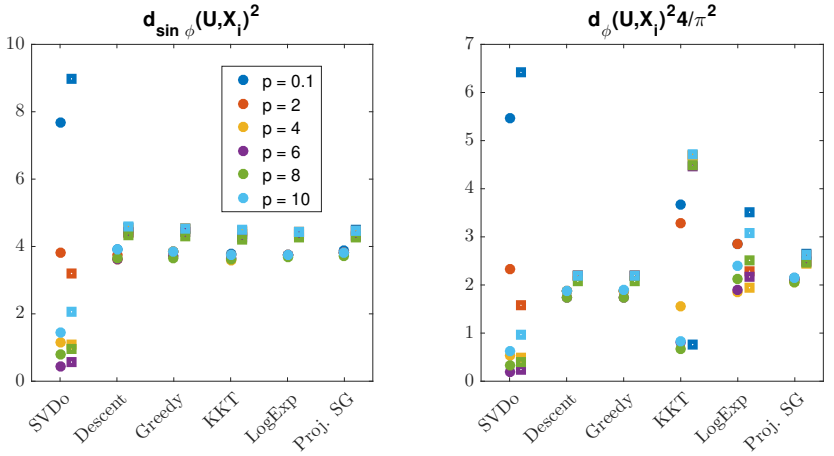


Figure 3.40: Dissimilarities between the final (V_{ref}, W_{ref})'s and the (V_i, W_i)'s for d_ϕ (squares) and $d_{\sin\phi}$ (circles).

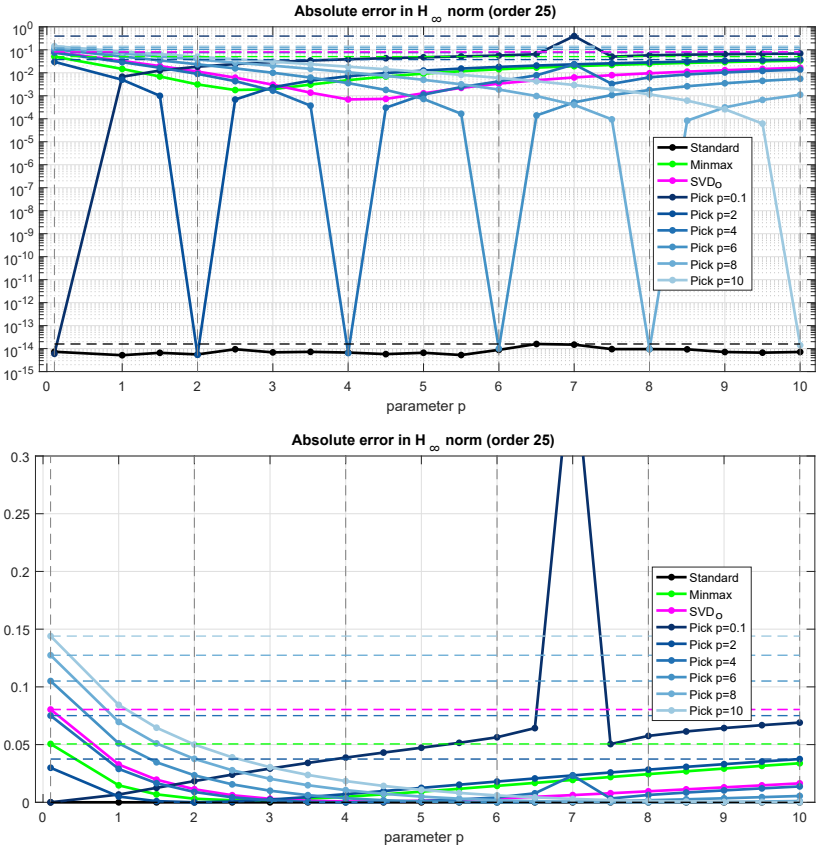


Figure 3.41: Comparison of the three methods on system (3.38), in logarithmic and linear scale.

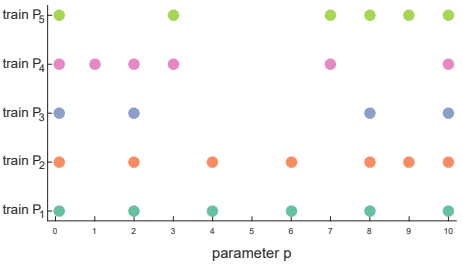


Figure 3.42: Values of parameter kept in P.

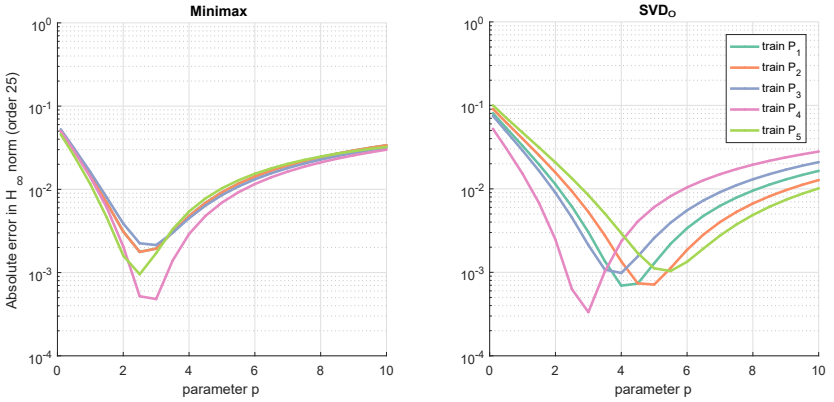


Figure 3.43: Results for *minimax* and SVD_o approaches for various sets P.

3.5 Conclusion

In this chapter, we have proposed a formulation to extract the best common subspace of a fixed dimension shared across various higher dimensional subspaces. In contrast to the usual CCA-like methods, we minimize the maximal dissimilarity and not the sum of the squared dissimilarities; we are then computing a center. In contrast to the usual minimum enclosing ball problem, we consider subspaces and not vectors in $\mathbb{R}^p$, which allows us also to work with a lower dimensional central subspace representing subspaces of various dimensions. However, this problem is nonsmooth due to the minimax of the objective function. Moreover, the use of subspaces implies that the problem is constrained and nonconvex. To solve this nonsmooth nonconvex constrained problem, we proposed five approaches based on two types of dissimilarities. We have shown on synthetic data that all of the methods recover the expected central subspace, except the projected subgradient (whose implementation could probably still be tuned). We tested the minimax approach in the batch effect removal procedure used in the previous chapter, which showed that for a fixed dimension k, working with the associated subspaces instead of the datasets themselves improves classification performances. We then tested the minimax formulation on a parametric model order reduction and showed that our formulation can be used to choose a unique representative minimizing the H_∞ norm on a set of parameter values.

