Statistical learning applied to cardiology : discriminative clustering and aortic stenosis phenogroups

Louis Ohl

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

UNIVERSITÉ LAVAL

THÈSE DE DOCTORAT

Apprentissage statistique appliqué à la cardiologie

Clustering discriminant et phénogroupes de la sténose aortique

Louis OHL

Laboratoire d'Informatique, de Signaux et Systèmes de Sophia Antipolis (I3S)
UMR7271 UCA CNRS

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur **et de** Université Laval

Dirigée par : Frédéric PRECIOSO, Professeur, Université Côte d'Azur
Co-dirigée par : Arnaud DROIT, Professeur, Université Laval
Co-encadrée par : Pierre-Alexandre MATTEI, Chercheur, Inria
Soutenue le : 22 Mai 2024

Devant le jury, composé de :
Maxime SERMESANT, Directeur de recherche, Inria
Christophe BIERNACKI, Directeur de recherche, Université Lille 1
Jakob VERBEEK, Chercheur, FAIR, META
Julie HUSSIN, Professeur adjointe, Institut de cardiologie de Montréal
Théo PEZEL, Chef de clinique assistant, Hôpital Lariboisière, département de cardiologie

APPRENTISSAGE STATISTIQUE APPLIQUÉ À LA CARDIOLOGIE

CLUSTERING DISCRIMINANT ET PHÉNOGROUPES DE LA STÉNOSE AORTIQUE

Statistical learning applied to cardiology

Discriminative clustering and aortic stenosis phenogroups

Louis OHL

⋈

Jury :

Président du jury
Maxime SERMESANT, Directeur de recherche, Inria

Rapporteurs
Christophe BIERNACKI, Directeur de recherche, Université Lille 1
Jakob VERBEEK, Chercheur, FAIR, META

Examinateurs
Julie HUSSIN, Professeur adjointe, Institut de cardiologie de Montréal
Théo PEZEL, Chef de clinique assistant, Hôpital Lariboisière, département de cardiologie

Directeur de thèse
Frédéric PRECIOSO, Professeur, Université Côte d'Azur

Co-directeur de thèse
Arnaud DROIT, Professeur, Université Laval

Co-encadrant de thèse
Pierre-Alexandre MATTEI, Chercheur, Inria

Université Côte d'Azur

Louis OHL
Apprentissage statistique appliqué à la cardiologie
Clustering discriminant et phénogroupes de la sténose aortique
xviii+211 p.

À Roger Dusséaux, dont le coeur tacite insuffle mes réflexions

Mathematics has two faces : it is the rigorous science of Euclid, but it is also something else. Mathematics presented in the Euclidean way appears as a systemic, deductive science ; but mathematics in the making appears as an experimental, inductive science. Both aspects are as old as the science of mathematics itself.

— George Pólya, How to Solve it.

Apprentissage statistique appliqué à la cardiologie

Résumé

La sténose de la valve aortique (SA) est une maladie chronique progressive dont la prévalence risque de tripler dans les décennies à venir en Amérique du Nord et par conséquent ses impacts en santé et économie. À l'heure actuelle, aucun médicament contre la SA n'est disponible. La nécessité de pharmacothérapies adaptées pousse donc à l'exploration des différentes causes de la progression de la SA chez les patients. Bien qu'il existe déjà certaines sous-catégories de la SA, ces dernières sont difficiles à identifier et par conséquent à cibler par une thérapie.

Afin de découvrir et identifier des causes potentielles de la SA, nous formulons la recherche de ces phénogroupes en tant que problème de partitionement. Le partitionnement est un problème issu du domaine d'apprentissage automatique consistant à répartitr de multiples observations en groupes nommés *clusters* selon leurs similarités. Afin d'accompagner ce problème d'apprentissage automatique, nous utilisons l'étude sur le progression des déterminants métaboliques de la SA (étude PROGRESSA). L'étude PROGRESSA comprend trois modalités: clinicopathologique, protéomique et radiomique pour 351 patients avec suivi annuel. La structure de PROGRESSA est complexe: elle est de grande dimension avec des variables de natures différentes. De plus, les différentes modalités ne se recouvrent pas nécessairement.

Dans ce contexte, nous formulons le problème de partitionnement à travers un prisme discriminatif, ce qui permet d'intégrer avec facilité des modèles d'apprentissage profond, notamment pour manipuler des données grande dimensions. Ces dernières années ont été marquées par l'arrivée de méthodes de partitionnement profonds, souvent basés sur la maximisation de l'information mutuellee. Cependant, les récents succès de ces méthodes sont souvent spécifique à un type unique de données et ne permettent donc pas d'anticiper leur applicabilité à un problème multi-source.

Afin de construire une solution pour le problème de partitionnement multi-source, cette thèse s'orchestre autour du développement d'un ensemble de méthodes de clustering nommé information mutuelle généralisée (GEMINI) à partir du Chapitre 3. Cet ensemble de méthodes permet d'utiliser n'importe quelle architecture de réseau de neurones profonds sur des données de natures variées. Nous montrons également comment cette méthode peut être améliorée pour incorporer des méchanismes de sélections de vaiables afin de faciliter l'interprétation des clusters au Chapitre 4: Sparse GEMINI. Puis nous complètons le spectre des modèles entraînables par GEMINI avec l'introduction d'arbres non supervisés donnant un clustering avec explication intégrée dans le chapitre 5.

Enfin, nous terminons cette thèse avec un pipeline intégrant divers variants de GEMINI pour la découverte de phénogroupes de la SA dans l'étude PROGRESSA au Chapitre 6. Certains de ces phénogroupes montrent une mortalité accentuée et sont caractérisés par des marqueurs spécifiques, par exemple liés aux lipoprotéines, au diabète ou à la bicuspidie des valves aortiques. Ces phénogroupes peuvent ainsi être ciblés par des thérapies spécifiques afin de réduire le risque de progression de la maladie.

Mots-clés : Partitionnement discriminatif, apprentissage non supervisé, cardiologie, sténose aortique, phénogroupes

Statistical learning applied to cardiology

Abstract

Aortic valve stenosis (AS) is a chronic progressive disease whose prevalence is likely to triple in the coming decades in North America, with a consequent impact on health and the economy. However, efficient drug therapies for this disease are not available. The need for appropriate medication is therefore driving the exploration of the various causes of AS progression in patients. There exist a few sub-categories of the disease that could be differently targeted by drugs, but they are hard to define and identify.

To alleviate the finding of different possible causes of AS, we formulate the search of phenogroup (i.e. disease subtypes) as a clustering problem. Clustering is a family of approaches from machine learning that consists in gathering multiple observations deemed similar in categories called clusters. To support this machine learning problem instance, we employ the metabolic determinants of the progression of AS study (PROGRESSA study). The PROGRESSA dataset comprises 3 modalities: clinicopathological, proteomics and radiomics data for 351 patients with yearly follow-ups. The structure of the PROGRESSA study is challenging for current clustering algorithms: it is high-dimensional with mixed data types. Moreover, the different modalities of the data do not necessarily overlap, making it to a multi-source clustering problem.

In this context, we formulate the clustering problem through the lens of discriminative clustering: a point of view that leverages the easy integration of deep learning models for handling and concatenating high-dimensional data. Within this framework, the last decade witnessed the impressive rise of deep clustering methods that often involves the maximisation of mutual information. However, the recent success of deep clustering models are often over-specified for one type of data and therefore hardly account for multi-modal data.

To pave the way for a multi-source discriminative clustering algorithm, we developed a set of discriminative clustering methods called generalised mutual information (GEMINI) in Chapter 3. Thanks to its discriminative construction, this set of methods can be used with any deep neural network architecture on data of various types. We also show how this method can be improved to incorporate variable selection mechanisms to facilitate the interpretation of clusters in Chapter 4: Sparse GEMINI. Then, we complete the spectrum of models trainable by GEMINI in Chapter 5 with the introduction of unsupervised trees giving a clustering with integrated explanation.

Finally, we conclude this thesis in Chapter 6 with a pipeline integrating various GEMINI variants for the discovery of AS phenogroups in the PROGRESSA study. Some of these phenogroups show increased mortality and are characterised by specific markers, for example linked to lipoproteins, diabetes or bicuspid aortic valves. These phenogroups can therefore be targeted by specific therapies to reduce the risk of disease progression.

Keywords: Discriminative clustering, unsupervised learning, cardiology, aortic stenosis, phenogroups

Remerciements

Admittedly, I rarely feel as overwhelmed as now writing those couple lines. I feel like it would there's not enough words to express the gratitude I feel towards everybody whom I met during this path. This journey across France and Canada would definitely not have been the same without any of you, through shades of theater, scuba diving, music and delicious ice creams. Let's start this journey again:

I would like to thank very deeply all of my supervisors, Frédéric Precioso, Pierre-Alexandre Mattei, Arnaud Droit and Mickaël Leclercq for their patience, the example they set in their domain and overall those fantastic years we had together. It has been pleasant to work with you during the discovering journey that is a PhD, and I feel like hardly any amount of thanks would illustrate how grateful I feel: from the joys of discovery, the curiosity of our questions, the stressful adversity of the administrative conundra and the care for excellent writing. I am delighted that our three years together was a nonlinear path sprinkled with smiles.

Of course, I can't speak about smiling without mentioning the amazing Maasai team that I have cherished for the past 3 years, and slightly more as an intern before. You have been a wonderful environment for working, learning about science and beyond: just being yourself. Discussing with any of you has always been pleasant, whether around beer, skying, kayaking, hiking, or plainly taking a coffee. So thank you all for being there in this journey: Stéphane the talented guitarist, Giulia the unbeatable beach volley player, Hugo the mountain man, Kévin the card master, Mansour the penguin-eraser office-mate, Cédric the optimal "psytrans"-port scientist, Rémy the "How-are-the-penguins" inspector, Gianluigi my expert in italian translation, and so many more!

Still around the lab, I would like to thank also my wonderful band mates with whom I spent amazing time playing music from simple rock to some of my weird ideas. So thanks a lot to Thibaud, Mansi, Gabriele, Vasiliki, Santiago and Stef (again) from Bandito, then Antoine, Audrey, Victor and Mathieu from Fondamental. Still on the music, with a silly note (pun intended), I would like to thank the following nonexhaustive list of some awesome artists that accompanied the PhD: Tom Cardy, The Pineapple Thief, Syncatto, Sylvan, Porcupine Tree, Leprous, Dream Theater, Haken, TesseracT, Animals as Leaders, Meshuggah, Fleshgod Apocalypse.

Outside of the lab lies a town called Nice where I happened to spend too many nights with my dear fellows of adventures. Nights would sure have been different if it was not for your permanent happiness, your care and warming friendship: Chacha, Robi, Ricou, Laulau, Steps (you're everywhere dude, it's the third time you appear) and the surrounding Lisboa team, you're the best!

With some roots tied to Lyon, I would like also to share all of my thanks to the support of the Tuesday team, one of the most insane and chill team of friends I ever deserved to have. Bastien, Matthis, Victor, Paul, Maxime, your place belongs deep in my memories. Strong kuddos go as well to my computer science buddy of INSa Lyon, Amine!

Mentioning quickly my engineering school leads me to think back on my theatrical hobbies. I am glad to have found the theater group of the CNRS at Cote d'Azur, the "Compagnie Sun7". It was a real pleasure to perform with all of you up to the Oléron island: Isabelle, Camille, Corinne, Yolande, Landry and Romain. Mentioning Isabelle, I thank you very much for encouraging me to

participate to the “Ma thèse en 180 secondes”, the contest was all laughs and a good show! Cheers to the MT180 crew, you are an exceptional team!

Crossing the Atlantic, I would like to thank the entire ADLab for their warm welcome in the growing cold of the Québec fall. The months we spent together were full of discovery whether in biology, hikes or Québec’s shades of poutines. For this marvellous explorations, I warmly thank Paul-Emmanuel, Milan, Elloise, Simon, Steven, Sophiane and Vivian. Of course, the experience would not be complete without my peculiar flatmates: Naheyma, Jonas and Loris!

Now that I come back to France, I will write in French. De retour sur la côte d’Azur, je tiens aussi à remercier chaleureusement Manu et David pour les très nombreux accueils qu’ils m’ont témoignés et le grand nombre de soirées passées ensemble à apprécier la vie le plus simplement possible en compagnie d’une excellente cuisine.

Et comme beaucoup de voyages se terminent en rentrant à la maison, je tiens à remercier très profondément ma famille pour tout son soutien. Vous m’avez encouragé à aller jusqu’au bout, sans nécessairement comprendre de quoi il en ressortait à chaque fois. Je suis très heureux d’avoir pu partager un bout de cette grande aventure avec vous. Votre place va plus profond que mon coeur, et je ne saurai dire s’il reste encore assez de chaleur pour vous affirmer le plus grand des mercis. Merci Papa, Maman, Blanche & Basile; je vous aime.

Contents

1 The challenges of aortic stenosis and the clustering of PROGRESSA **1**
- 1.1 Aortic stenosis . . . 2
  - 1.1.1 Definition . . . 2
  - 1.1.2 Diagnosis, prognosis and treatment . . . 2
  - 1.1.3 Expected challenges . . . 3
  - 1.1.4 Implementing machine learning in cardiology . . . 4
  - 1.1.5 The PROGRESSA study . . . 5
- 1.2 Clustering . . . 5
  - 1.2.1 Definition . . . 5
  - 1.2.2 Limitations for the PROGRESSA study . . . 6
- 1.3 Thesis outline . . . 7

Of clustering

2 Clustering **13**
- 2.1 Intuition for the PROGRESSA dataset . . . 13
- 2.2 Modelling frameworks . . . 14
  - 2.2.1 Starting from Bayes theorem . . . 14
  - 2.2.2 Generative models . . . 15
  - 2.2.3 Discriminative models . . . 18
- 2.3 The challenge of learning in discriminative clustering . . . 19
  - 2.3.1 The shortcomings of classical statistical tools . . . 19
  - 2.3.2 Objective functions in classification . . . 20
  - 2.3.3 Mutual information as a promising objective . . . 21
- 2.4 Other clustering models . . . 23
  - 2.4.1 K-means . . . 23
  - 2.4.2 Spectral clustering . . . 24
  - 2.4.3 Hierarchical clustering . . . 24
- 2.5 Thrive of mutual information: from clustering to representation learning . . . 25
  - 2.5.1 Early usage of mutual information for clustering . . . 25
  - 2.5.2 Towards deeper networks . . . 26
  - 2.5.3 From discrete to continuous output variables: contrastive learning and infoMax . . . 27
  - 2.5.4 Dissonance between MI and performances . . . 29
- 2.6 Interpreting the clusters . . . 30
  - 2.6.1 The curse of dimensionality . . . 30
  - 2.6.2 Variable selection . . . 30
  - 2.6.3 Projections and subspaces . . . 31
  - 2.6.4 Intrinsically interpretable models . . . 31

3 The generalised mutual information: a discriminative clustering framework 33
3.1 Introduction . . . 33
3.2 Is MI a good clustering objective? . . . 34
3.2.1 Entropy perspectives . . . 34
3.2.2 A practical example . . . 35
3.3 Extending the MI to the GEMINI . . . 37
3.3.1 The discriminative clustering framework for GEMINIs . . . 37
3.3.2 The One-vs-All GEMINI . . . 38
3.3.3 The One-vs-One GEMINI . . . 42
3.4 GEMINI in practice . . . 44
3.4.1 Geometric considerations . . . 44
3.4.2 Estimating a GEMINI . . . 45
3.4.3 Complexity considerations . . . 45
3.4.4 Further speed-ups for the OvO Wasserstein . . . 46
3.4.5 The *GemClus* package . . . 47
3.5 Experiments . . . 48
3.5.1 When MI fails because of the modelling . . . 48
3.5.2 Resistance to outliers . . . 49
3.5.3 Leveraging a manifold geometry . . . 50
3.5.4 Fitting MNIST . . . 52
3.5.5 Number of non-empty clusters and architecture . . . 53
3.5.6 Number of clusters and training time . . . 54
3.5.7 Cifar10 clustering using a SIMCLR-derived kernel . . . 55
3.5.8 A practical application with Graph Neural Networks: the Enron email dataset 56
3.6 Conclusion . . . 57

4 Sparsifying the generalised mutual information for joint feature selection and discriminative clustering 61
4.1 Introduction . . . 62
4.2 Sparse GEMINI . . . 62
4.2.1 Unsupervised logistic regression architecture . . . 62
4.2.2 The LassoNet architecture . . . 63
4.3 Optimisation . . . 64
4.3.1 Training and model selection . . . 64
4.3.2 Extension proposal: the dynamic training regime . . . 65
4.3.3 Gradient considerations . . . 65
4.3.4 Implementation in GemClus . . . 65
4.4 Experiments . . . 66
4.4.1 Metrics . . . 66
4.4.2 Default hyperparameters . . . 66
4.4.3 Numerical experiments . . . 67
4.4.4 Examples on MNIST and variations . . . 71
4.4.5 Real datasets . . . 73
4.4.6 Discussion . . . 75
4.5 Conclusion . . . 76

5 From neural networks to trees: explainable discriminative clustering with Kauri and Douglas **79**
5.1 Introduction . . . 79
5.2 Training trees . . . 80
5.2.1 How do we train supervised trees? . . . 81
5.2.2 How do we train unsupervised trees? . . . 81
5.2.3 Related motivating example . . . 82
5.3 Kauri: K-means as unsupervised reward ideal . . . 82
5.3.1 Notations and modelling . . . 83
5.3.2 Objective . . . 83
5.3.3 Tree branching . . . 87
5.3.4 Gain metrics . . . 87
5.4 A fast implementation for Kauri . . . 91
5.4.1 Pre-computing kernel stocks . . . 91
5.4.2 Optimising split evaluation . . . 91
5.4.3 An iterative rule for split stocks . . . 93
5.4.4 Complete picture . . . 94
5.4.5 Implementation in GemClus . . . 94
5.5 Douglas: DNDTs optimised using GEMINI leverage apprised splits . . . 97
5.5.1 The Douglas model . . . 97
5.5.2 Implementation in GemClus . . . 98
5.6 Experiments . . . 98
5.6.1 Performances with as many leaves as clusters . . . 100
5.6.2 Performances with more leaves than clusters . . . 101
5.6.3 Varying the kernel . . . 108
5.6.4 Pure numpy Douglas performances . . . 108
5.6.5 A qualitative example of the obtained decision tree . . . 109
5.7 Conclusion . . . 110

Applying clustering for aortic stenosis

6 A GEMINI pipeline for the identification of phenogroups of aortic stenosis **115**
6.1 Introduction . . . 115
6.2 Known phenogroups . . . 116
6.3 Methods . . . 117
6.3.1 PROGRESSA modalities and datasets . . . 117
6.3.2 Preprocessing . . . 120
6.3.3 Applying multiple Sparse GEMINI models . . . 120
6.3.4 Variables ranking . . . 121
6.3.5 Accuracy filtering for time stability . . . 121
6.3.6 Consensus clustering . . . 121
6.4 From clusters to phenogroups . . . 123
6.4.1 Results and metrics . . . 123
6.4.2 Identifying phenogroups . . . 124
6.5 Conclusion . . . 133

7 Final words, ongoing and future works **135**
7.1 Overview of the thesis contributions . . . 135
7.1.1 Discriminative clustering . . . 135
7.1.2 Phenogroups of aortic stenosis . . . 135
7.2 Ongoing work . . . 135
7.2.1 Adding supervision to consensus clustering . . . 135
7.2.2 Towards heterogenous source clustering . . . 136
7.3 Perspectives and future work . . . 137
7.3.1 Integration of radiomics in PROGRESSA clustering . . . 137
7.3.2 Exploiting pretrained model for clustering . . . 137
7.3.3 Learning metrics . . . 138

Bibliography **139**

List of Figures **159**

List of Tables **161**

Annexes

A Proof of the mutual information convergence for the separation of two Gaussian distributions **167**
A.1 Mutual information of the good boundary model . . . 167
A.2 Mutual information of the misplaced boundary model . . . 169
A.3 Differences of mutual information . . . 170

B Proofs for the GEMINI propositions **173**
B.1 Proof of Prop. 3.3.1 . . . 173
B.2 Proof of Prop. 3.3.3 . . . 177
B.3 Proof of Prop. 3.3.4 . . . 179
B.4 Proof of Prop. 3.3.5 . . . 180

C Derivation and gradients of GEMINIs **183**
C.1 f-divergence GEMINI . . . 183
C.1.1 Generic f function . . . 183
C.1.2 Kullback-Leibler divergence . . . 189
C.1.3 Total Variation distance . . . 190
C.1.4 Squared Hellinger distance . . . 190
C.2 Maximum Mean Discrepancy . . . 192
C.2.1 OvA scenario . . . 192
C.2.2 OvO scenario . . . 195
C.3 Gradients for the Wasserstein-GEMINI . . . 197
C.3.1 Gradient for the Wasserstein distance . . . 197
C.3.2 Complete gradient for the OvA Wasserstein . . . 198
C.3.3 Complete gradient for the OvO Wasserstein . . . 198

D Implementing MMD-GEMINI using matrix multiplications **199**
D.1 OvA scenario 199
D.1.1 Alternative computation of the forward pass 199
D.1.2 Gradient 200
D.2 OvO scenario 204
D.2.1 Alternative computation of the forward pass 204
D.2.2 Gradient 204

Essential nomenclature

Algebra

a	Scalar
$\boldsymbol{a}$	Vector
$\boldsymbol{A}$	Matrix

Dimensions

n	Number of observations/samples
d	Number of variables or features per sample
K	Number of clusters

Sets

$\mathcal{D}$	Dataset
$\mathcal{X}$	Data space
$\mathcal{C}$	Cluster containing samples
$[\![n]\!]$	The set of natural integers from 1 to n
Δ^K	The K-simplex
$\mathcal{H}$	Reproducing kernel Hilbert space
$\mathcal{Z}$	An intermediate representation space
$\lvert\cdot\rvert$	Cardinal

Variables

$\boldsymbol{x}$	An observation from the dataspace $\mathcal{X}$
$\boldsymbol{x}_{ij}$	The j-th feature of the i-th observation in the dataset $\mathcal{D}$
y	The cluster membership
$\boldsymbol{z}$	A continuous variable from the intermediate space $\mathcal{Z}$

Probability theory

p_θ	Distribution with parameters θ
$\mathbb{E}$	Expectation of a random variable
$\mathbb{V}$	Variance of a random variable
$\mathbb{H}$	Entropy of a random variable
$\delta_{\boldsymbol{x}}$	Delta Dirac distribution located on the position $\boldsymbol{x}$
$\mathbb{1}$	Indicator function
$\mathcal{N}$	Multivariate and univariate Gaussian distribution
$\mathcal{I}$	Mutual information of any kind

Distances and norms

$\|\cdot\|$	Norm
$c(\cdot,\cdot)$	A distance function (or cost) between samples in the dataspace $\mathcal{X}$
$\kappa(\cdot,\cdot)$	A kernel function between samples in the dataspace $\mathcal{X}$
$D(\cdot\|\cdot)$	Any statistical distance or divergence between two distributions
D_{KL}	Kullback-Leibler divergence
D_{IPM}	Integral probability metric
D_{MMD}	Maximum mean discrepancy using the kernel κ
$D_{\mathcal{W}}$	Wasserstein-1 distance using the cost c
D_f	f-divergence
D_{H^2}	Squared Hellinger distance
D_{TV}	Total variation distance
$D_{\text{KL-sym}}$	Symmetric Kullback-Leibler divergence

CHAPTER 1

The challenges of aortic stenosis and the clustering of PROGRESSA

2.1 Intuition for the PROGRESSA dataset 13
2.2 Modelling frameworks 14
2.2.1 Starting from Bayes theorem 14
2.2.2 Generative models 15
2.2.3 Discriminative models 18
2.3 The challenge of learning in discriminative clustering 19
2.3.1 The shortcomings of classical statistical tools 19
2.3.2 Objective functions in classification 20
2.3.3 Mutual information as a promising objective 21
2.4 Other clustering models 23
2.4.1 K-means 23
2.4.2 Spectral clustering 24
2.4.3 Hierarchical clustering 24
2.5 Thrive of mutual information: from clustering to representation learning . 25
2.5.1 Early usage of mutual information for clustering 25
2.5.2 Towards deeper networks 26
2.5.3 From discrete to continuous output variables: contrastive learning and infoMax 27
2.5.4 Dissonance between MI and performances 29
2.6 Interpreting the clusters 30
2.6.1 The curse of dimensionality 30
2.6.2 Variable selection 30
2.6.3 Projections and subspaces 31
2.6.4 Intrinsically interpretable models 31

1.1 Aortic stenosis

1.1.1 Definition

As you read those first lines, a small amount of blood is passing through your beating heart. Eventually, blood will leave the organ and roam the body for another lap. When leaving, blood passes through one of the last cardiac chambers: the *lower left chamber*, also known as the *left ventricle*. There, a valve continuously closing and opening partakes to the pumping role of the heart. However, for some of us, and especially with most prevalence in high income countries (Coffey et al., 2021 ; Iung et al., 2019 ; Lindman et al., 2016), this valve may someday narrow or open poorly, thus affecting the quality of blood flow to the body. This disease is known as *aortic stenosis* (AS) and is estimated to affect 3 million individuals in North America. Among the elderly population (older than 75 years old), estimates range from 3.5% to 12.4% depending on the severity of AS in Europe and North America (Osnabrugge et al., 2013).

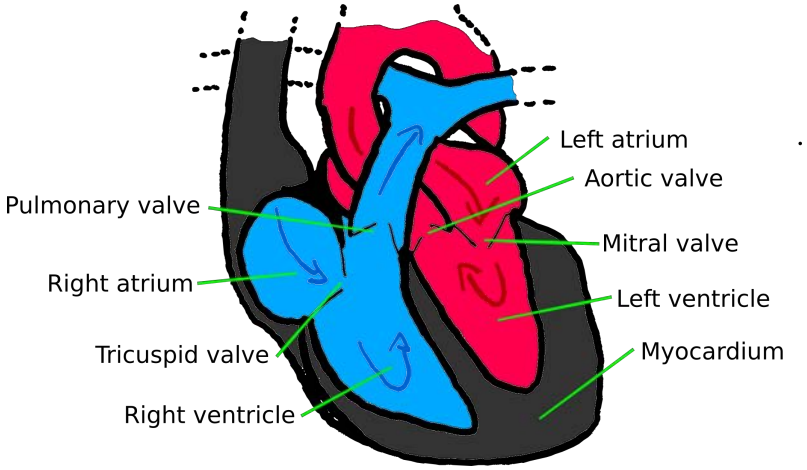


Figure 1.1 – Apical four chambers drawing of the heart. The aortic valve is located close to the mitral valve in the left ventricle.

The progression of this chronic disease is often classified into the following categories: *mild*, *moderate*, *severe* or *critical*. However, it is often difficult to detect the disease in a reasonable time. In fact, patients may live for years without symptoms of aortic stenosis before realising it, a moment at which immediate intervention is often required. Notably, according to Thoenes et al. (2018):

> "A diagnosis of severe symptomatic AS is associated with an average life expectancy of 2-3 years and necessitates a timely valvular intervention."

A non-exhaustive list of potential symptoms includes chest pain, fatigue, heart palpitations, or left ventricular hypertrophy (Thoenes et al., 2018). Without intervention, patients exhibit a higher mortality rate for the 4 following years (Généreux et al., 2023). Within this period, it is estimated that 62.7% of patients with moderate AS showing heart failure with reduced ejection fraction may suffer from an all-cause mortality or heart failure hospitalisation (Khan et al., 2023).

1.1.2 Diagnosis, prognosis and treatment

Echocardiography is the primary imaging modality used for diagnosing and assessing hemodynamic severity and progression rate of AS, which ultimately determines optimal timing for intervention (Beyersdorf et al., 2021 ; Otto et al., 2021). Notably, the European and US guidelines recommend aortic valve replacement (AVR) upon observation of an abnormal left ventricular ejection fraction, a rate of peak transvalvular velocity progression greater than 0.3 m/s per year or repeated and markedly elevated B-type natriuretic peptide (Sevilla, Revilla-Orodea, & San Román, 2021).

Certain risk factors can be associated with AS such as the body mass index (BMI), hypertension, alcohol and tobacco consumption or diabetes (Back & Larsson, 2020). However, these factors alone often remain insufficient to forecast the progression rate of the disease.

To treat patients affected by severe aortic stenosis, surgery remains the only therapeutic option with surgical AVR or transcatheter aortic valve implantation (TAVI) (Beyersdorf et al., 2021 ; Otto et al., 2021), as to this day no effective pharmacological treatment to prevent the development and/or progression of AS exists. This absence of validated drugs emerges from the lack of accurate identification of pathophysiological mechanisms to target. For instance, the predominant pathobiological process among women is valvular fibrosis (Simard et al., 2017 ; Tastet et al., 2020), whereas men are more prone to aortic valve tissue calcification. Younger patients are concerned by the infiltation and oxydation of lipids within valvular tissue and older patients rather suffer from the relationship between osteoporosis and ectopic calcificationc of soft tissues, also known as the calcification paradox (Persy & D'Haese, 2009).

Nonetheless, recent developments suggest the usage of ultrasound therapy for a safe and feasible treatment of calcific AS (Messas et al., 2023), and other pharmaceutical targets are currently being investigated.

1.1.3 Expected challenges

Today, the prevalence of AS is expected to increase substantially with the ageing of the population and its ensuing economic and health burden is expected to do the same (Coffey et al., 2021 ; Osnabrugge et al., 2013 ; Roth et al., 2020). Specifically, the prevalence of AS is expected to triple in the 30 coming years (Andell et al., 2017 ; Danielsen, Aspelund, Harris, & Gudnason, 2014 ; Lindman et al., 2016). Consequently, with a cost estimated up to 10.2 billion USD per year solely in North America between 1996 and 2011, AS has a dramatic impact on patients, society, and healthcare (Moore, Chen, Mallow, & Rizzo, 2016). At the individual level, studies estimate a cost of 11,000 USD per year per asymptomatic patient and 13,000 USD per year per symptomatic patient in the United States (Moore et al., 2016) or 28,000 EUR per patient in Italy at the time of intervention (Veronesi, Beccagutti, Corbo, Blini, & Degli Esposti, 2015). From 2000 to 2017, drastic increases in mortality were reported by Hartley et al. (2021) over 23 countries in the EU including UK, with smaller increases for western Europe than in eastern Europe.

However, despite the increasing use of TAVI (Pilgrim & Windecker, 2018), the unequal access to health facilities brings imbalance in mortality in favour of more urban and high-income population, *e.g.* Damluji et al. (2020). In this sense, countries that were able to establish a well-developed access to TAVI rather observed plateaus of mortality rates instead of increases, *e.g.* Germany or the Netherlands (Hartley et al., 2021). Yet, the access to surgery is not a sufficient criterion for the challenges of AS. Studies highlight denials of aortic valve replacement for 30% to 50% of patients

with even severe symptomatic AS, often due to overestimations of the risk of operation (D. S. Bach et al., 2009 ; Malouf et al., 2012 ; Morris et al., 1993). To quote Thaden, Nkomo, et Enriquez-Sarano (2014):

> "Outcome is known to be poor in symptomatic patients with severe AS, but, despite this knowledge, multiple studies have documented poor adherence to evidence-based guidelines and inappropriate denial of surgery [...]."

There is a dire need for progress in the screening, diagnosis and risk and treatment stratification throughout the world. Today, the core challenge of AS is the existence of almost no biomarkers and strategies for identifying the optimal intervention moment in the disease treatment. Indeed, Thaden et al. (2014) report that:

> "Because there are no effective therapies for AS, management relies on optimal timing for AVR."

Hence, recent advances look for novel biomarkers or screening methods for unmasking even asymptomatic AS (Pibarot & Dumesnil, 2012). Beyond the question of the optimal time for intervention, the discovery of new biomarkers of the disease could lead to a specific development of an adequate pharmacological treatment. Therefore, the identification of groups of individuals sharing common characteristics that could be targeted by such a medication is an important challenge for aortic stenosis. Such groups are called *phenogroups*.

1.1.4 Implementing machine learning in cardiology

Finding and unmasking the factors that could predict the different facets of AS is an exhausting task which cannot be done manually. It is thus of interest for practitioners to automate the research for patterns that could be relevant in the context of AS.

The last decades witnessed the spectacular rise of *machine learning*, a field at the crossing of mathematics, statistics and computer science that could be briefly described as the art of uncovering patterns (Bishop, 2007, Chapter 1) and for which cardiology observes a growing, but careful, interest (Ben Ali et al., 2021 ; Sermesant, Delingette, Cochet, Jais, & Ayache, 2021), *e.g.* in mortality prediction of patients with known coronary artery disease (Pezel et al., 2022). A machine learning model can be described as a function f that processes a data set of *observations* x, sometimes called *samples*, and produces a *response* y. Briefly written:

$$\underbrace{y}_{\text{Response}} = \underbrace{f}_{\text{Model}}\Big(\underbrace{x}_{\text{Observation}} \Big). \tag{1.1}$$

A machine learning model is designed to achieve a specific *task*. This task varies according to observations, response, availability of *labels* to assess the quality of proposed responses, and the design of the model f. Observations can vary in nature: images, texts, customer information, social media interactions, or, of course, health records regarding aortic stenosis. The response can also vary in nature. It may, for example, be a price forecast, a weather forecast, or a category. Finally, the choice of the model affects the response produced. It can be either deterministic, *e.g.* summing two digits, or stochastic, *e.g.* answering with text an open question. In general, learning refers to the concept of optimising the model f until a specific criterion is met. This criterion is chosen according to the machine learning task.

Since the end of the twentieth century, drastic improvements in information technology led to the collection of increasing observed data, both in quantity and in size. Accompanied by the development of powerful machines in terms of memory and speed, this context served as a breeding ground for neural networks, a specific family of machine learning models. A neural network consists in a stack of multiple *layers*, where each layer's goal is to extract simplified concepts from the previous layer's concept. Thus:

> "(By) gathering knowledge from experience, this approach avoids the need for human operators to formally specify all the knowledge that the computer needs. The hierarchy of concepts enables the computer to learn complicated concepts by building them out of simpler ones" (Goodfellow, Bengio, & Courville, 2016, Introduction)

From this prism, *deep learning* refers to the specific case of a large number of such layers stacked within a neural network.

Machine and deep learning algorithms have been previously used for various cardiology tasks, including risk prediction and decision making optimisation (Ahmad et al., 2018 ; Feeny et al., 2019 ; Motwani et al., 2017 ; Tokodi et al., 2020). Recently, several studies reported the applicability and accuracy of machine and deep learning-based algorithms to detect AS in various settings (Chang et al., 2021 ; Cohen-Shelly et al., 2021 ; Hernandez-Suarez et al., 2019 ; Kwon et al., 2020 ; Wang et al., 2020).

1.1.5 The PROGRESSA study

Data observations are the fuel for machine learning. To leverage a model for AS, the Québec Heart and Lung Institute (Institut Universitaire de Pneumologie et Cardiologie de Québec, IUCPQ) and Laval University (Université Laval, UL) allowed us to exploit the *metabolic determinants of the progression of aortic stenosis* (PROGRESSA, NCT01679431) study directed by Philippe Pibarot, PhD, DMV. This study, started in 2005, was first introduced by Capoulade, Després, et al. (2012) with an initial number of 104 patients. Since then, this database unique in the world has been growing constantly (Capoulade et al., 2015 ; Lachmann et al., 2021 ; Tastet et al., 2017), so our work will focus on the version of PROGRESSA from the 30th of December 2020, now including 351 patients.

All patients presented in the dataset were affected by aortic stenosis with at least mild severity. Entries contain multiple visits of each patients at an average rate of a visit every 2 years. Patients were excluded if they had symptomatic AS, moderate or greater aortic regurgitation, or mitral valve disease (stenosis or regurgitation), left ventricular ejection fraction lower than 50%, and if they were pregnant or lactating.

The PROGRESSA study comprises multiple variables for each patient. These information can be divided essentially in *clinical* data, *e.g.* age, weight, blood pressure, *echocardiographic* data *e.g.* left ventricular mass, aortic jet velocity, *proteomic* data *i.e.* the level of expression of given genes in a patient, and imagery *e.g.* CT scans.

Thanks to the PROGRESSA study, we can explore biomarkers or risk factors associated to AS with machine learning. This problem can be solved using the specific task of clustering that we now introduce.

1.2 Clustering

1.2.1 Definition

Clustering is a fundamental learning task that involves separating data samples into several groups, each named cluster. A cluster should be a cohesive set of elements that may share some common properties. Ideally, these commonly shared properties should allow us to distinguish one cluster from the other (Bouveyron, Celeux, Murphy, & Raftery, 2019, Chapter 1). Therefore, this task is useful for exploring and uncovering knowledge in data analysis, *e.g.* biology with microarray analysis (McLachlan, Bean, & Peel, 2002 ; Sturn, Quackenbush, & Trajanoski, 2002), customer segmentation (Kansal, Bahuguna, Singh, & Choudhury, 2018 ; Kashwan & Velu, 2013), social network analysis (Bedi & Sharma, 2016 ; Himelboim, Smith, Rainie, Shneiderman, & Espina, 2017), political campaign analysis (Bode, Hanna, Yang, & Shah, 2015) or, in our context, phenogroup discovery of aortic stenosis (Kwak et al., 2020).

Given a potentially large collection of *data samples* $\boldsymbol{x}_i$ gathered into a dataset $\mathcal{D}$, a clustering algorithm is a deterministic or probabilistic model f that assigns each sample $\boldsymbol{x}_i$ to a cluster y_i. While $\boldsymbol{x}_i$ may take several values in a potentially large data space $\mathcal{X}$ *e.g.* images, tabular entries, or graphs, the cluster assignment is only an integer bounded by the desired maximal number of clusters K, formally:

$$\begin{aligned} f : \mathcal{X} &\mapsto [\![K]\!], \\ \boldsymbol{x}_i &\mapsto f(\boldsymbol{x}_i) = y_i. \end{aligned} \tag{1.2}$$

Conceptually, clustering belongs to the family of *unsupervised learning*. This means that the dataset does not contain any information about a potential ideal target. Therefore, we do not have explicit information guiding the optimisation of the model f.

In the absence of such targets, clustering hinges on two main questions (Hennig, 2015). The first main question concerns the assessment of correct clustering. We are interested there in knowing if the discovered clusters are insightful and teach us something about the data. However, we must emphasise that there is no global consensus on the definition of a cluster. Consequently, it was shown that each clustering algorithm cannot satisfy simultaneously multiple properties (Kleinberg, 2003). Overall, the use case drives the need and the "clustering is in part in the eye of the beholder" (Estivill-Castro, 2002). We may thus say that there are *no absolute best clustering* algorithm, yet some may be relatively better depending on the context. The second major question to address in clustering is the actual, sometimes called optimal, number of clusters *i.e.* the value of K in Eq. (1.2). For instance, a clustering algorithm could find more insightful clusters when searching only for 5 of them instead of 10. Perhaps more would be more beneficial. Yet again, the lack of formal definition of clusters implies that no method can be absolutely better than others in finding the correct number of clusters if that number is even existing. However, by restricting clusters to some narrow definition, methods for assessing the quality of the number of clusters exist (Biernacki, Celeux, & Govaert, 2000 ; Davies & Bouldin, 1979 ; Rousseeuw, 1987 ; Tibshirani, Walther, & Hastie, 2001).