Comparison of the proposed methods. The first approach, *Greedy*, can be seen as a greedy approach that at each iteration moves the current iterate in the direction of the furthest point. Despite its simplicity, this approach works quite well for both dissimilarities. It is the fastest on all of our tests, and recovers the best solution in terms of objective function. The only exception is in the special case without noise (Case 3 in Section 3.4.1) where this algorithm appears to get stuck at a stationary point.

The second method proposed, *KKT*, is based on the interpretation of the first order optimality conditions using dissimilarity $d_{\sin\phi}$ and can be seen as a dual method. Compared to the previous greedy method, its main advantage is to provide bounds on the objective function using the duality gap. While the objective function evolution may show intervals where the value increases, the method always recovers solutions that are among the best. Its behavior is, however, more erratic with occasionally very large increases in the value of the objective function if we consider the dissimilarity d_ϕ.

The third proposed algorithm, *Proj.SG*, is a projected subgradient based on the fact that while the objective function is not differentiable everywhere, the nondifferentiable points are rare and at such points a subgradient can easily be computed if we consider dissimilarity $d_{\sin\phi}$. This method gives the worst overall performance in our tests, however there are opportunities for significant improvements in the choice of the specific subgradient and in the step size choice.

The fourth proposed method, *Descent*, minimizes at each iteration an upper bound of the objective function (using dissimilarity d_ϕ). This allows choosing the locally best update direction and step size. Computing such best choices is costly, however less iterations are needed to attain a similar objective function value than the previous methods. This descent method is then a bit slower than some other ones, but it recovers the best solution. An interesting feature of this method is that we provide a convergence proof which guarantees that the method stops close to a stationary point.

The last approach investigated, *LogExp*, optimizes a smoothed version of the objective function for $d_{\sin\phi}$ using a trust region method in Manopt. The level of accuracy in the approximation is decreased at each step to improve the results. This method is among the best ones while not the fastest. Results are less good when using d_ϕ.

From our tests, we conclude that the best option in terms of final results and speed for both dissimilarities is the greedy approach. The only drawback is in the third case studied, without noise. However, in practice with real datasets such a case is unlikely to happen, and the results could be used as an initial point for the descent method and/or the KKT-based algorithms to evaluate if we are close to a stationary point or not. While a bit slower, the *Descent* method is also a good choice that provides some guarantees on convergence.

Other possible approaches. We have investigated five approaches, those are not the only possible of course. Other approaches to solve nonconvex non-differentiable constrained problem exist. For example, another manifold optimization toolbox is ROPTLIB [66], a C++ library for Riemannian optimization which also provides a Matlab-interface. Compared to Manopt, ROPTLIB has the advantage to contain some implementations compatible with nonsmooth cost function: a Riemannian BFGS, a Riemannian subgradient [62] and a Riemannian gradient sampling [28, 63].

BFGS methods are quasi-Newton methods using the update formula

$$x^{(t+1)} = x^{(t)} + \alpha^{(t)} p^{(t)}$$

with $p^{(t)} = -H^{(t)} \nabla f(x^{(t)})$, and $H^{(t)}$ an approximation of the inverse of the Hessian. A usual requirement is that the second-order approximation $\tilde{f}$ of f in $x^{(t)}$ has its gradient matching the gradient of f in $x^{(t)}$ and $x^{(t-1)}$. If the BFGS method does not encounter any nonsmooth point, it often converges to the optimum. However such methods may fail with nonsmooth functions without ad hoc adaptations like in the line search step [173].

The Riemannian subgradient implementation generalizes the Wolfe conditions (see Section 3.2) to nonsmooth functions on manifold.

Gradient Sampling (GS) is a descent method adapted to nonsmooth problem. As explained in [28], the concept is quite simple, and is based on the fact that nonsmooth functions encountered in practice are usually differentiable almost everywhere. Assuming that the function is locally Lipschitz continuous

ensures that. So the function is differentiable at any randomly chosen point x with probability one. The function is often not differentiable at local minima, however it is generally possible to generate a sequence of iterates that will converge to a stationary point. The difficulty with nonsmooth function is that close to stationary points, the step size to ensure a decrease of the objective function value can be really small, and lead to convergence to nonstationary points. To avoid that, the GS method computes a descent direction at a point $x^{(t)}$ based on its gradient $\nabla f(x^{(t)})$ and gradients of randomly chosen points $y_1^{(t)}$, ... , $y_k^{(t)}$ close to $x^{(t)}$. A line search is then used to find a step size.

Another quite simple approach could be to use successive linear approximations of the problem using Taylor approximations of Problem (3.8) using $d_{\sin\phi}$. At each iteration, the next iterate is computed as the minimum of the linear approximation around the current iterate. As the orthonormality constraint can be relaxed as a convex set, the problem is then an SDP convex optimization problem that can be solved using for example the CVX toolbox in Matlab. This is however heavy to solve due to number of constraints.

Still another possibility would be to use deflation: a first central subspace of dimension one is computed, removed from each subspace, and the next best central subspace is then computed. Such a deflation approach is used in CCA-like methods, however as here we are minimizing a maximum there is no guarantee that the best central subspace of dimension k is included in the best central subspace of dimension $k+1$.

Chapter 4

Conclusion and perspectives

The capability to extract useful information from large datasets (in the context of merging various datasets, but also in a larger context) is becoming a key asset in bioinformatics and other fields. A global view of the entire process is essential for an in-depth understanding of the final results and so to be able to interpret and use it.

The first question to address to extract information from data is "What are we looking for?". This is an important step of the process as this will lead to the precise formulation of the objective function. It is however not always easy to answer to such a question as sometimes we are not sure of what exactly is needed. It is important to keep in mind the final use of the result: who will use it, in which context, what properties should have an ideal solution, what do we know of the data and what does it imply. However, the major objective of extracting information is to better understand those data. The ideal is then to obtain a model which is simple and interpretable, based on realistic hypotheses. Having a good insight of the data is then necessary if we want to use more complex models. If the model is dependent on very specific assumptions that are not met in practice, even the best algorithm will not give useful results. Without any certainty, the best choice is to use the most simple model possible: general hypotheses that cover a large enough range of possible data, but also without too many parameters to tune.

Once the problem is well-defined, the next question is how to solve it. This usually comes to the choice of the algorithm to solve the associated optimization problem. Many different tools can exist to solve the same problem, however they often are related to each other and should lead to similar solutions. Ideally, the choice of the specific algorithm should not influence (significantly) the final result. Depending if we want to favour simplicity of the implementation, speed, or convergence guarantees, the best algorithm can change. A faster method can

necessitate a parameter tuning step that may be less accurate if the dataset changes while an algorithm with convergence guarantees may be slower. To start an analysis and without a priori preference, going for the simplest one is generally a good choice.

When applying the methods on data, an often crucial step is parameter tuning. An ideal method will not have too many parameters to tune, and results would not depend heavily on their values. Fortunately, an approximate solution is often enough in practice, as sometimes it is not even possible to define exactly what is a "best" solution.