Hence, clustering is a relevant task for finding phenogroups in the PROGRESSA dataset regarding AS as we are conducting an exploratory analysis. The number of phenogroups is for now unknown, yet our medical collaborators established from previous studies (Capoulade, Clavel, et al., 2012 ; Fatima et al., 2019 ; Gardezi et al., 2018) the hypothesis that the expression of aortic stenosis would comprise 5 main disease phenogroups according to the predominant pathobiological

process. These hypothetised groups are: lipidic, inflammatory, thrombotic (Sellers et al., 2019), fibrotic (Simard et al., 2017) and calcific (Lindman et al., 2016), *i.e.* at least phenogroups. The discovery and interpretation of such clusters would eventually open the path to personalised pharmacotherapy for treating AS.

1.2.2 Limitations for the PROGRESSA study

The structure of the PROGRESSA dataset is peculiar and will drive our choices in the design of a clustering algorithm in this thesis. The data set is made up of 3 distinct parts: a *clinicopathological* part which contains clinical, demographic, echocardiographic and metabolic data; a *proteomics* part, *i.e.* the level of expression of some selected proteins, and a *radiomics* part with the actual images and videos from echocardiography, Doppler, computed tomography (CT) scan records.

Each patient is associated with at least three entries in the clinicopathological dataset, one per medical visit. However, not all of them underwent proteomic analysis. Out of the 351 initial patients, only 141 have corresponding entries to proteomic expressions. Moreover, the proteomics analysis was always carried exactly 4 times from the first to the 4th visit. Finally, all patients are associated with a set of 1 or more images, even videos, resulting from various scans. However, these images are related only to each patient's first visit.

Beside images that are high-dimensional, the PROGRESSA dataset contains more than 500 variables both continuous or discrete, many of which containing missing values for only 351 patients. This poses a challenging high-dimensional clustering problem on mixed-type data. Moreover, the medical interpretation of clusters is challenging because of the large number of variables and several may be noisy and irrelevant to the phenogroups we seek. Consequently, a selection strategy must be considered to reduce the number of used variables per cluster and thus ease the interpretation.

The structure of the PROGRESSA dataset imposes a specific case of clustering: the *heterogeneous source* clustering. As all parts of the dataset cannot be naively concatenated due to temporal differences in visits, we need to design an approach that could leverage a new structure in the dataset on which to perform clustering. This need for a specific structure can be rephrased as finding a suitable representation of the dataset. Such representations can be learnt using neural networks.

That is why we will focus in this thesis on the elaboration of a clustering algorithm compatible with neural networks, such that it could handle data variety, from tabular to images and pave the way to heterogeneous clustering.

1.3 Thesis outline

This thesis aims at building a framework able to handle the structure of the PROGRESSA dataset for the purpose of clustering and hence phenogroup identification. To that end, we will revisit the general spectrum of discriminative clustering and propose a novel generic framework. In particular, this framework alleviates the training of any neural network for clustering. We list with every chapter here the associated contributions, whether published or in process.

Chapter 2

We start with a review of clustering algorithms with a specific focus on the deep discriminative clustering algorithms. We discuss the advantages and limitations of each model and incorporate a discussion of the algorithmic and probabilistic tools for interpreting the obtained clusters. Notably, we highlight how mutual information appeared as a common choice of objective function in recent years for training neural networks for discriminative clustering.

Chapter 3

We then discuss how the mutual information presents some core limitations and extend its definition to add geometrical constraints in the clustering. This extension is called *generalised mutual information* (GEMINI) and will serve as a basis for a complete clustering framework throughout the thesis. We introduce as well the package *GemClus*, a minimal-requirements Python package for repeating and exploiting most of the models of this thesis.

Published as a conference paper Louis Ohl, Pierre-Alexandre Mattei, Charles Bouveyron, Warith Harchaoui, Mickaël Leclercq, Arnaud Droit, and Frederic Precioso. Generalised mutual information for discriminative clustering. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 3377–3390. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/16294049ed8de15830ac0b569b97f74a-Paper-Conference.pdf

Preprint, submitted to a journal (extension) Louis Ohl, Pierre-Alexandre Mattei, Charles Bouveyron, Warith Harchaoui, Mickaël Leclercq, Arnaud Droit, and Frédéric Precioso. Generalised Mutual Information: a Framework for Discriminative Clustering. *arXiv preprint arXiv:2210.06300*, September 2023. URL https://hal.science/hal-04198445

Software, open-source Louis Ohl, Pierre-Alexandre Mattei, and Frédéric Precioso. GemClus. https://github.com/gemini-clustering/GemClus, March 2024

Chapter 4

After constructing the GEMINI framework, we highlight how the discriminative nature of neural networks and logistic regressions in clustering can easily incorporate regularisations leading to sparse models. This extension, called Sparse GEMINI, adds feature selection to the GEMINI framework, enhancing thus the interpretation of clusters.

Preprint, submitted to a journal Louis Ohl, Pierre-Alexandre Mattei, Charles Bouveyron, Mickaël Leclercq, Arnaud Droit, and Frédéric Precioso. Sparse gemini for joint discriminative clustering and feature selection. *arXiv preprint arXiv:2302.03391*, 2023.

Chapter 5

Beyond feature selection, we show how we can construct explainable clustering with decision trees where the selected variables obey structured rules and introduce two models: Kauri and Douglas. Specifically, Kauri is an end-to-end decision tree learnt using a kernel K-means objective. Douglas is a differentiable tree using combinations of soft bins of features trained with any GEMINI.

Preprint, submitted to a conference Louis Ohl, Pierre-Alexandre Mattei, Mickaël Leclercq, Arnaud Droit, and Frédéric Precioso. Kernel kmeans clustering splits for end-to-end unsupervised decision trees. *arXiv preprint arXiv:2402.12232*, 2024

Chapter 6

With the complete GEMINI framework built throughout the previous chapters, we develop a complete pipeline for preprocessing, clustering and selecting variables in the PROGRESSA dataset. Involving an application of GEMINI in consensus clustering, we develop a complete pipeline to obtain clusters of patients that are stable despite the progression of severity of AS and interpret them as phenogroups. We present as well briefly some related works by Sanabria et al. regarding the PROGRESSA study for which our main contribution was the decision process of the model construction and code revisions.

In submission Melissa Sanabria, Lionel Tastet, Simon Pelletier, Mickael Leclercq, Louis Ohl, Lara Hermann, Pierre-Alexandre Mattei, Frédéric Precioso, Nancy Coté, Philippe Pibarot, and Arnaud Droit. Deep learning-based algorithm to predict aortic stenosis progression from the progressa cohort, 2023.

Under writing Marie-Ange Fleury*, Louis Ohl*, Lionel Tastet, Mickaël Leclercq, Frédéric Precioso, Pierre-Alexandre Mattei, Jérémy Bernard, Mylène Shen, Nancy Côté, Arnaud Droit, and Philippe Pibarot. Enhancing risk stratification in aortic stenosis using echocardiography and artificial intelligence, 2024

Conclusion

Finally, we summarise all contributions of the thesis and offer insights on the theoretical proposal of heterogeneous source clustering with the GEMINI framework, which is part of our future works.

PART

Of clustering

CHAPTER 2

Clustering

3.1 Introduction . . . 33
3.2 Is MI a good clustering objective? . . . 34
3.2.1 Entropy perspectives . . . 34
3.2.2 A practical example . . . 35
3.3 Extending the MI to the GEMINI . . . 37
3.3.1 The discriminative clustering framework for GEMINIs . . . 37
3.3.2 The One-vs-All GEMINI . . . 38
3.3.2.1 Replacing the Kullback-Leibler divergence with other distances 38
3.3.2.2 f-divergence GEMINIs . . . 39
3.3.2.3 IPM and Wasserstein-GEMINIs . . . 40
3.3.3 The One-vs-One GEMINI . . . 42
3.4 GEMINI in practice . . . 44
3.4.1 Geometric considerations . . . 44
3.4.2 Estimating a GEMINI . . . 45
3.4.3 Complexity considerations . . . 45
3.4.4 Further speed-ups for the OvO Wasserstein . . . 46
3.4.5 The *GemClus* package . . . 47
3.5 Experiments . . . 48
3.5.1 When MI fails because of the modelling . . . 48
3.5.2 Resistance to outliers . . . 49
3.5.3 Leveraging a manifold geometry . . . 50
3.5.4 Fitting MNIST . . . 52
3.5.5 Number of non-empty clusters and architecture . . . 53
3.5.6 Number of clusters and training time . . . 54
3.5.7 Cifar10 clustering using a SIMCLR-derived kernel . . . 55
3.5.8 A practical application with Graph Neural Networks: the Enron email dataset . . . 56
3.6 Conclusion . . . 57

2.1 Intuition for the PROGRESSA dataset

The clustering of the PROGRESSA dataset requires the integration of multiple modalities: tabular data, images. Therefore, we are interested in developing an algorithm that could easily concatenate or merge representations of all modalities, whether at the data level or an intermediate

representation level. This operation is called *fusion* (Bramon et al., 2011). In practice, we will see that we did not perform fusion of radiomics in Chapter 6 and only fused proteomics with clinical data.

Throughout the many flavours of clustering algorithms, little did address the task of heterogenous data clustering. This problem arises when the data are composed of many different sources that vary in nature, for example conjunctions of graphs with images, or tabular data accompanied by sound. Therefore, models that were designed for one specific type of input space are not fit and we need to rethink models such that they can integrate the various input sources.

A trivial and simplistic solution would be to concatenate the input spaces, in the hope that traditional model can consider this concatenated input as a single block. However, doing so means merging various metric spaces that may not be comparable. Moreover, the challenge of adequately balancing even a linear combination of two metrics from two different metric spaces remains an open problem.

A deep-learning-orientated solution would rather be to transform each of the various sources into identical hence comparable metric spaces. This can be achieved with neural networks, one per source, and each finishing onto identical layers or layers of similar dimensions.

We explore in this chapter different approaches to do clustering of data. We specifically focus on the discriminative models because they allow an easy fusion of representations due to their absence of parametric hypothesis on the data distribution. We show how mutual information imposed itself as a strong objective in deep clustering and finally discuss some approaches to help interpreting the clusters with feature selection.

2.2 Modelling frameworks

Clustering is the task of grouping data samples. Each data sample is formally described as a random variable $\boldsymbol{x}$, taking values from $\mathcal{X}$. This variable is uni- or multidimensional, with a mix of continuous and/or discrete dimensions. We consider that we have a dataset of n independent and identically distributed samples $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{n}$. The membership of the cluster is indicated by y, a discrete random variable taking values in $[\![K]\!]$, where K is the number of clusters to determine.

To link these two random variables, a model is required. However, the nature of the assumptions that are made for the model greatly impacts the algorithmic procedures for learning. Here we describe two major contrasting frameworks: the *generative* and the *discriminative* modellings.

2.2.1 Starting from Bayes theorem

A clustering algorithm can be described as a probabilistic distribution p that assigns to a given sample $\boldsymbol{x}$ a cluster membership y. This distribution is controlled by a set of parameters θ. Thus, a clustering model is the parameterisation of the distribution p by θ and so it is written $p_\theta(y|\boldsymbol{x})$. According to the Bayes theorem, we can devise a definition of the clustering model $p_\theta(y|\boldsymbol{x})$:

$$p_\theta(y|\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x}|y)p_\theta(y)}{p_\theta(\boldsymbol{x})}. \tag{2.1}$$

This theorem highlights that building a distribution binding clusters y and data $\boldsymbol{x}$ necessarily implies the existence of three other distributions: one for the generation of the data given a cluster $p_\theta(\boldsymbol{x}|y)$, one for the proportion of clusters $p_\theta(y)$ and one for the probability of observing the data

(a) Generative modelling (b) Discriminative modelling

Figure 2.1 – The generative and discriminative modelling frameworks for clustering models. Observed variables are shaded.

$p_\theta(\boldsymbol{x})$, often called *likelihood*. We further note that as the clustering distribution is dependent on θ, most other distributions are consequently also dependent on θ, unless specified otherwise.

Given a clustering model $p_\theta(y|\boldsymbol{x})$, the final clustering of a dataset is the assignment of each sample in the dataset $\mathcal{D}$ to the cluster for which the conditional probability is maximal. The k-th cluster is defined as:

$$\mathcal{C}_k = \{\boldsymbol{x} \in \mathcal{D} | k = \arg\max_y p_\theta(y|\boldsymbol{x})\}. \tag{2.2}$$

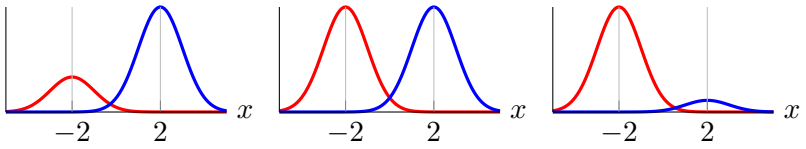
The definition offered by the Bayes theorem on the clustering models brings two different ways of defining the clustering model and its parameters. The first one is the generative modelling which views the clustering membership as a latent variable explaining how the data was generated, and the second one is the discriminative modelling which seeks immediately the clusters from the data. With probabilistic graphical models (Koller & Friedman, 2009, Chapter 1), we can summarise both views with Figure 2.1.

2.2.2 Generative models

In generative modelling, knowing the latent cluster y is sufficient to describe the distribution of its associated data $\boldsymbol{x}$, see Figure 2.1a. The design of this model is therefore focused on the right hand-side factors of the Bayes theorem in Eq. (2.1): a generative model is the design of $p_\theta(\boldsymbol{x}|y)$, with learnable proportions $p_\theta(y)$ (Bouveyron et al., 2019). Often, the distribution $p_\theta(\boldsymbol{x}|y)$ is very simple, *e.g.* Gaussian distributions. The generative approach can be interpreted as creating a ready-made template of how clusters would look like, then stretching the template until it fits as best as possible the observed data. The key idea of fitness is often measured with the likelihood:

$$p_\theta(\boldsymbol{x}) = \sum_{y=1}^{K} p_\theta(\boldsymbol{x}|y) p_\theta(y). \tag{2.3}$$

The intuition is that a model similar to the true process that generate the data should be likely to generate again similar samples to those observed. The clustering distribution $p_\theta(y|\boldsymbol{x})$ is then a consequence of the modelling. Indeed, a generative model indirectly specifies the clustering distribution because it is implicitly proportional to the generative process $p_\theta(\boldsymbol{x}|y)p_\theta(y)$. We give a simple example of a generative model where each cluster obeys a Gaussian distribution in Figure 2.2. Each cluster distribution is therefore written $p_\theta(\boldsymbol{x}|y) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_y, \sigma_y^2)$ and each cluster proportion $p_\theta(y) = \pi_y$. The parameters θ comprise the location $\boldsymbol{\mu}_y$, the scale σ_y^2 and proportion π_y for each cluster. In the one-dimensional binary example from Figure 2.2, we have a total of 6 parameters with $\theta = \{\boldsymbol{\mu}_{\text{red}}, \sigma^2_{\text{red}}, \pi_{\text{red}}, \boldsymbol{\mu}_{\text{blue}}, \sigma^2_{\text{blue}}, \pi_{\text{blue}}\}$. The resulting likelihood is defined as:



(a) 25% of red cluster (b) 50% of red cluster (c) 90% of red cluster

Figure 2.2 – An example of generative models with 2 clusters in 1 dimension. The parameters θ of the model comprise the Gaussian distribution locations $\boldsymbol{\mu}_{\text{red}} = -1$, $\boldsymbol{\mu}_{\text{blue}} = 2$, the scales $\sigma^2_{\text{red}} = \sigma^2_{\text{blue}} = 1$ and the proportions of each clusters.

$$p_\theta(\boldsymbol{x}) = p_\theta(y = \text{red})p_\theta(\boldsymbol{x}|y = \text{red}) + p_\theta(y = \text{blue})p_\theta(\boldsymbol{x}|y = \text{blue}) \tag{2.4}$$

$$= \pi_{\text{red}}\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{\text{red}}, \sigma^2_{\text{red}}) + \pi_{\text{blue}}\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{\text{blue}}, \sigma^2_{\text{blue}}). \tag{2.5}$$

For a mixture of 2 d-dimensional Gaussian distribution with proportions π_1, π_2, locations $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ and the same covariance $\boldsymbol{\Sigma}$, it is possible to show that the clustering distribution is defined (Bishop, 2007, Eq. 4.64):

$$p_\theta(y = 1|\boldsymbol{x}) = \text{Sigmoid}(\boldsymbol{w}^\top \boldsymbol{x} + b), \tag{2.6}$$

where the sigmoid function is defined for all real values a:

$$\text{Sigmoid}(a) = \frac{1}{1 - e^{-a}}, \tag{2.7}$$

and the coefficients are:

$$\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2), \tag{2.8}$$

and:

$$b = -\frac{1}{2}\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log\frac{\pi_1}{\pi_2}. \tag{2.9}$$

We observe in Eq. (2.6) that the clustering distribution $p_\theta(y|\boldsymbol{x})$ is therefore drawn around a linear decision boundary of coefficients $\boldsymbol{w}$ and b when the covariances are equal between 2 Gaussian distributions.

To optimise the parameters, the most straightforward method is *maximum likelihood*, *i.e.* maximising the value of the likelihood $p_\theta(\boldsymbol{x})$ of all samples $\boldsymbol{x}_i$ in the dataset $\mathcal{D}$. Under the common assumption of i.i.d. samples in the dataset, the maximum likelihood parameter is defined as:

$$\hat{\theta}_{\text{MLE}} = \arg\max_\theta \prod_{\boldsymbol{x}\in\mathcal{D}} p_\theta(\boldsymbol{x}). \tag{2.10}$$

However, this likelihood is often challenging to maximise due to the unobserved and latent nature of the cluster membership y. In such a case, an expectation-maximisation (EM) algorithm can be used to find a local maximum for the likelihood (McLachlan & Krishnan, 2007). This algorithm alternates between two steps. In the first step, called expectation, the probability of cluster

membership is estimated using the current state of the model parameters θ. In the second step, called maximisation, the parameters θ are optimised to maximise the likelihood given the current cluster membership probabilities. This procedure is repeated multiple times until convergence. Still today, this algorithm remains one of the most commonly used in mixture modelling.

The most common example of a generative clustering model is the Gaussian Mixture Model (GMM), where each cluster is assumed to follow a Gaussian distribution (Banfield & Raftery, 1993 ; Bouveyron et al., 2019). Other examples comprise mixtures Gaussian copulae for handling mixed types of variables (Marbac, Biernacki, & Vandewalle, 2017), mixtures of multivariate t-distribution (Peel & McLachlan, 2000) for robustness against outliers thanks to heavily tailed distributions, for instance, applied on Box-Cox transformed data (Lo, Brinkman, & Gottardo, 2008), mixtures of Poisson distributions (Karlis & Tsiamyrtzis, 2008), or mixtures of multinomial distributions (Goodman, 1974).