To validate the approach, a good idea is to first apply it on data generated artificially where all influencing parameters are controlled and the optimal solution known. As the background truth is not always verifiable in practice, being sure that the method works when the hypotheses of the model are met is a good preliminary check. When available, it is good practice to compare the results to references. New methods should have performances comparable to already existing ones on a same dataset, that ideally is well-studied and whose background truth is (at least partially) known. An important feature to check, but not always easy to obtain, is robustness and stability of the solution. Various algorithms with sligthly different parameter choices and/or hypothesis giving similar results is a positive sign. However, the best test is to check consistency of the results when applied on various datasets. But probably the best way to assess the validity of the results is to be able to understand what they represent, to interpret them. For example, if the data represent some physical system, can we interpret physically the obtained components?

Finally, all such nice models are generally pointless if they are not usable for somebody that did not conceive it. It is then important to provide an interpretable model, and to be able to explain concisely the main ideas and assumptions behind the method. Furnishing an implementation easy to use, with some pre-selected parameter values if necessary, will also facilitate an external use.

In this thesis, we investigated the problem of extracting common information from multiple datasets using linear approaches. We have approached the problem from two different angles, which led to two different formulations of the problem. In the first chapter we merge all datasets by removing their differences while in the second one we directly extract their common part.

In Chapter 2, we investigated the first approach. To capture the differences in behavior of the datasets, we proposed to use low rank matrix factorization. Such methods have the advantages to be simple in the sense that the underlying model is linear, and that the lower rank reduces the dimensionality of the data. The linearity also facilitates the interpretation of the components obtained as they are linear combinations of the original data. Building on existing methods, we have proposed a spatiotemporal Independent Component Analysis algorithm that has the advantage to treat on an equal footing both spatial and temporal options. We then validated the method on gene expres-

sion datasets and we showed that the computed components recover biological information present in pathways and experimental conditions, giving better results than standard reference algorithms in some cases. Finally, we used this approach in a large comparison of various batch effect removal methods in the context of a classification task. We analysed the effects of those methods when varying the classifiers on a large cohort of gene expression studies, with a focus on location-scale versus matrix factorization methods. While less used than the location-scale methods, matrix factorization methods proved to be more robust to changes in class frequencies between training and testing sets, and if the batch information is incomplete. An interesting solution could then be to combine both approaches, as in the FA method.

In Chapter 3, we proposed methods to extract a common representative of multiple datasets. This problem can be formulated in different ways and we choose to look for a central subspace that minimizes the maximal dissimilarity to all datasets. This minimax approach facilitates avoiding recovering a subspace representing very well all datasets except a few ones. Working with subspaces has the advantage that we do not have to take into consideration the specific representation of the central representative. However it makes the problem constrained and nonconvex. We proposed five methods, that vary from simple and fast to slower but with guaranteed convergence and robustness. Tested first on synthetic data, all methods (except the projected subgradient, whose implementation can probably still be improved) recover the solution. In the context of batch effect removal from gene expression datasets, the Grassmannian minimax approach gave results similar to the Grassmannian l_2 center of mass. For a fixed dimension k, working with associated subspaces gives better results than if considering the datasets themselves. We also showed that when tested on a parametric model order reduction problem, the minimax approach can help to choose a unique representative minimizing the H_∞ norm of the reduced system on a range of parameter values.

Both approaches are complementary: the first one (that removes the differences) is based on weaker assumptions and the risk is to not clean the datasets sufficiently, while the second one (that keeps the common part) may keep no or useless component(s) if the assumption of the presence of a common component is not met. The first approach is better for a more general analysis without a specific question in mind, while the second one is more oriented to a specific goal by construction. In both cases, it is a good idea to examine the resulting datasets before using them in further analysis. For the first approach, a plot of the first principal components can help to quickly check if most batch effects were removed. For the second approach, the dissimilarity values $d(U, X_i)$ can help to check if the common component U is indeed present in all datasets.

Perspectives

Of course, the proposed implementations of the algorithms, especially in Chapter 3, could be improved for a faster convergence, the projected subgradient in particular. As mentioned in Section 3.5, other algorithmic approaches could also be tested to solve the minimax problem.

An important question that was not investigated in this thesis is the choice of k, the dimension of the lower rank matrix factorization in Chapter 2 and of the common subspace in Chapter 3. Many heuristics exist to choose k, but its optimal value is often difficult to fix without a priori information or clear trends in the data. A simple approach could be deflation, however in cases like minimax nothing guarantees that the optimal subspace of dimension $k-1$ is included in the k-dimensional one. In the minimax case, an ideal k would be big enough to represent as much as possible the data, but not too big in order to avoid adding dimensions that are not present in all datasets. To fairly compare the solutions for different values of k, the objective function should be scaled by k as its maximal value is bounded by k or $k\frac{\pi}{2}$ depending on the dissimilarity used. To encourage the selection of a higher dimension (and so represent a larger part of the data), we could also incorporate a term evaluating how much of the data is left out in this representative, using a criterion like the one proposed in [97]:

$$\min_k \left(\max_i \frac{d^2(U_k^*, X_i)}{k} - \max_i d^2(U_k^\perp, X_i) \right) \tag{4.1}$$

with U_k^* the solution of Problem (3.8) and $U_k^\perp$ denoting the $p-k$ orthogonal complement to U_k^*. We could imagine computing a solution for a fixed k and use formula (4.1) to evaluate if k is good enough. If not, we could use the current U_k^* to initialize the next run of the algorithm to hopefully speed up its convergence.

As shown in the tests of Section 3.4, the proposed algorithms generally appear to converge to the solution. However, we showed that at least in a specific configuration some methods can fail. A more in-depth analysis of the consequences of the nonconvexity of Problem (3.8) is also of interest.

As stated before, the final use is probably the most important point to validate a method. Testing the various algorithms on more real datasets, possibly in other contexts, would be interesting.

More generally speaking, we believe that with the increasing amount of data accessible and the increasing interest of the public and industry for its analysis and all the benefits (economical or human) it can bring, how to extract useful information from multiple datasets is an important question that is, and will be, more and more studied. In bioinformatics and genomics especially, the number of features and variety of data available call for machine learning techniques to handle at the very least the first exploratory steps.

In essence, bioinformatics data often require cautious handling. While the techniques are constantly improving, such data are often noisy due to the data collection process. Having access to the background truth, if there is a well-defined one, can also be difficult. Such data are also sensitive as they often come from real persons, possibly with serious diseases. Bad diagnosis or prediction in this context could lead to much more significant consequences than a bad targeting for a personalized advertising for example. This is why questions (among many others) of interpretability and reproducibility in machine learning and data mining is of particular concern in bioinformatics.

Reproducibility is crucial to be able to guarantee efficiency of possible future treatments derived from the analysis, and can be interpreted at different levels: reproducibility of the experimental data (same protocol on the same patients gives similar measures), of the data analysis (same code implementation on same data gives same results), of the results (same conclusions for same analysis on similar data). In a context similar to this thesis, reproducibility of the data analysis can be ensured by making the code available. Reproducibility of the results is, we hope, encouraged by the use of multiple datasets in the analysis. The increasing number of datasets publicly available on repositories like the Gene Expression Omnibus and the easy access to softwares like Bioconductor facilitate the reproducibility of analysis. However it is often not easy to understand the various data formats and their implications or to choose among the large number of existing methods to handle them, especially for nonspecialists. Even with access to the specific data and method implementation used in an analysis, it can be difficult to reproduce the same results as a slight change in some parameter value or the software version can influence results. We are then convinced that data format standardization and sharing of data and code should still be improved and encouraged.

In this thesis, interpretability is promoted by the use of linear approaches that associate weights to each feature or sample, that can be interpreted as the importance of the corresponding feature or sample. We believe however that the use of up-to-date techniques like deep learning is unavoidable, but maybe more as a first step to provide research directions: with patient lives involved, results should be interpreted and checked by human specialists. This is why, in a society where hyper-specialization in a field is encouraged, it is important for data scientists working with bioinformatics data to understand at least basically the biology behind the data, and collaborate with genomics specialists.

Bibliography

[1] Thomas Abeel et al. "Robust biomarker identification for cancer diagnosis with ensemble feature selection methods". In: *Bioinformatics* 26.3 (2010), pp. 392–398.

[2] P.-A. Absil and Kyle Gallivan. *Note on the convex hull of the Stiefel manifold*. Tech. rep. 2011.

[3] P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[4] P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. "Riemannian geometry of Grassmann manifolds with a view on algorithmic computation". In: *Acta Applicandae Mathematicae* 80.2 (2004), pp. 199–220.

[5] Bijan Afsari. "Riemannian L_p center of mass: existence, uniqueness, and convexity". In: *Proceedings of the American Mathematical Society* 139.2 (2011), pp. 655–673.

[6] Bijan Afsari, Roberto Tron, and René Vidal. "On the convergence of gradient descent for finding the Riemannian center of mass". In: *SIAM Journal on Control and Optimization* 51.3 (2013), pp. 2230–2260.

[7] Noga Alon et al. "Testing of clustering". In: *SIAM Journal on Discrete Mathematics* 16.3 (2003), pp. 393–417.

[8] Orly Alter, Patrick O. Brown, and David Botstein. "Singular Value Decomposition for Genome-Wide Expression Data Processing and Modeling". English. In: *Proceedings of the National Academy of Sciences of the United States of America* 97.18 (2000), pp. 10101–10106.

[9] Orly Alter, Patrick O Brown, and David Botstein. "Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms". In: *Proceedings of the National Academy of Sciences* 100.6 (2003), pp. 3351–3356.

[10] Jesus Angulo. "Structure tensor image filtering using Riemannian L_1 and L_∞ center-of-mass". In: *Image Analysis & Stereology* 33.2 (2014), pp. 95–105.

[11] Marc Arnaudon and Frank Nielsen. "On approximating the Riemannian 1-center". In: *Computational Geometry* 46.1 (2013), pp. 93–104.

[12] Miroslav Bačák et al. “A Second Order Nonsmooth Variational Model for Restoring Manifold-valued Images”. In: *SIAM Journal on Computing* 38.1 (2016), pp. 567–597.

[13] Mihai Badoiu and Kenneth L. Clarkson. “Smaller core-sets for balls”. In: *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2003, pp. 801–802.

[14] Mihai Bādoiu, Sariel Har-Peled, and Piotr Indyk. “Approximate clustering via core-sets”. In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM. 2002, pp. 250–257.

[15] L. G. Bazhenov. “Convergence conditions of a minimization method of almost-differentiable functions”. In: *Cybernetics and Systems Analysis* 8.4 (1972), pp. 607–609.

[16] Amir Beck. *First-order methods in optimization*. Vol. 25. SIAM, 2017.

[17] Christopher G. Bell et al. “Genome-wide DNA methylation analysis for diabetic nephropathy in type 1 diabetes mellitus”. In: *BMC medical genomics* 3.1 (2010), p. 33.

[18] Asa Ben-Hur et al. “Support vector clustering”. In: *Journal of machine learning research* 2.Dec (2001), pp. 125–137.

[19] Monica Benito et al. “Adjustment of systematic microarray data biases”. In: *Bioinformatics* 20.1 (2004), pp. 105–114.

[20] Peter Benner, Serkan Gugercin, and Karen Willcox. “A survey of projection-based model reduction methods for parametric dynamical systems”. In: *SIAM review* 57.4 (2015), pp. 483–531.

[21] Dennis S. Bernstein. *Matrix mathematics: Theory, facts, and formulas with application to linear systems theory*. Vol. 41. Princeton university press Princeton, 2005.

[22] Dimitri P Bertsekas. “Nonlinear Programming. Athena Scientific Belmont”. In: *Massachusets, USA* (1999).

[23] Dimitri P. Bertsekas. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

[24] Christopher M. Bishop. *Pattern recognition and machine learning*. Ed. by Springer. 2006.

[25] Nicolas Boumal et al. “Manopt, a Matlab Toolbox for Optimization on Manifolds”. In: *Journal of Machine Learning Research* 15 (2014), pp. 1455–1459.

[26] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[27] Yaroslav Bulatov et al. “Hand recognition using geometric classifiers”. In: *Biometric Authentication*. Springer, 2004, pp. 753–759.

[28] James V. Burke et al. “Gradient Sampling Methods for Nonsmooth Optimization”. In: *arXiv preprint arXiv:1804.11003* (2018).

[29] Jean-François Cardoso. “High-order contrasts for independent component analysis”. In: *Neural Comput.* 11.1 (Jan. 1999), pp. 157–192.

[30] Jean-François Cardoso and Antoine Souloumiac. “Blind Beamforming for Non Gaussian Signals”. In: *IEE Proceedings-F* 140 (1993), pp. 362–370.

[31] Marta Cavaleiro and Farid Alizadeh. “A faster dual algorithm for the Euclidean minimum covering ball problem”. In: *Annals of Operations Research* (2017), pp. 1–13.

[32] Jair Cervantes et al. “Support vector machine classification for large data sets via minimum enclosing ball clustering”. In: *Neurocomputing* 71.4-6 (2008), pp. 611–619.

[33] Prabhakar Chalise and Brooke L. Fridley. “Integrative clustering of multi-level ‘omic data based on non-negative matrix factorization algorithm”. In: *PloS one* 12.5 (2017), e0176278.

[34] Chao Chen et al. “Removing batch effects in analysis of expression microarray data: an evaluation of six batch adjustment methods”. In: *PloS one* 6.2 (2011), e17238.

[35] D. Chessel and M. Hanafi. “Analyses de la co-inertie de K nuages de points”. In: *Revue de statistique appliquée* 44.2 (1996), pp. 35–60.