Model-based clustering benefits from a large panel of statistical tools. Notably, it is possible to assess the adequation of the model to the data and validate a correct choice of number of clusters with internal scores. For example, the Bayesian information criterion (BIC, Schwarz, 1978) is defined for parameters $\hat{\theta}_{\text{MLE}}$ obtained after training:

$$\text{BIC}(K) = 2 \sum_{\boldsymbol{x} \in \mathcal{D}} \log p_{\hat{\theta}_{\text{MLE}}}(\boldsymbol{x}) - \nu_K \log n, \tag{2.11}$$

where ν_K is the number of free parameters in $\hat{\theta}_{\text{MLE}}$ for K clusters. For instance: in the previous GMM example, we had $\nu_K = 5$ because the proportions of the clusters π_{red} and π_{blue} depend on each other. BIC encourages models to fit well the data with a strong likelihood while maintaining a low number of parameters. This criterion focuses on the good number of components that fit well the data in terms of likelihood. However, this does not imply that it is correct for clustering the data. For example, a cluster for which the samples describe a spline cannot be covered by a single Gaussian distribution. In this case, we would like to fit multiple Gaussian distributions at different part of the spline and consider all of them as describing a single cluster. A more adequate internal score for mixture-based clustering models is the integrated complete likelihood (ICL, Biernacki et al., 2000) for K clusters and M components:

$$\text{ICL}(K, M) = \text{BIC}(M) - \sum_{\boldsymbol{x} \in \mathcal{D}} \sum_{y=1}^{K} p_{\hat{\theta}_{\text{MLE}}}(y|\boldsymbol{x}) \log p_{\hat{\theta}_{\text{MLE}}}(y|\boldsymbol{x}). \tag{2.12}$$

The ICL penalises the BIC by subtracting the entropy of the cluster memberships of the model. As the entropy becomes smaller when the model decisions are clear-cut, ICL encourages models for which the number of clusters leads to distinct separations.

The difficulty of generative models for clustering lies essentially in the choice of the generative distribution $p_\theta(\boldsymbol{x}|y)$. In practice, the choice of distribution, *e.g.* Poisson, Gaussian, is often guided by the expertise developed on the dataset. However, the statistical challenges emerging from the high-dimensional nature of modern data call for adequate regularisations and constraints of the distributions (Bouveyron & Brunet-Saumard, 2014b, Chapter 8).

Beyond the usage of well-known distributions with scalar or matrix parameters, the last decade witnessed the rise of generative modelling with neural networks, notably with the variational auto-encoders (VAE, Kingma & Ba, 2014 ; Rezende, Mohamed, & Wierstra, 2014) and the generative adversarial networks (GAN, Goodfellow et al., 2014). Both methods aim to build generative models in which the latent variable $\boldsymbol{z}$ is *continuous* contrary to generative clustering models. Starting

from ease-to-sample distributions, these models transform low-dimensional latent variables to high-dimensional variables using complex nonlinear transformations. VAEs, as part of the broader class of Deep Latent Variable Models (DLVM), parameterise the generative distribution with the output of a function g from the latent variable: $p(x|g(\boldsymbol{z}))$. For instance, g can be a neural network returning the mean and diagonal covariance of a Gaussian distribution. To optimise the generative process, an amortised variational inference method is often used with an encoder network as the proposal distribution for approximating the posterior distribution of the latent variable, hence the name variational auto-encoder. GANs focus instead on optimising a zero-sum game, where the generative distribution must produce samples of sufficient quality to fool a discriminative network, sometimes called a critic. The goal of this discriminator is to differentiate true samples coming from a target distribution, the data, from samples created *de novo* by the generator. The core limitation of these initial models in the context of clustering is the usage of a continuous latent variable, the code, rather than a discrete variable, the cluster. Proposals exist to make the latent variable categorical (Jang, Gu, & Poole, 2017), but this does not imply that they were intended for clustering. Thus, extensions were proposed to adapt VAEs and GANs for clustering.

In the VAE framework, the first straightforward approach was to propose a Gaussian mixture model as prior distribution * (Dilokthanakul et al., 2016 ; Jiang, Zheng, Tan, Tang, & Zhou, 2017). Thus, the choice of a specific cluster restricts the sampling in the latent space that generates the corresponding set of clustered data. To provide further improvement, X. Li, Chen, Poon, et Zhang (2019) proposed to use latent tree models as a distribution over the latent space to leverage structure learning jointly with clustering. There is today still ongoing work on the optimisation of the prior for clustering (L. Yang, Fan, & Bouguila, 2021) or the addition of constraints in VAE (H. Ma, 2022).

Turning to GAN, a simple clustering approach is to extend the discriminator's task to assign each sample to one class instead of guessing fake from generated data (Springenberg, 2015). Another approach proposed by Mukherjee, Asnani, Lin, et Kannan (2019) with their clusterGAN model is to constrain the sampling space to a mixture of continuous and discrete variables that depend on the cluster. Recently, GAN mixtures have also been proposed (Mello, Assunção, & Murai, 2022). To still take into account and correct the errors that could arise from poorly initialised clusters, other GAN-based approaches tend to regularise their model using auto-encoders (Mrabah, Bouguessa, & Ksantini, 2020 ; Zhou, Hou, & Feng, 2018).

In general, generative modelling presents compelling tools for clustering. However, the difficulty of making assumptions is one of the main keys to the success of the model. Moreover, in the context of mixed types of variables, this choice of parametric assumption becomes even harder. Often, assumptions of independence between categorical and continuous variables are made to achieve mixed-type variables clustering with mixture models (Marbac et al., 2017 ; Marbac & Sedki, 2017). For VAEs, this can be done by training individualised VAE per variable (C. Ma, Tschiatschek, Turner, Hernández-Lobato, & Zhang, 2020). However, these solutions remain at the data level and are hardly applicable for the concatenation of specified representations on heterogeneous sources of data. Although this fusion problem of intermediate representations of the data could be leveraged by generative neural networks such as GANs or VAE, we believe that this choice remains suboptimal. On the one hand, the difficulty of training GANs (Arjovsky & Bottou, 2017 ; Bottou, Curtis, & Nocedal, 2018 ; Salimans et al., 2016) limits their practical application, and on the other hand, the

∗. We use here the term prior in the sense of parametric assumption on the latent variable, instead of some belief on the model's parameters.

latent space of a VAE which maps samples to distributions according to the variational distribution hardly account for modality fusion.

2.2.3 Discriminative models

To avoid the burden of choosing parametric assumptions following the generative modelling, we turn to the discriminative point of view. In this context, we do not take any assumption at all regarding the data distribution and denote it $p_{\text{data}}(\boldsymbol{x})$. However, we assume to be able to sample from a dataset of fixed samples. The discriminative framework can be interpreted as taking the data as is and inferring clusters instead of finding assignments suiting a notion of likelihood. Thus, we only design a discriminative model $p_\theta(y|\boldsymbol{x})$ and we obtain:

$$p_\theta(\boldsymbol{x}, y) = p_{\text{data}}(\boldsymbol{x}) p_\theta(y|\boldsymbol{x}). \tag{2.13}$$

From a generative perspective, the discriminative modelling corresponds to the joint distribution of a data distribution and inference distribution with decoupled parameters (Minka, 2005). The data is generated by a set of external parameters θ' and the complete model is written $p_{\theta,\theta'}(\boldsymbol{x}, y) = p_\theta(y|\boldsymbol{x}) p_{\theta'}(\boldsymbol{x})$. Note however that in contrast to this generative view, we do not even assume the existence of such parameters in our discriminative models when writing $p_{\text{data}}(\boldsymbol{x})$.

This framework can be conveniently used with any function ψ_θ whose outputs lie in the simplex of dimension K: the number of clusters. This output can then be considered as the parameters of a categorical distribution defining the conditional cluster membership:

$$y|\boldsymbol{x} \sim \text{Categorical}(\psi_\theta(\boldsymbol{x})). \tag{2.14}$$

Thanks to the degrees of freedom for the definition of ψ_θ, the discriminative clustering framework can tolerate various softmax-ended neural networks. For example, we can consider logistic regressions where the discriminative model takes the form:

$$\psi_\theta(\boldsymbol{x}) = \text{Softmax}(\boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b}), \tag{2.15}$$

where the parameters are $\theta = \{\boldsymbol{W}, \boldsymbol{b}\}$. The softmax function is defined for any real vector and returns a stochastic vector:

$$\text{Softmax}(\boldsymbol{z}) = \left[\frac{e^{\boldsymbol{z}_1}}{Z}, \ldots, \frac{e^{\boldsymbol{z}_i}}{Z}, \ldots, \frac{e^{\boldsymbol{z}_d}}{Z}\right], \quad \text{with} \quad Z = \sum_{i=1}^{d} e^{\boldsymbol{z}_i}. \tag{2.16}$$

Therefore, in contrast to generative modelling, discriminative modelling encompasses our hypotheses through the design of the decison boundary. For example, Eq. (2.16) shows that a logistic regression is the discriminative equivalent of a mixture of Gaussian distributions with equal variances because the decision boundaries of the latter are also linear as we showed in Eq. (2.6).

Beyond the scope of clustering, discriminative modelling is the core approach used for classification with neural networks. Thus, a neural network that was well designed for a classification task on a specific type of data can immediately be used for clustering. This motivates us to choose discriminative modelling for solving the clustering task in the case of PROGRESSA.

However, the discriminative perspective in the context of clustering severely affects the procedures for learning optimal parameters.

2.3 The challenge of learning in discriminative clustering

We now discuss how discriminative models can be trained and the implied properties.

2.3.1 The shortcomings of classical statistical tools

Due to the absence of a model on the data distribution $p_{\text{data}}(\boldsymbol{x})$, we cannot use maximum likelihood to learn the optimal parameters, and consequently neither expectation-maximisation nor variational inference. Additionally, we cannot construct a generative cluster distribution $p_\theta(\boldsymbol{x}|y)$ since it will be constrained by the empirical distribution $p_{\text{data}}(\boldsymbol{x})$, which is undefined for samples outside of the dataset. In other words, a discriminative model cannot generate samples outside the dataset. The only elements we can estimate are the proportions of the clusters through marginalisation:

$$p_\theta(y) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[p_\theta(y|\boldsymbol{x})\right]. \tag{2.17}$$

In summary, the absence of parametric assumptions on the data distribution leads to the absence of joint modelling:

$$\underbrace{p_\theta(y|\boldsymbol{x})}_{\text{Known}} \times \underbrace{p_{\text{data}}(\boldsymbol{x})}_{\textit{Unknown}} = \underbrace{p_\theta(\boldsymbol{x}|y)}_{\textit{Unknown}} \times \underbrace{p_\theta(y)}_{\text{Estimable}} = \underbrace{p_\theta(\boldsymbol{x}, y)}_{\textit{Unknown}}. \tag{2.18}$$

The only thing we assume to be able is sampling from the data distribution, owing to the presence of a dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$.

Therefore, a different approach must be taken for training discriminative models in clustering. To build such an approach, we will draw inspiration from the supervised models in classification that are often discriminative. These models leverage the learning with an objective function that is decoupled from the model.

2.3.2 Objective functions in classification

In the absence of parametric assumptions on the data, we are interested only in grouping the samples based on some meaningful criterion. In classification tasks, in contrast to clustering, we have access to labels that provide us with this guiding criterion. Therefore, we know that there exists an ideal distribution $p_{\text{data}}(y|\boldsymbol{x})$ that we must match as best as possible with our model $p_\theta(y|\boldsymbol{x})$. To that end, we must measure the distance of our current model from this ideal distribution and make it as close as possible. The most common distance between two distributions is the Kullback-Leibler (KL) divergence. For two arbitrary distribution q_1 and q_2, the KL divergence is defined:

$$D_{\text{KL}}\left(q_1(\boldsymbol{z}))\|q_2(\boldsymbol{z})\right) = \mathbb{E}_{\boldsymbol{z}\sim q_1(\boldsymbol{z})}\left[\log \frac{q_1(\boldsymbol{z})}{q_2(\boldsymbol{z})}\right]. \tag{2.19}$$

Note that the KL is not a distance due to its non-symmetry. Incorporating in this divergence the definition of our target model $p_{\text{data}}(y|\boldsymbol{x})$ and our classification model $p_\theta(y|\boldsymbol{x})$ would only tell us how far apart these two distributions are for one specific datum $\boldsymbol{x}$. Therefore, we wrap up this divergence in an expectation over the data distribution to ensure that *on average*, our model sticks as best as possible to the targets. This data distribution is part of the ideal model $p_{\text{data}}(\boldsymbol{x})$. Thus, we get the average distance of our parameters to the data distribution $\mathcal{L}(\theta)$:

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[D_{\text{KL}}\left(p_{\text{data}}(y|\boldsymbol{x})\|p_\theta(y|\boldsymbol{x})\right)\right], \tag{2.20}$$

$$= \mathbb{E}_{\boldsymbol{x},\sim p_{\text{data}}(\boldsymbol{x}),y\sim p_{\text{data}}(y|\boldsymbol{x})}\left[\log\frac{p_{\text{data}}(y|\boldsymbol{x})}{p_\theta(y|\boldsymbol{x})}\right], \tag{2.21}$$

$$= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}),y\sim p_{\text{data}}(y|\boldsymbol{x})}\left[\log p_{\text{data}}(y|\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}),y\sim p_{\text{data}}(y|\boldsymbol{x})}\left[\log p_\theta(y|\boldsymbol{x})\right], \tag{2.22}$$

$$= \text{cst} - \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\sum_{k=1}^{K} p_{\text{data}}(y=k|\boldsymbol{x})\log p_\theta(y=k|\boldsymbol{x})\right]. \tag{2.23}$$

The constant here depends only on the ideal distribution and thus is not impacted by our choice of discriminative parameters θ. Under the assumption that we have access to a dataset containing n samples independently and identically distributed from $p_{\text{data}}(\boldsymbol{x}, y)$, we can estimate the expectation using Monte Carlo. Noting the dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ which specifies the class to which each sample should belong, the target distribution can be approximated using a delta Dirac distribution for all $\boldsymbol{x}_i$: $p_{\text{data}}(y|\boldsymbol{x} = x_i) \approx \mathbb{1}[y = y_i]$. As we ignore the constant first term from Eq. (2.23), we get:

$$\mathcal{L}(\theta) = -\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\sum_{k=1}^{K} p_\theta(y=k|\boldsymbol{x})\log p_\theta(y=k|\boldsymbol{x})\right], \tag{2.24}$$

$$\approx -\frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{x}_i,y_i\in\mathcal{D}}\sum_{k=1}^{K}\mathbb{1}[y_i=k]\log p_\theta(y=k|\boldsymbol{x}=\boldsymbol{x}_i), \tag{2.25}$$

$$= -\frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{x}_i,y_i\in\mathcal{D}}\log p_\theta(y=y_i|\boldsymbol{x}=\boldsymbol{x}_i). \tag{2.26}$$

Notice that we replaced the ideal target distribution $p_{\text{data}}(y|x)$ with the simple indicator function, which is equal to 1 only on the observed class. This objective is called *cross-entropy* and is one of the most natural *objective function*, sometimes called *cost*, in classification tasks and enjoys good properties: it is convex w.r.t. to the outputs of the model and ensures the minimisation of the KL divergence to the empirical estimate of p_{data}. Moreover, the minimisation of the cross-entropy is equivalent to maximising the likelihood of the data distribution in classification.

Owing to its differentiability and its sole dependence on the output of the model, this objective can be trained by gradient descent on condition that the model's outputs are differentiable w.r.t. θ. This means differentiating the cross-entropy and backpropagating the derivative throughout the complete model parameters such that the cross-entropy gets lowered. When the derivative of the objective is 0, the model found a local minimum.

Throughout this example, we see that the notion of objective function depending on the output of the model providing a convenient gradient is a functional solution for training discriminative models. The cross-entropy naturally emerged from the distance comparison between our model and a target model, guiding thus the gradient descent. We are interested in finding such an objective function that would fit into the context of clustering for discriminative models.

2.3.3 Mutual information as a promising objective

For the classification task, we relied on the distance in the KL sense to a target distribution to train the model to develop the cross-entropy loss. For the clustering task with generative modelling,

we rely on the likelihood of the data describing how fit our model is to the data. Therefore, we must focus on a key property that is desirable in the discriminative clustering case. The key idea that the final clustering should reflect insights on the data distribution is that specific property. Indeed, clusters inform us about the data and conversely, knowing the data informs us about the cluster. This notion is conveyed through the *dependence* between two random variables. In our specific case: the clusters y and the data $\boldsymbol{x}$ must be as dependent as possible.

To seek dependence between two random variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ there exists a score that we can use as an objective function based as well on the KL divergence: the *mutual information* (MI), defined as:

$$\mathcal{I}(\boldsymbol{\alpha};\boldsymbol{\beta}) = D_{\mathrm{KL}}\left(p(\boldsymbol{\alpha},\boldsymbol{\beta}) \| p(\boldsymbol{\alpha})p(\boldsymbol{\beta})\right). \tag{2.27}$$

The mutual information can be seen as a measure of how dependent two random variables are: the greater, the more dependent. Indeed, following from Theorem 2.3.1, a null mutual information value indicates that the clusters y and the data $\boldsymbol{x}$ are independent, *i.e.* unrelated.

Theorem 2.3.1 (Independence in mutual information)**.** *Let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be two random variables. Both variables are independent if and only if their mutual information is equal to 0.*

Remark 2.3.1. *Theorem 2.3.1 is a direct consequence of $D_{KL}(p\|q) = 0 \iff p = q$.*

The upper bound of mutual information is the minimum of the entropies of y and $\boldsymbol{x}$. Thus, maximising mutual information to increase dependence between data and clusters seems a coherent objective. However, we will show in Chapter 3 that this maximum can be attained with solutions that do not provide satisfactory clusters.

The definition from Eq. (2.27) cannot be optimised as such because the joint model $p_\theta(\boldsymbol{x}, y)$ and the data distribution $p_{\text{data}}(\boldsymbol{x})$ are unknown in the discriminative context. Fortunately, well-known properties of MI can invert the distributions on which the KL divergence is computed (Bridle, Heading, & MacKay, 1992 ; Krause, Perona, & Gomes, 2010) via Bayes' theorem:

$$\begin{aligned}
\mathcal{I}(\boldsymbol{x}; y) &= D_{\mathrm{KL}}\left(p_\theta(\boldsymbol{x}, y) \| p_{\text{data}}(\boldsymbol{x})p_\theta(y)\right), && (2.28)\\
&= \mathbb{E}_{\boldsymbol{x},y\sim p_\theta(x,y)}\left[\log \frac{p_\theta(\boldsymbol{x}, y)}{p_{\text{data}}(\boldsymbol{x})p_\theta(y)}\right], && (2.29)\\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(x)}\left[\mathbb{E}_{y\sim p_\theta(y|\boldsymbol{x})}\left[\log \frac{p_\theta(y|\boldsymbol{x})p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})p_\theta(y)}\right]\right]. && (2.30)
\end{aligned}$$

Then, we can simplify the factors on the data distribution inside the log and reidentify the KL divergence between a conditional distribution and a single marginal within an expectation:

$$\mathcal{I}(\boldsymbol{x}; y) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[D_{\mathrm{KL}}\left(p_\theta(y|\boldsymbol{x}) \| p_\theta(y)\right)\right]. \tag{2.31}$$

Owing to the expectation of the data distribution, we can derive estimates of mutual information and its gradients w.r.t. θ using Monte Carlo. Thus, the usage of the product rule within the KL divergence to get an estimable version is the key property to compute mutual information in discriminative modelling. This estimate depends only on the output of the model $p_\theta(y|\boldsymbol{x})$ from which we can estimate the proportions of the clusters through marginalisation. Note that the cost function for clustering in Eq. (2.31) is interestingly similar to the KL we minimised in supervised

learning from Eq. (2.20). In both equations, we sample on a data distribution and optimise the KL divergence between the model's clustering outputs and a target. In Eq. (2.31), this target to reach is the ideal classifier on the dataset and we minimise the KL. In Eq. (2.20), the sampling distribution is the empirical approximation of the data distribution, the target is the cluster proportions and we maximise the KL. Finally, we can maximise mutual information through gradient *ascent*.