[36] George Chrystal. “On the problem to construct the minimum circle enclosing n given points in the plane”. In: *Proceedings of the Edinburgh Mathematical Society* 3 (1885), pp. 30–33.

[37] Yongjun Chu and David R Corey. “RNA sequencing: platform selection, experimental design, and data interpretation”. In: *Nucleic acid therapeutics* 22.4 (2012), pp. 271–274.

[38] Gary A Churchill. “Fundamentals of experimental design for cDNA microarrays”. In: *Nature genetics* 32.4s (2002), p. 490.

[39] Pierre Comon and Christian Jutten. *Handbook of Blind Source Separation: Independent Component Analysis and Applications*. Independent Component Analysis and Applications Series. Elsevier Science, 2010.

[40] Philip L. De Jager et al. “Alzheimer’s disease: early alterations in brain DNA methylation at ANK1, BIN1, RHBDF2 and other loci”. In: *Nature neuroscience* 17.9 (2014), p. 1156.

[41] P.M. Dearing and Christiane R. Zeck. “A dual algorithm for the minimum covering ball problem in Rn”. In: *Operations Research Letters* 37.3 (2009), pp. 171–175.

[42] Christine Desmedt et al. “Strong time dependence of the 76-gene prognostic signature for node-negative breast cancer patients in the TRANSBIG multicenter independent validation series”. In: *Clinical cancer research* 13.11 (2007), pp. 3207–3214.

[43] Gunther Dirr, Uwe Helmke, and Christian Lageman. “Nonsmooth Riemannian optimization with applications to sphere packing and grasping”. In: *Lagrangian and Hamiltonian Methods for Nonlinear Control 2006*. Springer, 2007, pp. 29–45.

[44] Ashley S. Doane et al. “An estrogen receptor-negative breast cancer subset characterized by a hormonally regulated transcriptional program and response to androgen”. In: *Oncogene* 25.28 (2006), pp. 3994–4008.

[45] Bruce Draper et al. “A flag representation for finite collections of subspaces of mixed dimensions”. In: *Linear Algebra and its Applications* 451 (2014), pp. 15–32.

[46] Alan Edelman, Tomás A. Arias, and Steven T. Smith. “The geometry of algorithms with orthogonality constraints”. In: *SIAM journal on Matrix Analysis and Applications* 20.2 (1998), pp. 303–353.

[47] D. Jack Elzinga and Donald W. Hearn. “The minimum covering sphere problem”. In: *Management science* 19.1 (1972), pp. 96–104.

[48] Manel Esteller. “Cancer epigenomics: DNA methylomes and histone-modification maps”. In: *Nature reviews genetics* 8.4 (2007), p. 286.

[49] Kaspar Fischer, Bernd Gärtner, and Martin Kutz. “Fast smallest-enclosing-ball computation in high dimensions”. In: *European Symposium on Algorithms*. Springer. 2003, pp. 630–641.

[50] Kyle A. Gallivan et al. “Efficient algorithms for inferences on grassmann manifolds”. In: *IEEE Workshop on Statistical Signal Processing*. IEEE. 2003, pp. 315–318.

[51] Bernhard C. Geiger and Gernot Kubin. “Signal enhancement as minimization of relevant information loss”. In: *SCC 2013; 9th International ITG Conference on Systems, Communication and Coding*. VDE. 2013, pp. 1–6.

[52] Deena M.A. Gendoo et al. *genefu: Computation of Gene Expression-Based Signatures in Breast Cancer*. R package version 2.16.0. 2019.

[53] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Vol. 3. JHU Press, 1996.

[54] Serkan Gugercin and Athanasios C. Antoulas. “A survey of model reduction by balanced truncation and some new results”. In: *International Journal of Control* 77.8 (2004), pp. 748–766.

[55] Mohamed Hanafi, Achim Kohler, and El-Mostafa Qannari. "Connections between multiple co-inertia analysis and consensus principal component analysis". In: *Chemometrics and intelligent laboratory systems* 106.1 (2011), pp. 37–40.

[56] Sariel Har-Peled and Kasturi Varadarajan. "Projective clustering in high dimensions using core-sets". In: *Proceedings of the eighteenth annual symposium on Computational geometry*. ACM. 2002, pp. 312–318.

[57] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Corrected. Springer, Aug. 2003.

[58] Donald W. Hearn and James Vijay. "Efficient algorithms for the (weighted) minimum circle problem". In: *Operations Research* 30.4 (1982), pp. 777–795.

[59] Gen Hori et al. "Blind gene classification based on ICA of microarray data". In: *3rd International Conference on Independent Component Analysis and Signal Separation. In Proc ICA*. Vol. 3. 2001, pp. 332–336.

[60] Roman Hornung, Anne-Laure Boulesteix, and David Causeur. "Combining location-and-scale batch effect adjustment with data cleaning by latent factor adjustment". In: *BMC bioinformatics* 17.1 (2016), p. 27.

[61] Roman Hornung et al. "Improving cross-study prediction through addon batch effect adjustment or addon normalization". In: *Bioinformatics* 33.3 (2016), pp. 397–404.

[62] Seyedehsomayeh Hosseini, Wen Huang, and Rohollah Yousefpour. "Line search algorithms for locally Lipschitz functions on Riemannian manifolds". In: *SIAM Journal on Optimization* 28.1 (2018), pp. 596–619.

[63] Seyedehsomayeh Hosseini and André Uschmajew. "A Riemannian gradient sampling algorithm for nonsmooth optimization on manifolds". In: *SIAM Journal on Optimization* 27.1 (2017), pp. 173–189.

[64] Harold Hotelling. "Relations Between Two Sets of Variates". In: *Biometrika* 28.3/4 (1936), pp. 321–377.

[65] Wen Huang, Kyle A. Gallivan, and P.-A. Absil. "A Broyden class of quasi-Newton methods for Riemannian optimization". In: *SIAM Journal on Optimization* 25.3 (2015), pp. 1660–1685.

[66] Wen Huang et al. *ROPTLIB: an object-oriented C++ library for optimization on Riemannian manifolds*. Tech. rep. FSU16-14.v2. Florida State University, 2016.

[67] Philip M. Hubbard. "Approximating polyhedra with spheres for time-critical collision detection". In: *ACM Transactions on Graphics (TOG)* 15.3 (1996), pp. 179–210.

[68] Philip Martyn Hubbard. “Collision detection for interactive graphics applications”. In: *IEEE Transactions on Visualization and Computer Graphics* 1.3 (1995), pp. 218–230.

[69] Lawrence Hunter. “Molecular biology for computer scientists”. In: *Artificial intelligence and molecular biology* (1993), pp. 1–46.

[70] Aapo Hyvärinen and Erkki Oja. “Independent component analysis: algorithms and applications”. In: *Neural networks* 13.4-5 (2000), pp. 411–430.

[71] Mariya Ishteva, P.-A. Absil, and Paul Van Dooren. “Jacobi algorithm for the best low multilinear rank approximation of symmetric tensors”. In: *SIAM Journal on Matrix Analysis and Applications* 34.2 (2013), pp. 651–672.

[72] Jin-ichi Itoh and Minoru Tanaka. “The dimension of a cut locus on a smooth Riemannian manifold”. In: *Tohoku Mathematical Journal, Second Series* 50.4 (1998), pp. 571–575.

[73] Søren Kruse Jacobsen. “An algorithm for the minimax Weber problem”. In: *European Journal of Operational Research* 6.2 (1981), pp. 144–148.

[74] W. Evan Johnson, Cheng Li, and Ariel Rabinovic. “Adjusting batch effects in microarray expression data using empirical Bayes methods”. In: *Biostatistics* 8.1 (2007), pp. 118–127.

[75] M. Kathleen Kerr, Mitchell Martin, and Gary A. Churchill. “Analysis of variance for gene expression microarray data”. In: *Journal of computational biology* 7.6 (2000), pp. 819–837.

[76] Sookjeong Kim, Jong Kyoung Kim, and Seungjin Choi. “Independent arrays or independent time courses for gene expression time series data analysis”. In: *Neurocomputing* 71.10-12 (2008), pp. 2377–2387.

[77] Krzysztof C. Kiwiel. *Methods of descent for nondifferentiable optimization*. Vol. 1133. Springer, 1985.

[78] Wei Kong et al. “A review of independent component analysis application to microarray gene expression data”. In: *Biotechniques* 45.5 (2008), pp. 501–520.

[79] Daniel Kressner, Martin Plešinger, and Christine Tobler. “A preconditioned low-rank CG method for parameter-dependent Lyapunov matrix equations”. In: *Numerical Linear Algebra with Applications* 21.5 (2014), pp. 666–684.

[80] Piyush Kumar, Joseph S.B. Mitchell, and E. Alper Yildirim. “Approximate minimum enclosing balls in high dimensions using core-sets”. In: *Journal of Experimental Algorithmics (JEA)* 8 (2003).

[81] Piyush Kumar, Joseph S.B. Mitchellf, and E. Alper Yıldırıms. "Computing Core-Sets and Approximate Smallest Enclosing HyperSpheres in High Dimensions". In: *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments*. Vol. 111. SIAM. 2003, p. 45.

[82] Peter W. Laird. "Cancer epigenetics". In: *Human molecular genetics* 14.suppl_1 (2005), R65–R76.

[83] Thomas Larsson, Gabriele Capannini, and Linus Källberg. "Parallel computation of optimal enclosing balls by iterative orthant scan". In: *Computers & Graphics* 56 (2016), pp. 1–10.

[84] C.L. Lawson. "The smallest covering cone or sphere (C. Groenewod and L. Eusanio)". In: *SIAM Review* 7.3 (1965), p. 415.

[85] Cosmin Lazar et al. "Batch effect removal methods for microarray gene expression data integration: a survey". In: *Briefings in bioinformatics* 14.4 (2013), pp. 469–490.

[86] Daniel D. Lee and H. Sebastian Seung. "Algorithms for Non-negative Matrix Factorization". In: *Advances in Neural Information Processing Systems 13*. Ed. by T.K. Leen, T.G. Dietterich, and V. Tresp. MIT Press, 2001, pp. 556–562.

[87] Daniel D. Lee and Sebastian H. Seung. "Learning the parts of objects by non-negative matrix factorization". In: *Nature* 401.6755 (Oct. 1999), pp. 788–791.

[88] Su-In Lee and Serafim Batzoglou. "Application of independent component analysis to microarrays". In: *Genome biology* 4.11 (2003), R76.

[89] John A. Lee et al. "Type 1 and 2 mixtures of Kullback-Leibler divergences as cost functions in dimensionality reduction based on similarity preservation". In: *Neurocomputing* 112 (2013). Advances in artificial neural networks, machine learning, and computational intelligence, Selected papers from the 20th European Symposium on Artificial Neural Networks (ESANN 2012), pp. 92–108.

[90] Jeffrey T. Leek and John D. Storey. "Capturing Heterogeneity in Gene Expression Studies by Surrogate Variable Analysis". In: *PLoS Genet* 3.9 (Sept. 2007), e161.

[91] Jeffrey T. Leek et al. "Tackling the widespread and critical impact of batch effects in high-throughput data." In: *Nat Rev Genet* 11.10 (Oct. 2010), pp. 733–739.

[92] Ye Li and K.J. Ray Liu. "Adaptive blind source separation and equalization for multiple-input/multiple-output systems". In: *IEEE Transactions on Information Theory* 44.7 (1998), pp. 2864–2876.

[93] Wolfram Liebermeister. "Linear modes of gene expression determined by independent component analysis". In: *Bioinformatics* 18.1 (2002), pp. 51–60.

[94] Eric F. Lock et al. "Joint and individual variation explained (JIVE) for integrated analysis of multiple data types". In: *The annals of applied statistics* 7.1 (2013), p. 523.

[95] Sherene Loi et al. "Definition of clinically distinct molecular subtypes in estrogen receptor–positive breast carcinomas through genomic grade". In: *Journal of clinical oncology* 25.10 (2007), pp. 1239–1246.

[96] J. Luo et al. "A comparison of batch effect removal methods for enhancement of prediction performance using MAQC-II microarray gene expression data". In: *The pharmacogenomics journal* 10.4 (2010), pp. 278–291.

[97] Tim Marrinan. *A dual subgradient approach to computing an optimal rank Grassmannian circumcenter*. Personnal communication. 2019.

[98] Ann-Marie Martoglio et al. "A decomposition model to track gene expression signatures: preview on observer-independent classification of ovarian cancer". In: *Bioinformatics* 18.12 (2002), pp. 1617–1624.

[99] Chen Meng et al. "A multivariate approach to the integration of multi-omics datasets". In: *BMC bioinformatics* 15.1 (2014), p. 162.

[100] Wolfgang Meyer. *Toponogov's theorem and applications*. Ed. by College on Differential Geometry (Trieste). Lecture notes. 1989.

[101] Lance D. Miller et al. "An expression signature for p53 status in human breast cancer predicts mutation status, transcriptional effects, and patient survival". In: *Proceedings of the National Academy of Sciences of the United States of America* 102.38 (2005), pp. 13550–13555.

[102] Andy J. Minn et al. "Lung metastasis genes couple breast tumor size and metastatic spread". In: *Proceedings of the National Academy of Sciences* 104.16 (2007), pp. 6740–6745.

[103] Azam Moosavi, Razvan Stefanescu, and Adrian Sandu. "Efficient construction of local parametric reduced order models using machine learning techniques". In: *arXiv preprint arXiv:1511.02909* (2015).

[104] Esmaeel Moradi and Morteza Bidkhori. "Single Facility Location Problem". In: *Facility Location: Concepts, Models, Algorithms and Case Studies*. Ed. by Reza Zanjirani Farahani and Masoud Hekmatfar. Heidelberg: Physica-Verlag HD, 2009, pp. 37–68.