However, we must remember that we do not maximise the true mutual information. Indeed, we previously showed with Eq. (2.17) that the proportions $p_\theta(y)$ are estimated with Monte Carlo. This means that the true proportions of the clusters are often not known in practice because batch optimisation leads to estimates of this expectation. Consequently, by rewriting this estimate as a proposal distribution $q(y)$ for the proportions of the cluster, we can unfold mutual information as follows (Poole, Ozair, Van Den Oord, Alemi, & Tucker, 2019):

$$\mathcal{I}(\boldsymbol{x}; y) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[D_{\text{KL}}\left(p_\theta(y|\boldsymbol{x})\|p_\theta(y) \times \frac{q(y)}{q(y)}\right)\right], \tag{2.32}$$

$$= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[D_{\text{KL}}\left(p_\theta(y|\boldsymbol{x})\|q(y)\right) - D_{\text{KL}}\left(p_\theta(y)\|q(y)\right)\right], \tag{2.33}$$

$$\leq \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[D_{\text{KL}}\left(p_\theta(y|\boldsymbol{x})\|q(y)\right)\right]. \tag{2.34}$$

This implies that maximising mutual information through mini-batches is rather about an upper bound estimate of mutual information, even though the KL divergence between the estimate cluster proportions and the true proportions may be negligible due to our unbiased estimate. This upper bound also holds if the target variable is continuous in $\mathcal{I}(\boldsymbol{x}; \boldsymbol{z})$.

2.4 Other clustering models

Between generative and discriminative models, there exist other clustering models. We discuss some of these objectives and compare their (dis)advantages with mutual information.

2.4.1 K-means

K-means is the most standard clustering algorithm (Likas, Vlassis, & Verbeek, 2003). The algorithm consists of two alternating steps. Starting from a set of K centroids: $\{\boldsymbol{\mu}_k\}_{k=1}^K$, all samples in the dataset are affected to the cluster $\mathcal{C}_k$ matching their closest centroid $\boldsymbol{\mu}_k$. Then each centroid is recomputed as the mean of all samples assigned to the corresponding cluster. Overall, the K-means algorithm minimises the following objective (Elkan, 2003 ; Lloyd, 1982), although there are no guarantees of reaching the global optimum:

$$\boldsymbol{\mu}^\star_{1\dots K} = \underset{\boldsymbol{\mu}_{1\dots K}}{\arg\min} \sum_{k=1}^{K} \sum_{\boldsymbol{x}\in\mathcal{C}_k} \|\boldsymbol{x} - \boldsymbol{\mu}_k\|_2^2. \tag{2.35}$$

The simplicity of the algorithm makes it a standard basis for clustering algorithms. However, its decision boundaries are linear because they appear for all samples laying equidistantly of two close means. Consequently, this does not allow complex cluster shapes when the Euclidean distance is used between data samples. An alternative called kernel K-means (Dhillon, Guan, & Kulis, 2004) can be used to alleviate more complex boundaries. Instead of drawing a boundary in the Euclidean space, it is drawn in a reproducing kernel Hilbert space $\mathcal{H}$ endowed with projection φ (Hofmann, Schölkopf, & Smola, 2008). The new objective is described:

$$\mathcal{L}(\{\boldsymbol{\mu}_k\}_{k=1}^K) = \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in \mathcal{C}_k} \|\varphi(\boldsymbol{x}) - \boldsymbol{\mu}_k\|_{\mathcal{H}}^2. \tag{2.36}$$

Beyond its wide applicability, K-means offers multiple advantages. First, the definition of fixed centroids also allows the clustering of unseen samples, *i.e.* samples that were not part of the training dataset. Second of all, this algorithm is fast as it scales linearly to the number of samples n and the number of input features d. However, K-means has several drawbacks: it is very sensitive to outliers, *i.e.* samples far away from the main modes of the data, and it displays high sensitivity to the initialisation. Moreover, the clustering model conveyed by K-means is a mixture of Dirac models which does not provide a fine-grained quantification of the cluster conditional probability.

K-means can be seen as a generative clustering model as it is similar to a mixture of isotropic Gaussian distributions. It can also be seen and used as an objective function for discriminative clustering as we will show in Chapter 5.

2.4.2 Spectral clustering

Related to kernel K-means, spectral clustering is a multi-step algorithm (von Luxburg, 2007) in which an alternative representation of the data is derived and used as input for a K-means algorithm. At the start of the algorithm, an affinity matrix is computed between all samples. The choice of the notion of affinity is arbitrary: ϵ-neighbourhood, n nearest neighbours, or a kernel like RBF, polynomial. This affinity matrix allows us to define a Laplacian matrix that views the complete dataset as a graph. The nature of the Laplacian can vary, *e.g.* being normalised or symmetrised (Ng, Jordan, & Weiss, 2001 ; Shi & Malik, 2000). Then, an alternative representation of the data emerges from the spectrum of this Laplacian matrix. Specifically, the first K eigenvectors of the Laplacian matrix are used as new features. Consequently, each sample is associated with a vector of dimension K where its k-th component is its weight for the k-th eigenvector.

While spectral clustering is efficient when cluster structures are non-convex, its strong requirement is the construction of a graph between all samples which enforces the fusion of modalities when several modalities are present. Moreover, its multi-step nature makes it inapplicable to unseen samples. Once again, due to the final K-means algorithm, the spectral clustering can be seen as a delta Dirac distribution which only delivers hard membership to clusters for all samples.

2.4.3 Hierarchical clustering

Hierachical clustering is a method that iteratively gathers or separates samples to build clusters in an ordered manner. It is commonly used in bioinformatics (Bohbot et al., 2022 ; Lachmann et al., 2021) for the structured point of view it offers on the data. Through the construction of a dendrogram linking at low or high levels the samples, the introduced hierarchy offers insight into the proximity of samples. Two types of hierarchical clustering algorithms can be distinguished: the agglomerative ones which start with all samples in their own cluster to finish with a single cluster, *i.e.* bottom-up, and the divisive ones that run in the opposite way, *i.e.* top-down. It is important to note here that, unlike decision trees, hierarchical algorithms do not provide rules explaining the branches of the dendrogram.

The bottom-up approach greedily merges the clusters that are closest until all samples are in a single cluster. The naive complexity of such algorithm is $\mathcal{O}(n^3)$, but some improved proposals brought it down to $\mathcal{O}(n^2)$(Defays, 1977 ; Sibson, 1973). To obtain K clusters, it is necessary to

perform $n - K$ iterations of the algorithm. To perform well, this algorithm needs to define the notion of distance between clusters, *i.e.* sets of samples, called *linkage*. The most common forms of linkage are *single* (Sibson, 1973), *complete* (Defays, 1977), *centroids* or the Ward linkage (Ward Jr, 1963). Respectively, they correspond to the distance between the closest points of each cluster, the distance between the farthest points of each cluster, the distance between the centroids of each cluster, and the variation of the sum of squares through merging.

Divisive algorithms are built top-down (Roux, 2018) and try to find the best split for separating a cluster into two new ones. In such case, the proposals of split become combinatorial and reach $2^{n-1} - 1$ proposals for n samples (Edwards & Cavalli-Sforza, 1965). These models are therefore costly, yet account for few iterations, as we only need $K - 1$ splits to obtain K clusters. Multiple splitting procedures were proposed to alleviate the computation of all splits (Hubert, 1973 ; Williams & Lambert, 1959) including K-means (Mollineda & Vidal, 2000). One of the most well known divisive clustering algorithms is DIANA (Kaufman & Rousseeuw, 1990). Roux (2018) suggests that ratio-based splits are among the most efficient. To further accelerate the evaluation of the gain procedures and constrain the search space, combinations of divisive clustering algorithms with model-based clustering algorithms were proposed, *i.e.* generative models, hence using the maximum likelihood as a global objective for the model (Burghardt, Sewell, & Cavanaugh, 2022 ; Sharma, López, & Tsunoda, 2017). We can still note the usage of K-means as a heuristic for proposing splits at each node (Sharma et al., 2017).

In discriminative clustering, hierarchical clustering only offers hard cluster memberships for all samples. It is possible however to obtain soft assignments using generative modelling. For example, Baudry, Raftery, Celeux, Lo, et Gottardo (2010) use entropy gains to merge components of a mixture model into clusters. Moreover, similar to spectral clustering, it cannot be applied to unseen samples and is hardly compatible with modality fusion.

2.5 Thrive of mutual information: from clustering to representation learning

Mutual information is an elegant objective function, which can train through gradient ascent various differentiable models. We trace here the multiple usages it has met for the past 30 years in the context of clustering.

2.5.1 Early usage of mutual information for clustering

The MI was first used as an objective for learning discriminative clustering models by Bridle et al. (1992). They described MI as an objective that maximises the *fairness* of a model, *i.e.* the entropy of the cluster proportions, and aim to maximise *firmness*, *i.e.* minimising the conditional entropy. Thus, a good clustering model is fair but firm.

Two decades later, Krause et al. (2010) initiated again the work on discriminative clustering models with mutual information. Similarly to Bridle et al. (1992), they first propose to train a logistic regression but specifically add a constraint ℓ_2 to alleviate the constraint on the firmness of the classifier model. Among the multiple proposals of this framework called regularised mutual information (RIM), Krause et al. (2010) also propose to learn a regularised logistic model on a positive semi-definite kernel matrix of some dataset instead of the sample features to alleviate a non-linear decision boundary.

Another interesting approach, named the information bottleneck, focusses on the notion of information relevance. Indeed:

> "The problem of extracting a relevant summary of data, a compressed description that captures only the relevant or meaningful information, is not well-posed without a suitable definition of relevance" (Tishby, Pereira, & Bialek, 2000)

The information bottleneck framework is derived from signal processing theory and related to rate-distortion theory. As such, it considers an input variable Z, that must be quantised into a lower-dimensional or discrete variable called code $\tilde{Z}$. However, this quantisation must be done such that another output variable $\bar{Z}$, can be recovered from the code. The final model is therefore:

$$p^{\star}(\tilde{Z}|Z) = \underset{p(\tilde{Z}|Z)}{\arg\min}\, \mathcal{I}\left(\tilde{Z}; Z\right) - \beta \mathcal{I}\left(\tilde{Z}; \bar{Z}\right). \tag{2.37}$$

The distribution we seek to optimise $p(\tilde{Z}|Z)$ can be parametric or nonparametric. It can be for instance the discriminative clustering model with $\tilde{Z} = y$ and $Z = \boldsymbol{x}$.

The hyperparameter β controls the trade-off between a highly detailed compression ($\beta = \infty$) and a one-hot-encoding compression ($\beta = 0$) of Z into $\tilde{Z}$ (Tishby et al., 2000). The specific case of $\beta = 0$ corresponds precisely to the direct maximisation of MI as did Bridle et al. (1992). For more theoretical aspects, we refer to Shamir, Sabato, et Tishby (2010). Although its design makes it compatible with discriminative models, a variational approach to minimisation of the information bottleneck (Alemi, Fischer, Dillon, & Murphy, 2016) was found to produce a similar objective to generative models β -VAE (Higgins et al., 2017).

In a classification context, the output variable $\bar{Z}$ can be replaced by the classes l and the input variables Z by the data samples $\boldsymbol{x}$. In this spirit, Dhillon, Mallela, et Kumar (2003) used clustering as a tool to reduce the dimensions of input texts for a classification task. They wrote the information bottleneck as:

$$\mathcal{L} = \mathcal{I}\left(l; \boldsymbol{x}\right) - \mathcal{I}\left(l; y\right). \tag{2.38}$$

Thus, the information bottleneck compresses the words into clusters y that suffice to guess the class of documents. Notice that in this context, the hyperparameter β was set to 1. Slonim, Atwal, Tkačik, et Bialek (2005) adapted as well this framework for doing only clustering. To that end, they replaced the first mutual information by the expectation of cluster similarity measure $\texttt{sim}$. Taking a random subset of r samples per cluster to evaluate the similarity, their objective is:

$$\mathcal{L} = \mathbb{E}_{y\sim p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x}_{1\cdots r}\sim\prod_{i=1}^{r} p_\theta(\boldsymbol{x}_i|y)}\left[\texttt{sim}(\boldsymbol{x}_{1\cdots r})\right]\right] - \beta\mathcal{I}\left(\boldsymbol{x}; y\right). \tag{2.39}$$

It is interesting to see that in this specific example, one of the mutual informations was replaced to take into account distances between samples of clusters to regulate mutual information between the clusters and the data. We will use a similar implicit regularisation in Chapter 3, which will be the basis of the remaining chapters.

In contrast to the work of Bridle et al. (1992) and Krause et al. (2010), the early works using the information bottleneck framework used nonparametric models (Dhillon et al., 2003 ; Slonim et al., 2005), *i.e.* models that take the form:

$$p_\theta(y = k|\boldsymbol{x} = \boldsymbol{x}_i) = \tau_{ki}, \tag{2.40}$$

where the parameters are $\theta = \{\tau_{ki}\}_{i=1,k=1}^{n,K}$ and constrained such that $\sum_{k=1}^{K} \tau_{ki} = 1$ and τ_{ik} is positive. Such nonparametric models cannot generalise to new samples and are bound to cluster only the samples from the dataset on which they were trained.

2.5.2 Towards deeper networks

Two years after the RIM model, Krizhevsky, Sutskever, et Hinton (2012) introduced the AlexNet model, an example of strong modern deep learning success. Following this advance, depth also affected the neural networks involved in clustering tasks. To the best of our knowledge, the first deep clustering model involving mutual information was proposed by Hu, Miyato, Tokui, Matsumoto, et Sugiyama (2017) who replaced the constraint ℓ_2 of Krause et al. (2010) by a virtual adversarial constraint (Miyato, Maeda, Koyama, & Ishii, 2018). The key idea is to add small perturbations to the input samples and ensure that the assigned cluster is consistent with the clustering of the initial sample. Augmentations such as random cropping, scaling or rotations were also used to perturb the initial sample. The key idea is that these perturbations provide invariances in clustering (Ji, Henriques, & Vedaldi, 2019). Along with the progressive usage of Resnets (K. He, Zhang, Ren, & Sun, 2016) as deep networks, the neural network architecture was changed itself to provide regularisation through auxiliary loss terms (Ren et al., 2022), *e.g.* auxiliary clustering heads (Ji et al., 2019). As an example of regularisation impact: maximising MI with ℓ_2 constraint can be equivalent to a soft and regularised K-Means in a feature space (Jabi, Pedersoli, Mitiche, & Ayed, 2019). This drift towards deep networks and the introduction of data augmentation led to the emergence of contrastive learning as a core basis for deep discriminative clustering methods. We now detail its relationship to mutual information.

2.5.3 From discrete to continuous output variables: contrastive learning and infoMax

With the introduction of deep neural networks in clustering, the related field of *deep clustering* rapidly paved the way to the development of discriminative models that predict a continuous variable $\boldsymbol{z} \in \mathcal{Z}$ instead of a simple cluster assignment $y \in [\![K]\!]$. For example, the variable $\boldsymbol{z}$ can be defined as $\boldsymbol{z} = \psi_\theta(\boldsymbol{x}) + \sigma^2\epsilon$ where ϵ is a unit Gaussian noise (Alemi et al., 2016). The focus hence switched to *representation learning*. These models are no longer designed for clustering, but are compatible with clustering in the learnt representation space. To learn such models, lower bounds of mutual information between $\boldsymbol{x}$ and $\boldsymbol{z}$ are used as objective functions.

Several lower bounds have been designed, as named by Poole et al. (2019): MINE (Belghazi et al., 2018), NCE (Van den Oord, Li, & Vinyals, 2018), BA(Barber & Agakov, 2003). Some of these lower bounds rely on the key property that mutual information is invariant to the addition of a variable $\boldsymbol{v}$ independent of $\boldsymbol{x}$ and $\boldsymbol{z}$, noted shortly: $\mathcal{I}(\boldsymbol{x}, \boldsymbol{v}; \boldsymbol{z}) = \mathcal{I}(\boldsymbol{x}, \boldsymbol{z})$. Consequently, looking at each sample individually and detailing the probability distribution over all samples yields the following:

$$\mathcal{I}(\boldsymbol{x}_1, \boldsymbol{x}_{2\ldots,n}; \boldsymbol{z}) = \mathcal{I}(\boldsymbol{x}_1, \boldsymbol{z}) \,. \tag{2.41}$$

This implies that we can estimate mutual information by summing mutual informations between the representation and each individual samples:

$$\mathcal{I}\left(\boldsymbol{x}_{1,\cdots,n};\boldsymbol{z}\right)=\frac{1}{n}\sum_{i=1}^{n}\mathcal{I}\left(\boldsymbol{x}_i;\boldsymbol{z}\right). \tag{2.42}$$

Poole et al. (2019) show that we can obtain tractable lower bounds of MI in the equation above if we approximate the distribution $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ with a distribution $q(\boldsymbol{x}|\boldsymbol{z})$ taken from the energy-based variational family:

$$q(\boldsymbol{x}|\boldsymbol{z})\propto p_{\text{data}}(\boldsymbol{x})e^{f(\boldsymbol{x},\boldsymbol{z})}, \tag{2.43}$$

where f is a critic function. Specifically, Poole et al. (2019) obtain the NCE lower bound of Van den Oord et al. (2018):

$$\mathcal{I}\left(\boldsymbol{x};\boldsymbol{z}\right)\geq\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x}_{1,\dots n}),p_\theta(\boldsymbol{z}|\boldsymbol{x}_i)}\left[\log\frac{\exp^{f(\boldsymbol{x}_i,\boldsymbol{z}_i)}}{\sum_{j=1}^{n}\exp^{f(\boldsymbol{x}_i,\boldsymbol{z}_j)}}\right]+\log n. \tag{2.44}$$

Due to the constant $\log n$, this estimator of mutual information is biased. The NCE estimator can be linked to the temperature-scaled cross-entropy of contrastive learning (NT-XENT, T. Chen, Kornblith, Norouzi, & Hinton, 2020).

Contrastive learning is an integral part of representation learning through the lens of self-supervised learning. Representation learning consists in finding high-level features $\boldsymbol{z}_i$ extracted from the data $\boldsymbol{x}_i$ to perform a *downstream task*, *e.g.* clustering or classification. However, the nature of the model is different with regards to the mutual information in Eq. (2.44).

The key idea of contrastive learning is to perform a set of random augmentations on a sample $\boldsymbol{x}_i$, then maximise the similarity between the representation associated with this sample and its augmentation, while decreasing the similarity with any other sample. This choice implies that we *no longer* maximise mutual information between the data $\boldsymbol{x}$ and the representation variable $\boldsymbol{z}$, but a variable corresponding to the augmentation of the data $\text{Aug}(\boldsymbol{x})$. This also means that the conditional distribution $p(\text{Aug}(\mathbf{x})|\boldsymbol{x})$ is unknown. Nonetheless, we assume can sample augmentations of $\boldsymbol{x}$ easily. We can rewrite Eq. (2.44) in the context of contrastive clustering as:

$$\mathcal{I}\left(\boldsymbol{x};\text{Aug}(\boldsymbol{x})\right)\geq\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{p(\boldsymbol{x}_{1,\dots n}),p(\text{Aug}(\boldsymbol{x})|\boldsymbol{x}_i)}\left[\frac{\exp^{f_{\text{constrastive}}(\boldsymbol{x}_i,\text{Aug}(\boldsymbol{x}_i))}}{\sum_{j=1}^{n}\exp^{f_{\text{constrastive}}(\boldsymbol{x}_i,\text{Aug}(\boldsymbol{x}_j))}}\right]+\log n. \tag{2.45}$$

Note that in this context, the distributions on which we perform the expectations are not parameterised *any longer* by θ. The parameters of our neural networks are now *hidden in the critic* function $f_{\text{contrastive}}$, which must favour high similarities between samples and their respective augmentations:

$$f_{\text{constrastive}}(\boldsymbol{x}_i,\text{Aug}(\boldsymbol{x}_j))=\texttt{sim}\left(\psi_\theta(\boldsymbol{x}_i),\psi_\theta(\text{Aug}(\boldsymbol{x}_j))\right), \tag{2.46}$$

Contrastive learning provides a convenient framework that takes advantage of the key idea that the neighbourhood of a sample should remain in the neighbourhood of the representation of this sample. In this context, the nature of the neighbourhood is dependent on the choices of data augmentation. Although it is not related to clustering, this approach indirectly views each sample in an individual cluster because we maximise MI between two variations of $\boldsymbol{x}$. This means that the only elements that can go in the same cluster are the augmentations of the samples and called *positive pairs*, whereas all others including their augmentations are called *negative pairs*

and must be excluded. However, such setup completely loses the end goal of clustering that is to put samples into K categories. Therefore, additional tricks are required to bring back the learning of the continuous representations to a clustering model. We give now some examples of such tricks. Note that we purposefully omit some details for the sake of clarity, especially regarding how augmentations are handled through batches during learning and regularisations.