[105] Kary Mullis et al. "Specific enzymatic amplification of DNA in vitro: the polymerase chain reaction". In: *Cold Spring Harbor symposia on quantitative biology*. Vol. 51. Cold Spring Harbor Laboratory Press. 1986, pp. 263–273.

[106] Frank Nielsen and Gaëtan Hadjeres. "Approximating covering and minimum enclosing balls in hyperbolic geometry". In: *International Conference on Geometric Science of Information*. Springer. 2015, pp. 586–594.

[107] Frank Nielsen and Richard Nock. "Approximating smallest enclosing balls with applications to machine learning". In: *International Journal of Computational Geometry & Applications* 19.05 (2009), pp. 389–414.

[108] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Second. Springer Series in Operations Research and Financial Engineering. New York: Springer, 2006.

[109] Vegard Nygaard, Einar Andreas Rødland, and Eivind Hovig. "Methods that remove batch effects while retaining group differences may lead to exaggerated confidence in downstream analyses". In: *Biostatistics* 17.1 (2016), pp. 29–39.

[110] Hilary S. Parker and Jeffrey T. Leek. "The practical effect of batch on genomic prediction". In: *Statistical applications in genetics and molecular biology* 11.3 (2012).

[111] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[112] Peter Petersen. *Riemannian geometry*. Vol. 171. Springer, 2016.

[113] Frank Plastria. "Continuous covering location problems". In: *Facility location: applications and theory* 1 (2002), pp. 37–79.

[114] Vasiliki Plerou et al. "Random matrix approach to cross correlations in financial data". In: *Physical Review E* 65.6 (June 2002), pp. 066126+.

[115] Elijah Polak. *Computational methods in optimization: a unified approach*. Vol. 77. Academic press, 1971.

[116] Boris T Polyak. "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17.

[117] Sri Priya Ponnapalli et al. "A higher-order generalized singular value decomposition for comparison of global mRNA expression from multiple organisms". In: *PloS one* 6.12 (2011), e28072.

[118] Vijay K. Ramanan et al. "Pathway analysis of genomic data: concepts, methods, and prospects for future development". In: *TRENDS in Genetics* 28.7 (2012), pp. 323–332.

[119] Emilie Renard and P.-A. Absil. "Comparison of location-scale and matrix factorization batch effect removal methods on gene expression datasets". In: *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 2017, pp. 1511–1518.

[120] Emilie Renard, P.-A. Absil, and Kyle Gallivan. "Minimax center to extract a common subspace from multiple datasets". In: *27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning ,* 2019, pp. 275–280.

[121] Emilie Renard, Samuel Branders, and P.-A. Absil. "Independent component analysis to remove batch effects from merged microarray datasets". In: *Algorithms and Bioinformatics - 16th International Workshop*. Ed. by Martin C. Frith and Christian Nørgaard Storm Pedersen. Vol. 9838. Lecture Notes in Bioinformatics. Springer, 2016, pp. 281–292.

[122] Emilie Renard, Pierre Dupont, and Michel Verleysen. "User control for adjusting conflicting objectives in parameter-dependent visualization of data". In: *VAMP 2013 (EuroVis 2013), Visual Analytics using Multidimensional Projections Workshop*. Leipzig (Germany), June 2013.

[123] Emilie Renard, Kyle A. Gallivan, and P.-A. Absil. "A Grassmannian Minimum Enclosing Ball Approach for Common Subspace Extraction". In: *Latent Variable Analysis and Signal Separation*. Vol. 10891. Lecture Notes in Computer Science. Springer International Publishing, 2018, pp. 69–78.

[124] Emilie Renard, Andrew E. Teschendorff, and P.-A Absil. "Capturing confounding sources of variation in DNA methylation data by spatiotemporal independent component analysis". In: *22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN2014)*. 2014, pp. 195–200.

[125] Emilie Renard, Andrew E. Teschendorff, and Pierre-Antoine Absil. "Spatiotemporal ICA improves the selection of differentially expressed genes". In: *24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN2016)*. 2016, pp. 301–306.

[126] Emilie Renard et al. "Failing Xpert MTB/RIF modules: When? What warranty is needed from the customer's perspective? Experience in the DRC". In: *The International Journal of Tuberculosis and Lung Disease* 19 (2015), S206.

[127] Quentin Rentmeesters, P.-A. Absil, and Paul Van Dooren. "A filtering technique on the Grassmann manifold". In: *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems (MTNS)*. Vol. 5. 9. 2010.

[128] R. Tyrrell Rockafellar and Roger J.-B. Wets. *Variational analysis*. Vol. 317. Springer Science & Business Media, 2009.

[129] Florian Rohart et al. "MINT: A multivariate integrative method to identify reproducible molecular signatures across independent experiments and platforms". In: *BMC bioinformatics* 18.1 (2017), p. 128.

[130] Jason Rudy and Faramarz Valafar. "Empirical comparison of cross-platform normalization methods for gene expression data". In: *BMC bioinformatics* 12.1 (2011), p. 467.

[131] Douglas N. Rutledge and D. Jouan-Rimbaud Bouveresse. "Independent components analysis with the JADE algorithm". In: *TrAC Trends in Analytical Chemistry* 50 (2013), pp. 22–32.

[132] Matthieu Sainlez, P.-A. Absil, and Andrew E. Teschendorff. "Gene expression data analysis using spatiotemporal blind source separation." In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 2009, pp. 159–164.

[133] Takashi Sakai. *Riemannian geometry*. Vol. 149. American Mathematical Society, 1996.

[134] Ignacio Santamaria et al. "An order fitting rule for optimal subspace averaging". In: *Statistical Signal Processing Workshop (SSP), 2016 IEEE*. IEEE. 2016, pp. 1–4.

[135] Andrey A. Shabalin et al. "Merging two gene-expression studies via cross-platform normalization". In: *Bioinformatics* 24.9 (2008), pp. 1154–1160.

[136] Andrew H. Sims et al. "The removal of multiplicative, systematic bias allows integration of breast cancer gene expression datasets–improving meta-analysis and prediction of prognosis". In: *BMC medical genomics* 1.1 (2008), p. 1.

[137] Nguyen Thanh Son and Tatjana Stykel. "Solving parameter-dependent Lyapunov equations using the reduced basis method with application to parametric model order reduction". In: *SIAM Journal on Matrix Analysis and Applications* 38.2 (2017), pp. 478–504.

[138] Charlotte Soneson, Sarah Gerster, and Mauro Delorenzi. "Batch effect confounding leads to strong bias in performance estimates obtained by cross-validation". In: *PloS one* 9.6 (2014), e100335.

[139] Christos Sotiriou et al. "Gene expression profiling in breast cancer: understanding the molecular basis of histologic grade to improve prognosis". In: *Journal of the National Cancer Institute* 98.4 (2006), pp. 262–272.

[140] D. Stekel. *Microarray Bioinformatics*. Cambridge University Press, 2003.

[141] Charles J. Stone et al. "Spatiotemporal independent component analysis of event-related fMRI data using skewed probability density functions". In: *NeuroImage* 15.2 (Feb. 2002), pp. 407–421.

[142] John D. Storey and Robert Tibshirani. "Statistical significance for genomewide studies". In: *Proceedings of the National Academy of Sciences* 100.16 (2003), pp. 9440–9445.

[143] John D Storey et al. "Significance analysis of time course microarray experiments". In: *Proceedings of the National Academy of Sciences* 102.36 (2005), pp. 12837–12842.

[144] James Joseph Sylvester. "A question in the geometry of situation". In: *Quarterly Journal of Pure and Applied Mathematics* 1 (1857).

[145] Jonatan Taminau et al. "Comparison of merging and meta-analysis as alternative approaches for integrative gene expression analysis". In: *ISRN bioinformatics* 2014 (2014).

[146] Arthur Tenenhaus and Michel Tenenhaus. "Regularized generalized canonical correlation analysis". In: *Psychometrika* 76.2 (2011), pp. 257–284.

[147] Arthur Tenenhaus et al. "Variable selection for generalized canonical correlation analysis". In: *Biostatistics* 15.3 (2014), pp. 569–583.

[148] Michel Tenenhaus, Arthur Tenenhaus, and Patrick JF Groenen. "Regularized Consensus PCA". In: *arXiv preprint arXiv:1504.07005* (2015).

[149] Andrew E. Teschendorff, Emilie Renard, and Pierre A. Absil. "Supervised Normalization of Large-Scale Omic Datasets Using Blind Source Separation". In: *Blind Source Separation*. Ed. by Ganesh R. Naik and Wenwu Wang. Signals and Communication Technology. Springer Berlin Heidelberg, 2014, pp. 465–497.

[150] Andrew E. Teschendorff, Joanna Zhuang, and Martin Widschwendter. "Independent surrogate variable analysis to deconvolve confounding factors in large-scale microarray profiling studies." In: *Bioinformatics* 27.11 (2011), pp. 1496–1505.

[151] Andrew E. Teschendorff et al. "Age-dependent DNA methylation of genes that are suppressed in stem cells is a hallmark of cancer". In: *Genome Research* 20.4 (2010), pp. 440–446.

[152] Fabian J. Theis et al. "Spatiotemporal Blind Source Separation Using Double-Sided Approximate Joint Diagonalization". In: *In Proc. EUSIPCO 2005*. 2005, pp. 1–4.

[153] Natalie P. Thorne et al. "13 DNA Methylation Arrays: Methods and Analysis". In: *Microarray Innovations: Technology and Experimentation* (2008), p. 175.

[154] Roberto Tron, Bijan Afsari, and René Vidal. "Riemannian consensus for manifolds with bounded curvature". In: *IEEE Transactions on Automatic Control* 58.4 (2013), pp. 921–934.

[155] Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. "Core vector machines: Fast SVM training on very large data sets". In: *Journal of Machine Learning Research* 6.Apr (2005), pp. 363–392.

[156] Marc J. Van De Vijver et al. "A gene-expression signature as a predictor of survival in breast cancer". In: *New England Journal of Medicine* 347.25 (2002), pp. 1999–2009.

[157] Laura J. Van't Veer et al. "Gene expression profiling predicts clinical outcome of breast cancer". In: *Nature* 415.6871 (2002), pp. 530–536.

[158] Javier Via, Ignacio Santamaria, and Jesús Pérez. "Canonical correlation analysis (CCA) algorithms for multiple data sets: Application to blind SIMO equalization". In: *2005 13th European Signal Processing Conference*. IEEE. 2005, pp. 1–4.

[159] Dennis Wackerly, William Mendenhall, and Richard L. Scheaffer. *Mathematical statistics with applications*. Thomson Learning, 2008.

[160] Yixin Wang et al. "Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer". In: *The Lancet* 365.9460 (2005), pp. 671–679.

[161] Zhong Wang, Mark Gerstein, and Michael Snyder. "RNA-Seq: a revolutionary tool for transcriptomics". In: *Nature reviews genetics* 10.1 (2009), p. 57.

[162] Patrick Warnat, Roland Eils, and Benedikt Brors. "Cross-platform analysis of cancer microarray data improves gene expression based classification of phenotypes". In: *BMC bioinformatics* 6.1 (2005), p. 1.

[163] Gary Weickum, Mike Eldred, and Kurt Maute. "Multi-point extended reduced order modeling for design optimization and uncertainty analysis". In: *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2006, p. 2145.

[164] Johan A. Westerhuis, Theodora Kourti, and John F. MacGregor. "Analysis of multiblock and hierarchical PCA and PLS models". In: *Journal of chemometrics* 12.5 (1998), pp. 301–321.

[165] Wikipedia contributors. *DNA methylation — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=DNA_methylation&oldid=888248990`. [Online; accessed 18-April-2019]. 2019.

[166] Wikipedia contributors. *Gene expression — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Gene_expression&oldid=889205398`. [Online; accessed 18-April-2019]. 2019.

[167] Wikipedia contributors. *Gene regulatory network — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Gene_regulatory_network&oldid=891963017`. [Online; accessed 19-April-2019]. 2019.

[168] Wikipedia contributors. *Genomics — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Genomics&oldid=888612224`. [Online; accessed 18-April-2019]. 2019.

[169] Herman Wold. "Partial least squares". In: *Encyclopedia of statistical sciences* 6 (1985), pp. 581–591.

[170] Yung-Chow Wong. "Differential geometry of Grassmann manifolds". In: *Proceedings of the National Academy of Sciences* 57.3 (1967), pp. 589–594.

[171] Haleh Yasrebi. "Comparative study of joint analysis of microarray gene expression data in survival prediction and risk assessment of breast cancer patients". In: *Briefings in bioinformatics* 17.5 (2015), pp. 771–785.

[172] Ke Ye and Lek-Heng Lim. "Schubert varieties and distances between subspaces of different dimensions". In: *SIAM Journal on Matrix Analysis and Applications* 37.3 (2016), pp. 1176–1197.

[173] Jin Yu et al. "A quasi-Newton approach to nonsmooth convex optimization problems in machine learning". In: *Journal of Machine Learning Research* 11 (2010), pp. 1145–1200.

[174] Joanna Zhuang, Martin Widschwendter, and Andrew E. Teschendorff. "A comparison of feature selection and classification methods in DNA methylation studies using the Illumina Infinium platform". In: *BMC Bioinformatics* 13.1 (2012), p. 59.

[175] Andrei Zinovyev et al. "Blind source separation methods for deconvolution of complex signals in cancer biology". In: *Biochemical and biophysical research communications* 430.3 (2013), pp. 1182–1187.