A proposal by Caron et al. (2020) was to add K centroids in the representation space: $\boldsymbol{\mu}_k \in \mathcal{Z}$. Each output of the model $\boldsymbol{z}_i = \psi_\theta(\text{Aug}(\boldsymbol{x}_i))$ is mapped to a linear combination of these centroids. Then, the mutual information between the augmented samples and the centroids is maximised. However, a slightly altered version of NCE is used by Caron et al. (2020).

In the spirit of the SIMCLR model (T. Chen et al., 2020), Do, Tran, et Venkatesh (2021) propose to decompose the model in two different parts. One is a backbone ψ learning common representations for the second part that comprises two different projection heads: one for clustering χ, *i.e.* a softmax-ended function, and φ a projection to the continuous domain. Notice that we omit the parameters for brevity. The model thus comprises two functions, the clustering function:

$$\begin{aligned} \chi \circ \psi : \mathcal{X} &\mapsto \Delta^K, \\ \boldsymbol{x}_i &\mapsto \chi \circ \psi(\boldsymbol{x}_i) = y_i. \end{aligned} \tag{2.47}$$

and the representation function:

$$\begin{aligned} \varphi \circ \psi : \mathcal{X} &\mapsto \mathcal{Z}, \\ \boldsymbol{x}_i &\mapsto \varphi \circ \psi(\boldsymbol{x}_i) = \boldsymbol{z}_i. \end{aligned} \tag{2.48}$$

Then, by summing two mutual informations with different critics f_1 and f_2, one for the representation and one for the clustering, Do et al. (2021) achieve a model with features presenting high intra-cluster variability and low inter-group similarity:

$$\mathcal{L} = \underbrace{\mathcal{I}_{f_1}\left(\boldsymbol{x}; \text{Aug}(\mathbf{x})\right)}_{f_1(\cdot,\cdot)=\texttt{sim}(\varphi\circ\psi(\cdot),\varphi\circ\psi(\cdot))} + \underbrace{\mathcal{I}_{f_2}\left(\boldsymbol{x}; \text{Aug}(\mathbf{x})\right)}_{f_2(\cdot,\cdot)=\texttt{sim}(\chi\circ\psi(\cdot),\chi\circ\psi(\cdot))}. \tag{2.49}$$

Another line of works proposed instead to construct a critic function f that evaluates the similarity between the distribution of the clusters instead of the representations. Let us note the conditional distribution as a vector:

$$\boldsymbol{h}_k = [p_\theta(y = k|\boldsymbol{x}_1), \ldots, p_\theta(y = k|\boldsymbol{x}_i), \ldots, p_\theta(y = k|\boldsymbol{x}_n)] \in \mathbb{R}_n. \tag{2.50}$$

Using this formulation, Huang, Gong, et Zhu (2020) proposed to maximise $\mathcal{I}\left(\boldsymbol{h}_{1,\ldots K}; \bar{\boldsymbol{h}}\right)$ where $\bar{\boldsymbol{h}}$ is the cluster distribution obtained after applying random augmentations on the samples. Y. Li et al. (2021) extended this idea by adding a second mutual information between the representations, as we explained for Do et al. (2021).

The advantage of contrastive learning is the single-stage nature of training. However, the performances of these methods are tied to the choices of augmentations, which may not always be clear cut depending on the data. For instance, basic transformations like translations, Gaussian blur and scaling were shown to be efficient augmentations for classification and segmentation tasks of CT scans (Chlap et al., 2021 ; Garcea, Serra, Lamberti, & Morra, 2023). Yet, this does not imply that these augmentations could be beneficial in an unsupervised context.

Aside contrastive clustering, the InfoMax principle is another framework involving mutual information for learning continuous representations. It was started by Linsker (1988) and its goal was to construct a network such that:

> "The information that reaches a layer is processed so that the maximum amount of information is preserved. We have seen that this does not in general lead to a trivial one-to-one identity mapping [...]." (Linsker, 1988)

This principle was later refined by Hjelm et al. (2019) into the *Deep InfoMax* principle (DIM). Instead of focussing on a layer-wise maximisation of mutual information, the key proposal is to both maximise mutual information between data and clusters as we previously described, but also enforce high mutual information between subsets of visual features of an image and the clusters. Denoting g_{θ_1} the local feature learning function and ψ_{θ_2} the representation function, the DIM can be written as follows:

$$\theta_1^\star, \theta_2^\star = \arg\max_{\theta_1,\theta_2} \mathcal{I}\left(\boldsymbol{x}; \psi_{\theta_2} \circ g_{\theta_1}(\boldsymbol{x})\right) + \lambda \sum_{i=1}^{M} \mathcal{I}\left(g_{\theta_1}(\boldsymbol{x})^{(i)}; \psi_{\theta_2} \circ g_{\theta_1}(\boldsymbol{x})\right), \tag{2.51}$$

where $g_{\theta_1}^{(i)}$ is the i-th subset of M subsets of features, *e.g.* some pixels in an image. Notice that we omitted the prior matching constraint of Hjelm et al. (2019) for clarity. In this original work, the NCE estimator was used for mutual information (Van den Oord et al., 2018), see Eq. (2.44).

The DIM framework has been extended, for example, to focus beyond local subsets of features of data and consider augmentations of the data instead (Bachman, Hjelm, & Buchwalter, 2019) and incorporated into GANs (K. S. Lee, Tran, & Cheung, 2021) to avoid their mode collapse. Nonetheless, we find that the modern meaning of InfoMax differs from mutual information for clustering as seen in the previous sections.

2.5.4 Dissonance between MI and performances

Evaluating mutual information between two continuous random variables is challenging due to the intractability of the underlying integrals. Therefore, lower bounds were derived to alleviate the maximisation of mutual information. However, these bounds can either come with high variance or high bias. That is why Poole et al. (2019) proposed an interpolated lower bound to offer a trade-off between variance and bias: $\mathcal{I}_\alpha$. Nonetheless, it was noticed that MI is hardly predictive of downstream tasks (Tschannen, Djolonga, Rubenstein, Gelly, & Lucic, 2020) with the output continuous variable $\boldsymbol{z}$. In other words, a high value of mutual information does not clarify whether the learnt continuous representations are insightful and can leverage a second-step task such as clustering or classification.

This observation stands in contrast to the most recent papers on deep clustering that achieve good performance in supervised datasets, *e.g.* CIFAR10 and ImageNet (Dang, Deng, Yang, Wei, & Huang, 2021 ; D. H. Lee, Choi, Kim, & Chung, 2022 ; Park et al., 2021). Interestingly, the datasets used for benchmarking often focus on images and rarely other types of data such as tabular data (Min et al., 2018). Overall, it is plausible that the success of these methods may be due to the good design of the discriminative model's architecture which encompasses the underlying assumptions and its regularisations, rather than mutual information itself. Ren et al. (2022) noted that:

> "Due to the complexity brought by massive data, most of the existing deep clustering models are designed for specific data sets. Complex data from different sources and forms bring more uncertainties and challenges to clustering."

Consequently, we empirically observe that the number of clusters to find is an often-discarded question when the evaluation protocol lies on datasets in which the number of *classes*, not clusters, is known. Therefore, an interesting clustering algorithm should be able to find a relevant number of clusters, *i.e.* perform model selection. However, model selection for parametric deep clustering models is expensive (Ronen, Finder, & Freifeld, 2022).

2.6 Interpreting the clusters

Assuming that we get a clustering algorithm selected for a specific number of clusters, we can now enter the second important part of clustering: analysing the content of the clusters. Indeed, the definition of a cluster remains blurry enough to make clustering an ill-posed problem. This encourages a more in-depth study of the nature of the obtained clusters, whether informative or not.

2.6.1 The curse of dimensionality

To analyse the content of the clusters, the first basic step is to look at the variables involved in the clusters and whether some of them play a more crucial role. Although this may sound trivial, it is unfortunately not that easy as the number of variables grows. To some extent, clustering high-dimensional data will eventually fall under the curse of dimensionality (Bellman, 1957), a series of unexpected phenomena that appear for a large number of variables (Bouveyron & Brunet-Saumard, 2014b). This typically concerns multi-omics data, *e.g.* genomics or proteomics, and high-quality images. Consequently, selecting a subset of variables that can be representative of the clusters can facilitate the interpretation of the clusters obtained.

2.6.2 Variable selection

Selecting variables is thus motivated by the facilitation of the clusters' interpretation. However, for d input features, we can compute up to $2^d - 1$ possibilities of selected subsets of features. That is why we need to carefully design an algorithm for selecting the relevant variables.

Feature selection algorithms can be divided into two distinct categories (Dy, 2007 ; John, Kohavi, & Pfleger, 1994): filter methods and wrapper methods. Filter methods apply in an independent step feature selection using a relevance criterion to eliminate irrelevant features before performing clustering. This can be done, for example, using information theory (Cover, 1999) with the SVD-Entropy (Varshavsky, Gottlieb, Linial, & Horn, 2006) or spectral analysis (X. He, Cai, & Niyogi, 2005 ; von Luxburg, 2007 ; Zhao & Liu, 2007). Those methods are thus easily scalable and quick despite the challenge of defining unsupervised feature relevance (Dy, 2007). Wrapper methods encompass the selection process within the model and exploit their clustering results to guide the feature selection (Solorio-Fernández, Carrasco-Ochoa, & Martínez-Trinidad, 2020). Other related works sometimes refer to a third category named hybrid model (Alelyani, Tang, & Liu, 2018) or embedded models (Blum & Langley, 1997) as a compromise between the two first categories.

While the definition of relevance of a variable is more straightforward for supervised learning, its definition in unsupervised learning impacts the choice of selection criterion for filter methods or distribution design in model-based methods (Fop & Murphy, 2018). Often, the terms relevant variables, irrelevant variables (Tadesse, Sha, & Vannucci, 2005) for the notion of conveying information are used. Others may consider redundant variables as those that bring information already

available (Maugis, Celeux, & Martin-Magniette, 2009). Another key difference in the models would be to consider whether the informative variables are independent given the cluster assignment (local independence) or dependent (global independence from the uninformative variables), but the latter hardly accounts for the redundant variables (Fop & Murphy, 2018).

Finally, wrapper models for clustering in feature selection are often model-based (Maugis et al., 2009 ; Raftery & Dean, 2006 ; Scrucca & Raftery, 2018), implying that they assume a parametric mixture model that can explain the distribution of the data, including the distribution of the irrelevant variables. To perform well, these methods need a good selection criterion to compare models with each other (Marbac, Sedki, & Patin, 2020 ; Maugis et al., 2009 ; Raftery & Dean, 2006). However, most of these generative wrapper methods hardly scale both in sample quantity and/or variable quantity. There exists attempts of combining feature selection and discriminative clustering using MI maximisation. For instance, Kong, Deng, et Dai (2015) employ a proximal gradient technique to combine MI with an ℓ_1 (lasso) penalty on the weights. We will come back to the lasso penalty in Chapter 4.We detail some potential discriminative wrapper methods in Section 2.6.4, however, their training is often multi-stage.

2.6.3 Projections and subspaces

Feature selection should not be mistaken for dimensionality reduction, sometimes called feature reduction, which is the process of finding a latent space of lower dimension leveraging good manifolds for clustering, *e.g.* using matrix factorisation (R. Shen et al., 2012). In this sense, methods that seek a sparse subspace for spectral clustering (Peng, Kang, Yang, & Cheng, 2016) or K-means clustering through PCA (Long et al., 2021) are discriminative. However, the nature of the latter forces the clustering to be done according to linear boundaries due to the projections. Still, by enforcing the projection matrix to be sparse, feature selection can be recovered in the original space (Bouveyron & Brunet-Saumard, 2014a). Similarly, subspace clustering seeks to find clusters in different subspaces of the data (H. Chen, Wang, Feng, & He, 2018 ; Zografos, Ellis, & Mester, 2013) and is thus an extension of feature selection (Parsons, Haque, & Liu, 2004), particularly with the motivation that several latent variables could explain the heterogeneity of the data (Vandewalle, 2020). However, such problems usually incorporate a mechanism to merge clusters, which is also challenging, while we are interested in a method that selects features while producing a single clustering output.

It is also possible to perform this projection after the clustering for visualisation purposes (Biernacki, Marbac, & Vandewalle, 2021 ; Scrucca, 2010).

2.6.4 Intrinsically interpretable models

Beyond wrapper methods that perform variable selection along the objective, we can focus specifically on the subclass of intrinsically interpretable models. Not only do these models exploit a subset of selected variables, but they also display how that variable contributes to the clustering through their construction. This is for example the case of Sparse K-means (Witten, Tibshirani, & Witten, 2013). This algorithm benefits from the interpretability of the centroids defined by the K-means algorithm, yet restricts it to a subset of variables.

Recently, the literature has also focused on the creation of unsupervised decision trees (Bertsimas, Orfanoudaki, & Wiberg, 2021 ; Gamlath, Jia, Polak, & Svensson, 2021 ; Laber, Murtinho, & Oliveira, 2023 ; Tavallali, Tavallali, & Singhal, 2021). A decision tree is a set of iterative rules to

apply until no rule is left, in which case a decision is returned. Conveniently, the rules of a decision tree often present a binary decision on a single feature. Thus, a tree not only involves a subset of selected features but also orchestrates their relevance with the order of the rules to apply. These methods are therefore discriminative wrapper methods, but their construction always require a warm-up step, *e.g.* K-means.

CHAPTER 3

The generalised mutual information: a discriminative clustering framework

4.1 Introduction **62**
4.2 Sparse GEMINI **62**
4.2.1 Unsupervised logistic regression architecture 62
4.2.2 The LassoNet architecture 63
4.3 Optimisation **64**
4.3.1 Training and model selection 64
4.3.2 Extension proposal: the dynamic training regime 65
4.3.3 Gradient considerations 65
4.3.4 Implementation in GemClus 65
4.4 Experiments **66**
4.4.1 Metrics 66
4.4.2 Default hyperparameters 66
4.4.3 Numerical experiments 67
4.4.3.1 Specific case of the dynamic training regime 68
4.4.4 Examples on MNIST and variations 71
4.4.5 Real datasets 73
4.4.5.1 OpenML datasets 73
4.4.5.2 Scalability example with the Prostate-BCR dataset 74
4.4.6 Discussion 75
4.5 Conclusion **76**

3.1 Introduction

Clustering is a fundamental learning task that consists in separating data samples into several groups, each named cluster. This task hinges on two main questions concerning the assessment

of correct clustering and the actual number of clusters. However, this problem is ill-posed since a cluster lacks formal definitions which makes it a hard problem (Kleinberg, 2003).

We discussed in Chapter 2 the recent rise of deep clustering methods, mostly based on mutual information (MI) as an objective function to maximise in order to get a discriminative clustering model. Mutual information can be written in two ways, either as a measure of dependency between two variables $\boldsymbol{x}$ and y *e.g.* , data distribution $p(\boldsymbol{x})$ and cluster assignment $p(y)$:

$$\mathcal{I}(\boldsymbol{x}; y) = D_{\mathrm{KL}}\left(p(\boldsymbol{x}, y) \| p(\boldsymbol{x})p(y)\right), \tag{3.1}$$

or as an expected distance between cluster distributions and the data distributions:

$$\mathcal{I}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p(y)}\left[D_{\mathrm{KL}}\left(p(\boldsymbol{x}|y) \| p(\boldsymbol{x})\right)\right], \tag{3.2}$$

with D_{KL} being the Kullback-Leibler (KL) divergence.

It is interesting to observe that the increasing usage of scores derived from mutual information was carried with increasing regularisations. Regularisation techniques were employed to leverage the potential of MI, mostly by specifying model invariances. This started with ℓ_2 regularisation by Krause et al. (2010), later followed by the virtual adversarial training (Miyato, Maeda, et al., 2018) in deeper networks (Hu et al., 2017). Later on, data augmentations were employed to ensure that the clustering model learns representations that are invariant to these augmentations (Ji et al., 2019). The maximisation of MI thus gave way to contrastive learning objectives which aim at learning stable representations of data (Caron et al., 2020 ; T. Chen et al., 2020). Clustering methods also benefited from recent successful deep architectures (Huang et al., 2020 ; Y. Li et al., 2021 ; Tao, Takagi, & Nakata, 2021) by encompassing regularisations in the architecture.

However, most of the methods above rarely discuss their robustness when the number of clusters to find is different from the amount of preexisting known classes. While previous work was essentially motivated by considering MI as a dependence measure (Eq. 3.1), in this chapter we explore the alternative definition of MI as the expected distance between the data distribution implied by the clusters and the entire data (Eq. 3.2). We first start by questioning the quality of mutual information as an objective function without regularisations. Then, we propose an extension that incorporates cluster-wise comparisons of implied distributions and question the choice of the KL divergence with other possible statistical distances. This novel objective is called *generalised mutual information* (GEMINI). Through several qualitative and quantitative experiments, we show how GEMINI is a relevant distance-based objective for discriminative clustering and can address some limitations of MI.

3.2 Is MI a good clustering objective?

3.2.1 Entropy perspectives

Maximising MI directly can be a poor objective when y is continuous: a high MI value is not necessarily predictive of the quality of the features regarding downstream tasks (Tschannen et al., 2020). We support a similar argument for the case where the data $\boldsymbol{x}$ is a continuous random variable and the cluster assignment y is a categorical variable. Indeed, MI can be maximised by setting appropriately a sharp decision boundary that partitions evenly the data, *i.e.* when the distribution $p_\theta(y|\boldsymbol{x})$ converges to a Dirac distribution and $p_\theta(y)$ is the uniform distribution. Rewriting MI in terms of entropies:

$$\mathcal{I}(\boldsymbol{x}; y) = \mathbb{H}(y) - \mathbb{H}(y|\boldsymbol{x}), \tag{3.3}$$

highlights a requirement for balanced clusters, through the cluster entropy term $\mathbb{H}(y)$. Indeed, a uniform distribution maximises the entropy. This hints that an unregularised discrete mutual information for clustering can possibly produce uniformly distributed clusters among samples, regardless of how close they could be. Moreover, any sharp decision boundary minimises the negative conditional entropy, while ensuring balanced clusters maximises the entropy of cluster proportions.

3.2.2 A practical example

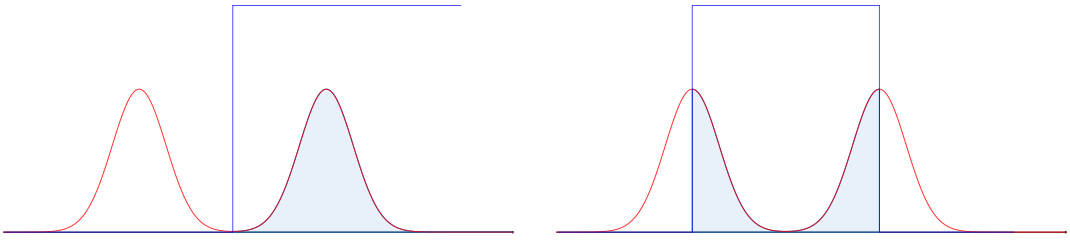
Let us consider a mixture of two Gaussian distributions with different means μ_0 and μ_1, s.t. $\mu_0 < \mu_1$ and same standard deviation σ:

$$p(x|y=0) = \mathcal{N}(x|\mu_0, \sigma^2), p(x|y=1) = \mathcal{N}(x|\mu_1, \sigma^2), \tag{3.4}$$

where y is the cluster assignment. We take balanced clusters proportions, *i.e.* $p(y=0) = p(y=1) = \frac{1}{2}$. This first model is the basis that generated the complete dataset $p(x)$. When performing clustering with a discriminative model, we are not aware of this true distribution. Consequently, we create other models. We want to compute the difference of mutual information between two decision boundaries that two different discriminative models $p_\theta(y|x)$ may yield. Thus, we define two decision boundaries: one which splits evenly the data space called p_A (good boundary) and another which splits it on a closed set p_B (misplaced boundary):

$$p_A(y=1|x) = \begin{cases} 1-\epsilon & x > \frac{\mu_1+\mu_0}{2} \\ \epsilon & \text{otherwise} \end{cases}, \tag{3.5}$$

$$p_B(y=1|x) = \begin{cases} 1-\epsilon & x \in [\mu_0, \mu_1] \\ \epsilon & \text{otherwise} \end{cases}. \tag{3.6}$$



(a) Good decision boundary (b) Misplaced boundary

Figure 3.1 – Example of maximised MI for a Gaussian mixture $\frac{1}{2}\mathcal{N}(\mu_0, \sigma^2) + \frac{1}{2}\mathcal{N}(\mu_1, \sigma^2)$. It is clear that figure 3.1a presents the best decision boundary and posterior between the two Gaussian distributions. Yet, as the posterior turns sharper, the difference between both MIs converges to 0.

Both these models are depicted in Figure 3.1. We show in App A that both models will converge to the same value of mutual information. The sketch of the proof is the following: we start by computing the proportions of clusters $p_A(y)$ (resp. $p_B(y)$), then compute the KL divergences

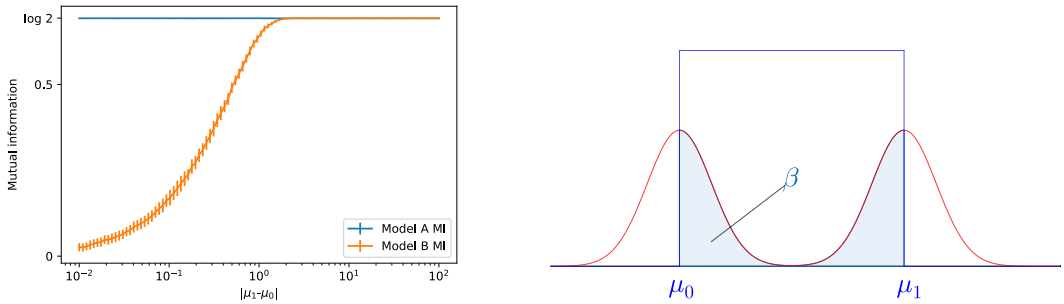
between the distribution $p_A(y|\boldsymbol{x})$ and $p_A(y)$ (resp. $p_B(y|\boldsymbol{x})$ and $p_B(y)$). For the good model A, we obtain a constant term, whereas the KL divergence of the model B depends on the proportion of data captured between μ_0 and μ_1:

$$\beta = \int_{\mu_0}^{\mu_1} p(x)\mathrm{d}x. \tag{3.7}$$

Integrating over the true distribution of the data, we finally obtain the respective values of mutual information for both models. Their difference then depends on the proportion β and the value of ϵ. We finish the demonstration by taking the limit with $\epsilon \longrightarrow 0$ and show:

$$\lim_{\epsilon \to 0} \mathcal{I}_A - \mathcal{I}_B = \log 2 - \mathbb{H}(\beta)\,, \tag{3.8}$$

where $\mathbb{H}(\beta)$ is the entropy.



(a) Differences of MI between models A and B

(b) Gaussian mixture distribution $p(x)$ with proportion β in between the two means μ_0 and μ_1

Figure 3.2 – Value of the mutual information for the two models splitting a Gaussian mixture depending on the distance between the two means μ_0 and μ_1 of the two generating Gaussian distributions. We estimate the MI by computing it 50 times on 1000 samples drawn from the Gaussian mixture.

To conclude, as the decision boundaries turn sharper, *i.e.* when ϵ approaches 0, the difference of mutual information between the two models is controlled by the entropy of proportion of data β between the two means μ_0 and μ_1. We know that for binary entropies, the optimum is reached for $\beta = 0.5$. In other words having μ_0 and μ_1 distant enough to ensure balance of proportions between the two clusters of model B leads to a difference of mutual information equal to 0. We experimentally highlight this convergence in Figure 3.2 where we compute the mutual information of models A and B as the distance between the two means μ_0 and μ_1 increases in the Gaussian distribution mixture. It is interesting to observe that it is easier to misplace the boundaries because of poor local maxima when the Gaussian distributions are far apart instead of close.

Globally, MI misses the idea in clustering that any two points close to one another may be in the same cluster according to some chosen distance. Hence regularisations are required to ensure this constraint. An early sketch of these insights was mentioned by Bridle et al. (1992) or Corduneanu et Jaakkola (2002). A similar example to ours can be found in the work of Ver Steeg, Galstyan, Sha, et DeDeo (2014). Finally, the non-predictiveness of MI was as well recently empirically highlighted by Zhang et Boykov (2023).

3.3 Extending the MI to the GEMINI

Given the identified limitations of MI, we now describe the discriminative clustering framework based on the expected distance equation of mutual information (Eq. 3.2). We then detail the different statistical distances we can use to extend MI to the Generalised Mutual Information (GEMINI).

3.3.1 The discriminative clustering framework for GEMINIs

We only consider two random variables: the data $\boldsymbol{x}$ which can be continuous or discrete and the cluster assignment y which is discrete. Instead of viewing the mutual information as a dependence-seeking objective, we view it as a clustering objective that aims at separating the data distribution given cluster assignments $p(\boldsymbol{x}|y)$ from the data distribution $p(\boldsymbol{x})$ according to the KL divergence:

$$\mathcal{I}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p(y)}\left[D_{\mathrm{KL}}\left(p(\boldsymbol{x}|y)\|p(\boldsymbol{x})\right)\right]. \tag{3.9}$$

To highlight the discriminative clustering design, we explicitly do not set any hypothesis on the data distribution by writing $p_{\text{data}}(\boldsymbol{x})$. The only part of the model that we design is a conditional distribution $p_\theta(y|\boldsymbol{x})$ with parameters θ of some learnable function ψ_θ (Minka, 2005):

$$y|\boldsymbol{x} \sim \text{Categorical}(\psi_\theta(\boldsymbol{x})). \tag{3.10}$$

The function ψ_θ can typically be a neural network of adequate design regarding the data, *e.g.* a CNN or a logistic regression. We assume for this entire section that the distribution is non-degenerate, *i.e.* each cluster has at least one sample assigned, and the distributions $p_\theta(\boldsymbol{x}|y)$ exists and are defined on the entire data space $\mathcal{X}$. Recalling Section 2.3.1, the only other terms we can estimate in discriminative clustering are the cluster proportions through marginalisation using Monte Carlo on i.i.d. samples:

$$p_\theta(y) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[p_\theta(y|\boldsymbol{x})\right]. \tag{3.11}$$

In contrast, the conditional distribution $p_\theta(\boldsymbol{x}|y)$ is unknown because we cannot compute the data distribution:

$$p_\theta(\boldsymbol{x}, y) = p_{\text{data}}(\boldsymbol{x})p_\theta(y|\boldsymbol{x}). \tag{3.12}$$

Fortunately, well-known properties of MI can invert the distributions on which the KL divergence is computed (Bridle et al., 1992 ; Krause et al., 2010) via Bayes' theorem:

$$\mathcal{I}(\boldsymbol{x}; y) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[D_{\mathrm{KL}}\left(p_\theta(y|\boldsymbol{x})\|p_\theta(y)\right)\right], \tag{3.13}$$

which can be estimated using Monte Carlo assuming i.i.d samples from $p_{\text{data}}(\boldsymbol{x})$. Since we highlighted earlier that the KL divergence in MI cannot distinguish inappropriate decision boundaries from better ones, we are interested in replacing it by other distances or divergences. However, replacing the KL in Eq. (3.13) would focus on the separation of cluster assignments from the cluster proportions which may be irrelevant to the data distribution. We rather alter Eq. (3.9) to clearly show that we separate data distributions given clusters from the entire data distribution because it allows us to take into account the data space geometry.

In this context of distance replacement, we question as well the relevance of the chosen distributions to compare. Taking the terminology from ensemble learning with regards to the

combination of multiple binary classifiers (Bishop, 2007, Section 4.1.2), we consider two cases: the *One-vs-All* (OvA) and the *One-vs-One* (OvO). In the context of binary classifiers, OvA means that the classifier must learn to distinguish a single class from any other, leading to as many classifiers as classes, and OvO means that the classifier must learn to distinguish a specific pair of classes, leading to as many classifiers as pairs of classes. In our context, OvA means that we evaluate the distance between one cluster distribution and the data distribution, which encompasses all cluster distributions. OvO means that we evaluate the distance between any pair of cluster distributions, which will be more expensive.

3.3.2 The One-vs-All GEMINI

3.3.2.1 Replacing the Kullback-Leibler divergence with other distances

The goal of GEMINI is to separate data distributions according to an arbitrary distance D, *i.e.* the KL divergence for another divergence or distance in MI. This brings the definition of our first GEMINI, the *One-vs-All*:

$$\mathcal{I}_D^{\text{OvA}}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p_\theta(y)}\left[D\left(p_\theta(\boldsymbol{x}|y)\|p_{\text{data}}(\boldsymbol{x})\right)\right], \tag{3.14}$$

as it compares the distance between the distribution of a specific cluster $p_\theta(\boldsymbol{x}|y)$ (one) against the entire data distribution $p(x)$ (all). This generalisation of MI still respects the dependence-seeking nature of MI as highlighted by Theorem 3.3.1.

Theorem 3.3.1 (Independence in OvA GEMINIs)**.** *Let D be a positive function between probability distributions such that $D(p\|q) = 0 \iff p = q$. Two random variables x and y are independent if and only if the one-vs-all GEMINI $\mathcal{I}_D^{\text{OvA}}(x; y)$ is null.*

Proof. Let D be a distance or divergence between two distributions such that $D(p\|q) = 0 \iff p = q$. Let x and y two random variables. If x and y are independent, then $p(x|y) = p(x)$ and reciprocally. Consequently, the definition of GEMINI unfolds as:

$$\begin{aligned}\mathcal{I}_D^{\text{OvA}}(x; y) &= \mathbb{E}_{y\sim p(y)}\left[D\left(p(x|y)\|p(x)\right)\right], &(3.15)\\ &= \mathbb{E}_{y\sim p(y)}\left[D\left(p(x)\|p(x)\right)\right], &(3.16)\\ &= 0. &(3.17)\end{aligned}$$

The reciprocal is true because D is always positive and the only way to get 0 through the expectation with a non-degenerate clustering distribution is to have D equal to 0. The demonstration also holds in the case of a degenerate distribution by considering the variable y' corresponding the non-empty clusters. □

There exist distances other than the KL to measure how far two distributions p and q are from each other. We can make a clear distinction between two types of distances, Csiszar's f-divergences (Csiszár, 1967) and Integral Probability Metrics (IPM) (Sriperumbudur, Fukumizu, Gretton, Schölkopf, & Lanckriet, 2009). Unlike f-divergences, IPM-derived distances like the Wasserstein distance or the Maximum Mean Discrepancy (MMD) (Gneiting & Raftery, 2007 ; Gretton, Borgwardt, Rasch, Schölkopf, & Smola, 2012) bring knowledge about the data throughout respectively a distance, sometimes called cost, c or a kernel κ: these distances are geometry-aware.

3.3.2.2 f-divergence GEMINIs

These divergences involve a convex function $f : \mathbb{R}^+ \mapsto \mathbb{R}$ such that $f(1) = 0$. This function is applied to evaluate the ratio between two distributions p and q, as in Eq. (3.18):

$$D_f\left(p\|q\right) = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}\left[f\left(\frac{p(\boldsymbol{z})}{q(\boldsymbol{z})}\right)\right]. \tag{3.18}$$

We will focus on three f-divergences: the KL divergence, the Total Variation (TV) distance and the squared Hellinger distance. While the KL divergence is the usual divergence for MI, the TV and the squared Hellinger distance present different advantages among f-divergences. Both of them are bounded between 0 and 1. It is consequently easy to check when any GEMINI using those is maximised contrarily to MI that is bounded by the minimum of the entropies of $\boldsymbol{x}$ and y (Gray & Shields, 1977). When used as distance between data conditional distribution $p_\theta(\boldsymbol{x}|y)$ and data distribution $p_{\text{data}}(\boldsymbol{x})$, we can apply Bayes' theorem in order to get an estimable equation to maximise, which only involves cluster assignment $p_\theta(y|\boldsymbol{x})$ and marginals $p_\theta(y)$ as summarised in Table 3.1, generalising thus the work of Bridle et al. (1992) who did it for the KL divergence. Note that all f-divergences are maximised when the two distributions p and q have disjoint supports (Liese & Vajda, 2006). Common f-divergence like the KL, the squared Hellinger, or the Pearson χ^2 divergence, except for the total variation distance, are specific cases of the α-divergence subclass. The convex function of α-divergence is parameterized by a real number α with:

$$f_\alpha(t) = \begin{cases} \frac{t^\alpha - \alpha t + (\alpha-1)}{\alpha(\alpha-1)}, & \alpha \neq 0, \alpha \neq 1, \\ t\ln t, & \alpha = 1, \\ -\ln t, & \alpha = 0. \end{cases} \tag{3.19}$$

However, the class of α-divergence is inappropriate in some cases for clustering. Indeed, we show with Proposition 3.3.1 (proof in B.1) that the maximisation of α-divergences can lead to any clustering with balanced clusters as the discriminative model $p_\theta(y|\boldsymbol{x})$ converges to a Dirac distribution.

Proposition 3.3.1. *Let $\{\mathcal{X}_k\}_{k=1}^K$ be a partition of $\mathcal{X}$ such that $\mathbb{P}(\boldsymbol{x} \in \mathcal{X}_k) = \frac{1}{K}$. Then for any α-divergence with $\alpha > 0$, OvA GEMINI is upper bounded by a function which depends only on the proportions of the clusters. If the clustering model follows a Dirac distribution: $p_\theta(y = k|\boldsymbol{x}) = \mathbb{1}[\boldsymbol{x} \in \mathcal{X}_k]$, then the upper bound is tight and GEMINI cannot be improved.*

It is worth mentioning in Proposition 3.3.1 that the proportions of the cluster $p(y = k)$ do not matter for the specific case of $\alpha = 2$ to achieve the global maximum, *i.e.* for the Pearson χ^2-divergence. We can infer from Proposition 3.3.1 the specific Corollary 3.3.1 since MI is a case of OvA α-divergence-GEMINI with $\alpha = 1$. We conclude that MI maximisation is a poor objective when a discriminative model can converge to a Dirac distribution as could be the case with very deep neural networks. In a sense, this is a generalisation of the introduction example of Section 3.2.

Corollary 3.3.1. *Let $\{\mathcal{X}_k\}_{k=1}^K$ be a partition of $\mathcal{X}$. Then the mutual information of a discriminative distribution $p(y|\boldsymbol{x})$ is upper bounded by the entropy of $\boldsymbol{x}$ and the upper bound is tight if the distribution is a Dirac model $p(y = k|\boldsymbol{x}) = \mathbb{1}[\boldsymbol{x} \in \mathcal{X}_k]$. The highest upper bound is reached when the partition is balanced.*

3.3.2.3 IPM and Wasserstein-GEMINIs

The IPM is another class of distance that incorporates knowledge from the data through a function f:

$$D_{\text{IPM}}\left(p\|q\right) = \sup_{f\in\mathcal{F}} \left(\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[f(\boldsymbol{z})\right] - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}\left[f(\boldsymbol{z})\right]\right), \tag{3.20}$$

where $\mathcal{F}$ is a set of functions. As backpropagation through suprema could be intractable, we choose to focus on two specific variations of the IPM for GEMINI: the MMD and the Wasserstein distance. Note however that not all Wasserstein distances are IPMs and while some of our propositions are formulated for IPMs, we consider as well the entire class of the Wasserstein distances.

The MMD corresponds to the distance between the respective expected embedding of samples from the distribution p and the distribution q in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$:

$$D_{\text{MMD}}\left(p\|q\right) = \|\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\varphi(\boldsymbol{z})\right] - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}\left[\varphi(\boldsymbol{z})\right]\|_{\mathcal{H}}, \tag{3.21}$$

where φ is the RKHS embedding. To compute this distance, we can use the kernel trick (Gretton et al., 2012) by involving the kernel function $\kappa(\boldsymbol{a},\boldsymbol{b}) = \langle\varphi(\boldsymbol{a}),\varphi(\boldsymbol{b})\rangle$. We then use Bayes' theorem to uncover a version of the MMD that can be estimated through Monte Carlo using only the predictions $p_\theta(y|\boldsymbol{x})$.

The Wasserstein distance is an optimal transport distance. It corresponds to the minimal amount of energy to transform a distribution into another according to an energy function yielding the cost c of moving the mass of a sample from one location to another:

$$D_{\mathcal{W}_c^d}(p\|q) = \left(\inf_{\gamma\in\Gamma(p,q)} \mathbb{E}_{\boldsymbol{x},\boldsymbol{z}\sim\gamma(\boldsymbol{x},\boldsymbol{z})}\left[c(\boldsymbol{x},\boldsymbol{z})^d\right]\right)^{\frac{1}{d}}, \tag{3.22}$$

where $\Gamma(p,q)$ is the set of all couplings between p and q, c a distance function in $\mathcal{X}$ and d a positive integer. Computing the Wasserstein-d distance between two distributions $p_\theta(\boldsymbol{x}|y=k)$ and $p_\theta(\boldsymbol{x})$ is difficult in our discriminative context because we only have access to a finite set of samples N. Note that in the remainder of the paper, we will focus on the Wasserstein-1 metric and note it $D_{\mathcal{W}}$. The Wasserstein-1 distance benefits from a dual definition which takes the form of a supremum over the set of 1-Lipschitz function, thus joining the form of IPMs from Eq. (3.20):

$$D_{\mathcal{W}}\left(p\|q\right) = \sup_{f,\|f\|_L\leq 1} \left(\mathbb{E}_{\boldsymbol{z}\sim p}\left[f(\boldsymbol{z})\right] - \mathbb{E}_{\boldsymbol{z}\sim q}\left[f(\boldsymbol{z})\right]\right). \tag{3.23}$$

Yet, evaluating a supremum as an objective to maximise is hardly compatible with the usual backpropagation in neural networks. This definition was used in attempts to stabilise GAN training (Arjovsky, Chintala, & Bottou, 2017) by using 1-Lipschitz neural networks (Gouk, Frank, Pfahringer, & Cree, 2021). However, the Lipschitz continuity was achieved at the time by weight clipping, whereas other methods such as spectral normalisation (Miyato, Kataoka, Koyama, & Yoshida, 2018) now allow arbitrarily large weights. The restriction of 1-Lipscthiz functions to 1-Lipschitz neural networks does not equal the true Wasserstein distance, and the term "neural net distance" is sometimes preferred (Arora, Ge, Liang, Ma, & Zhang, 2017). Still, estimating the Wasserstein distance using a set of Lipschitz functions derived from neural networks architectures brings more difficulties.

Globally, we hardly experimented the generic IPM for GEMINIs. Our efforts for defining a set of 1-Lipschitz critics, one per cluster for OvA, to perform the neural net distance (Arora et al., 2017)

were not fruitful. This is mainly because such an objective requires the definition of one neural network for the posterior distribution $p_\theta(y|\boldsymbol{x})$ and K other 1-Lipschitz neural networks for the OvA critics, *i.e.* a large number of parameters. Moreover, this brings the problem of designing not only one but many neural networks while the design of one accurate architecture ψ_θ for clustering is already a sufficient problem.

The idea of an expected Wasserstein distance was first proposed by Harchaoui (Harchaoui, 2020, Eq. (48)) under the name one-vs-rest with an additional cluster proportion factor. However, we found that this additional factor is not grounded enough. Notably, Harchaoui used the neural net distance (Arora et al., 2017) for estimating the Wasserstein rather than the true Wasserstein distance in the context of enforced representations by autoencoders for Gaussian mixture models, rather than an end-to-end neural network. Moreover, we can show, using the dual definition of the Wasserstein metric (Eq. 3.23), that the one-vs-rest Wasserstein preliminary work (Harchaoui, 2020) is equivalent to the one-vs-all Wasserstein-GEMINI.

To compute the Wasserstein-GEMINI, we instead use approximations of the distributions with weighted sums of N Diracs:

$$p_\theta(\boldsymbol{x}|y=k) \approx \sum_{i=1}^{N} m_i^k \delta_{\boldsymbol{x}_i}(\boldsymbol{x}) = p_N^k, \text{with} \quad m_i^k = \frac{p_\theta(y=k|\boldsymbol{x}_i)}{\sum_{j=1}^{N} p_\theta(y=k|\boldsymbol{x}_j)}, \tag{3.24}$$

where $\delta_{\boldsymbol{x}_i}$ is a Dirac located at sample location $\boldsymbol{x}_i$. For the distribution p_{data}, we approximate with the empirical distribution: $p_N = \sum_{i=1}^{N} \delta_{\boldsymbol{x}_i}/N$. We state in Prop. 3.3.2 that this empirical estimate of the Wasserstein distance converges to the correct Wasserstein distance. These importance weights are compatible with the `emd2` function of the python optimal transport package Flamary et al. (2021) which requires normalised histograms. Moreover, this implementation gracefully supports automatic differentiation through the earth mover's distance algorithm of Bonneel, Van De Panne, Paris, et Heidrich (2011).

Proposition 3.3.2. *Let $p(\boldsymbol{x}|y=k)$ and $p(\boldsymbol{x})$ be two distributions that we empirically approximate with importance-weighed Dirac estimators p_N^k, and p_N. Then, for all k_1, k_2:*

$$\lim_{N\to+\infty} D_{\mathcal{W}}(p_N^{k_1} \| p_N^{k_2}) = D_{\mathcal{W}}\left(p_\theta(\boldsymbol{x}|y=k_1) \| p_\theta(\boldsymbol{x}|y=k_2)\right), \tag{3.25}$$

and

$$\lim_{N\to+\infty} D_{\mathcal{W}}(p_N^{k_1} \| p_N) = D_{\mathcal{W}}\left(p_\theta(\boldsymbol{x}|y=k_1) \| p_\theta(\boldsymbol{x})\right). \tag{3.26}$$

Proof. We first need to show that p_N^k weakly converges to p. To that end, we will use the Portmanteau theorem (Billingsley, 1999, Theorem 2.1). Let f be any bounded and continuous function. Computing the expectation of $f(\boldsymbol{x})$ with respect to $p_\theta(\boldsymbol{x}|y=k)$ is:

$$\mathbb{E}_{\boldsymbol{x}\sim p_\theta(\boldsymbol{x}|y=k)}\left[f(\boldsymbol{x})\right] = \int_{\mathcal{X}} f(\boldsymbol{x}) p_\theta(\boldsymbol{x}|y=k) d\boldsymbol{x}, \tag{3.27}$$

which can be estimated using self-normalised importance sampling (Owen, 2013, Chapter 9). The proposal distribution we take for sampling is $p_{\text{data}}(\boldsymbol{x})$. Although we cannot evaluate both $p_\theta(\boldsymbol{x}|y)$ and $p_{\text{data}}(\boldsymbol{x})$ up to a constant, we can evaluate their ratio up to a constant which is sufficient:

$$\mathbb{E}_{\boldsymbol{x}\sim p_\theta(\boldsymbol{x}|y=k)}\left[f(\boldsymbol{x})\right] = \int_{\mathcal{X}} f(\boldsymbol{x})\frac{p_\theta(\boldsymbol{x}|y=k)}{p_{\text{data}}(\boldsymbol{x})}p_{\text{data}}(\boldsymbol{x})d\boldsymbol{x}, \tag{3.28}$$

$$= \int_{\mathcal{X}} f(\boldsymbol{x})\frac{p_\theta(y=k|\boldsymbol{x})}{p_\theta(y=k)}p_{\text{data}}(\boldsymbol{x})d\boldsymbol{x}, \tag{3.29}$$

$$\approx \sum_{i=1}^{N} f(\boldsymbol{x}_i)\frac{p_\theta(y=k|\boldsymbol{x}=\boldsymbol{x}_i)}{\sum_{j=1}^{N} p_\theta(y=k|\boldsymbol{x}=\boldsymbol{x}_j)}. \tag{3.30}$$

Now, by noticing in the last line that the importance weights are self normalised and add up to 1, we can identify them as the point masses of our previous Dirac approximations:

$$m_i^k = \frac{p_\theta(y=k|\boldsymbol{x}=\boldsymbol{x}_i)}{\sum_{j=1}^{N} p_\theta(y=k|\boldsymbol{x}=\boldsymbol{x}_j)}. \tag{3.31}$$

This allows to write that the Monte Carlo estimation through importance sampling of the expectation w.r.t $p_\theta(\boldsymbol{x}|y=k)$ is directly the expectation taken on the discrete approximation p_N^k. We can conclude that their is a convergence between the two expectations owing to the law of large numbers:

$$\lim_{N\to+\infty} \mathbb{E}_{\boldsymbol{x}\sim p_N^k(\boldsymbol{x})}\left[f(\boldsymbol{x})\right] = \mathbb{E}_{\boldsymbol{x}\sim p_\theta(\boldsymbol{x}|y=k)}\left[f(\boldsymbol{x})\right]. \tag{3.32}$$

Since f is bounded and continuous, the portmanteau theorem (Billingsley, 1999, Theorem 2.1) states that p_N^k weakly converges to $p_\theta(\boldsymbol{x}|y=k)$ when defining the importance weights as the normalised predictions cluster-wise.

When two series of measures p_N and q_N weakly converge respectively to p and q, so does their Wasserstein distance (Villani, 2009, Corollary 6.9), hence:

$$\lim_{N\to+\infty} D_{\mathcal{W}}\left(p_N^{k_1}\|p_N^{k_2}\right) = D_{\mathcal{W}}\left(p_\theta(\boldsymbol{x}|y=k_1)\|p_\theta(\boldsymbol{x}|y=k_2)\right). \tag{3.33}$$

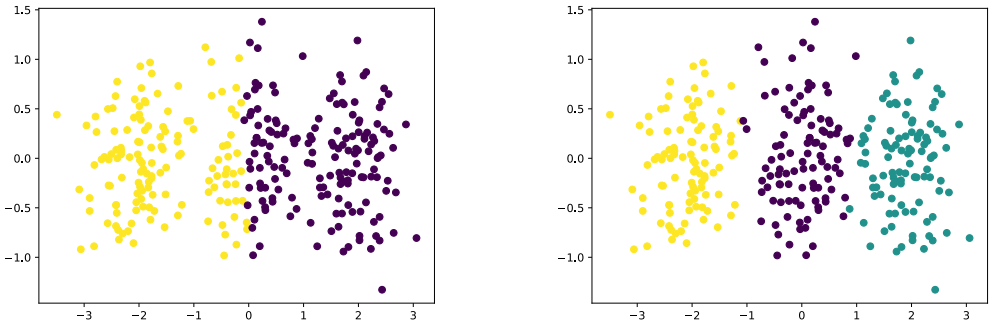
This concludes the proof.

□

3.3.3 The One-vs-One GEMINI

We question the relevance of evaluating a distance between the distribution of the data given a cluster $p_\theta(\boldsymbol{x}|y)$ and the data distribution $p_{\text{data}}(\boldsymbol{x})$ when the geometry is taken into account. We argue that it is intuitive in clustering to compare the distribution of one cluster against the distribution of *another cluster* rather than the data distribution. Indeed, considering the geometry of the data space through a kernel in the case of the MMD or a distance in the case of the Wasserstein metric implies that we can effectively measure how two distributions are close to one another. In the formal design of the mutual information, the distribution of each cluster $p_\theta(\boldsymbol{x}|y)$ is compared to the complete data distribution $p_{\text{data}}(\boldsymbol{x})$. Therefore, if one distribution of a specific cluster $p_\theta(\boldsymbol{x}|y)$ were to look alike the data distribution $p_\theta(\boldsymbol{x})$, for example with identical moments, then its distance to the data distribution could be 0, making it unnoticed when maximising OvA GEMINI.

Take the example of 3 distributions $\{p(\boldsymbol{x}|y=i)\}_{i=1}^3$ with respective different expectations $\{\mu_i\}_{i=1}^3$. We want to separate them using the OvA MMD-GEMINI with linear kernel. The mixture



(a) OvA (b) OvO

Figure 3.3 – Here, 3 clusters of equal proportions from isotropic Gaussian distributions are located in -2, 0 and 2 on the x-axis, with small covariance. The complete data distribution hence has its expectation in 0 on the x-axis. Consequently, maximising the OvA MMD-GEMINI with a logistic regression led to 2 clusters whereas the same model with the OvO MMD-GEMINI is able to see all 3 clusters.

of the 3 distributions creates a data distribution with expectation $\mu = \sum_{i=1}^{3} p(y=i)\mu_i$. However, if the distributions satisfy that this data expectation μ is equal to one of the subexpectations μ_i, then the associated distribution i will not provide any information since its MMD to the data distribution is equal to 0. We illustrate this example in figure 3.3.

To address this issue, we introduce the second GEMINI named *One-vs-One* (OvO) which compares cluster distributions from independent clusters y_a and y_b:

$$\mathcal{I}_D^{\text{OvO}}(\boldsymbol{x}; y) = \mathbb{E}_{y_a, y_b \sim p_\theta(y)} \left[D\left(p_\theta(\boldsymbol{x}|y_a) \| p_\theta(\boldsymbol{x}|y_b) \right) \right]. \tag{3.34}$$

The example of Figure 3.3 is tackled by OvO GEMINI since the distance between each pair of the 3 clusters is non-null. França, Rizzo, et Vogelstein (2020) demonstrated that such an objective is equivalent to kernel K-means in objective, however, their approach was focused only on nonparametric models whereas our approach does not require the specification of centroids and is done through gradient descent. The OvO GEMINI holds the same properties of independence and nullity as OvA GEMINI using the same proof with an additional expectation.

Theorem 3.3.2 (Independence in OvO GEMINIs)**.** *Let D be a positive function between probability distributions such that $D(p\|q) = 0 \iff p = q$. Two random variables x and y are independent if and only if the one-vs-one GEMINI $\mathcal{I}_{OvO}^{D}(\boldsymbol{x}; y)$ is null.*

Note that for most distances, OvO GEMINI is an upper bound of OvA GEMINI (proof in App. B.2).

Proposition 3.3.3. *Let D be an f-divergence or an IPM. Then: $\mathcal{I}_D^{OvA}(\boldsymbol{x}; y) \leq \mathcal{I}_D^{OvO}(\boldsymbol{x}; y)$.*

In the case of binary clustering, using an IPM distance implies equality between OvA GEMINI and OvO GEMINI (proof in B.3).

Proposition 3.3.4. *Let D be an IPM and $p(y|\boldsymbol{x})$ a clustering distribution y taking $K = 2$ values. Then: $\mathcal{I}_D^{OvA}(\boldsymbol{x}; y) = \mathcal{I}_D^{OvO}(\boldsymbol{x}; y)$.*

We can extend Proposition 3.3.1 to all f-divergences for OvO GEMINI with Proposition 3.3.5 (proof in App. B.4). We notably conclude that for the total variation and the squared Hellinger distance, Dirac distributions on an even partition of the data space are the only optimal solutions.

Proposition 3.3.5. *Let D be an f-divergence, $p_\theta(y|\boldsymbol{x})$ a clustering distribution such that $p_\theta(y = k) = \frac{1}{K}$. The OvO GEMINI is then upper bounded by a function depending only on the cluster proportions. For the upper bound to be tight, a sufficient condition is to have disjoint supports between cluster distributions $p_\theta(\boldsymbol{x}|y = k)$. The condition is necessary if the function f satisfies $f(0) + g(0) < \infty$ where $g(t) = tf\left(\frac{1}{t}\right)$ is the convex conjugate of f.*

3.4 GEMINI in practice

Table 3.1 – Definition of the GEMINI for f-divergences, MMD and the Wasserstein distance. We directly write here the equation that can be optimised to train a discriminative model $p_\theta(y|\boldsymbol{x})$ via stochastic gradient descent since they are expectations over the data.

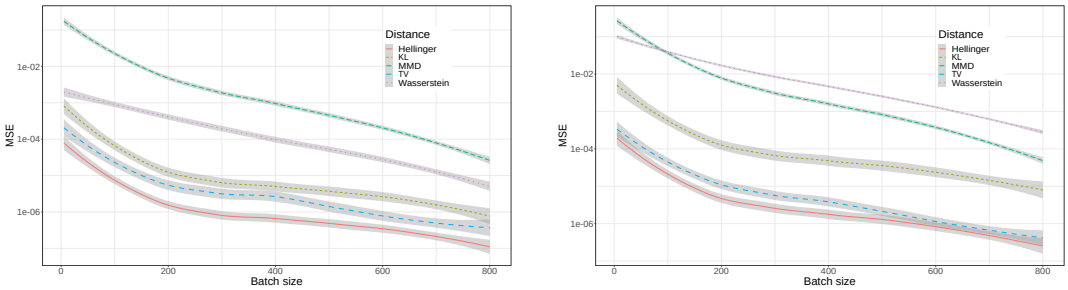
Name	Equation
KL OvA/MI	$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[D_{\text{KL}}\left(p_\theta(y|\boldsymbol{x})\|p_\theta(y)\right)\right]$
KL OvO	$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[D_{\text{KL}}\left(p_\theta(y|\boldsymbol{x})\|p_\theta(y)\right) + D_{\text{KL}}\left(p_\theta(y)\|p_\theta(y|\boldsymbol{x})\right)\right]$
Squared Hellinger OvA	$1 - \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{p_\theta(y)}\left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}}\right]\right]$
Squared Hellinger OvO	$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{V}_{p_\theta(y)}\left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}}\right]\right]$
TV OvA	$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[D_{\text{TV}}\left(p_\theta(y|\boldsymbol{x})\|p_\theta(y)\right)\right]$
TV OvO	$\frac{1}{2}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{y_a,y_b\sim p_\theta(y)}\left[\left\lvert\frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} - \frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)}\right\rvert\right]\right]$
MMD OvA	$\mathbb{E}_{p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[k(\boldsymbol{x}_a,\boldsymbol{x}_b)\left(\frac{p_\theta(y|\boldsymbol{x}_a)p_\theta(y|\boldsymbol{x}_b)}{p_\theta(y)^2} + 1 - 2\frac{p_\theta(y|\boldsymbol{x}_a)}{p_\theta(y)}\right)\right]^{\frac{1}{2}}\right]$
MMD OvO	$\mathbb{E}_{y_a,y_b\sim p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[k(\boldsymbol{x}_a,\boldsymbol{x}_b)\left(\frac{p_\theta(y_a|\boldsymbol{x}_b)p_\theta(y_a|\boldsymbol{x}_b)}{p_\theta(y_a)^2} + \frac{p_\theta(y_b|\boldsymbol{x}_a)p_\theta(y_b|\boldsymbol{x}_b)}{p_\theta(y_b)^2} - 2\frac{p_\theta(y_a|\boldsymbol{x}_a)p_\theta(y_b|\boldsymbol{x}_b)}{p_\theta(y_a)p_\theta(y_b)}\right)\right]^{\frac{1}{2}}\right]$
Wasserstein OvA	$\mathbb{E}_{p_\theta(y)}\left[D_{\mathcal{W}}\left(\sum_{i=1}^N m_i^y\delta_{\boldsymbol{x}_i}\|\sum_{i=1}^N\frac{1}{N}\delta_{\boldsymbol{x}_i}\right)\right]$
Wasserstein OvO	$\mathbb{E}_{y_a,y_b\sim p_\theta(y)}\left[D_{\mathcal{W}}\left(\sum_{i=1}^N m_i^{y_a}\delta_{\boldsymbol{x}_i}\|\sum_{i=1}^N m_i^{y_b}\delta_{\boldsymbol{x}_i}\right)\right]$

3.4.1 Geometric considerations

We stated in Section 3.3.2 that we present GEMINI as a distance between distributions evaluated in the data space $\mathcal{X}$ so that the distance D can take into account the topology of the data. In practice, we only design a discriminative model $p_\theta(y|\boldsymbol{x})$. Therefore, we need to compute all formulas of the

GEMINI through Bayes' theorem to get equations depending on $p_\theta(y|\boldsymbol{x})$ and $p_\theta(y)$. We summarise the equations from all aforementioned GEMINIs in Table 3.1 (see Appendix C for derivations). It is also important to consider the experimental purposes and context to choose a GEMINI. Indeed, when it is easier to design a distance than a kernel, Wasserstein-GEMINI is more compatible than the MMD-GEMINI and vice-versa. Moreover, MMD-GEMINI inherently computes expectations in a Hilbert space which allows computing centroids deemed representative of the clusters. This notion of centroid is less straightforward when using the Wasserstein metric.

3.4.2 Estimating a GEMINI



(a) OvA GEMINIs

(b) OvO GEMINIs

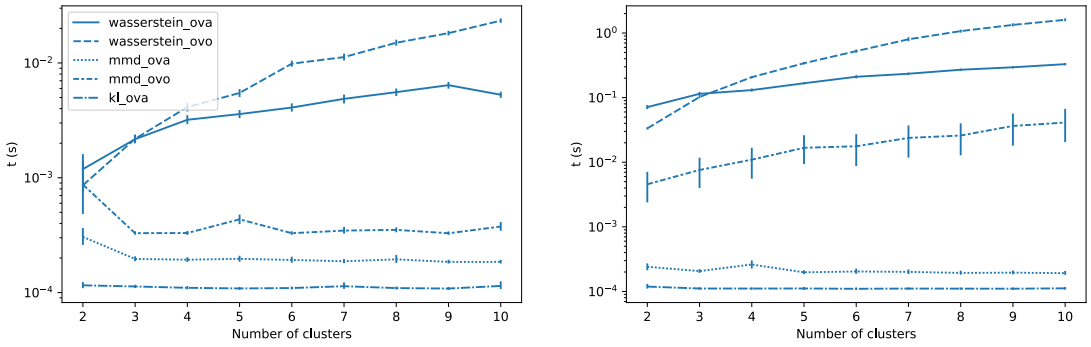
Figure 3.4 – Mean Squared Error (log scale) of estimates with varying batch sizes compared to the true value over a complete dataset of a 1000 samples. Each estimate was performed 50 times per batch size and GEMINI.

All GEMINIs in Table 3.1 can be estimated using Monte Carlo, making them compatible with mini-batch learning, *e.g.* batch sizes of a few hundred for large datasets similarly to prior works (Hjelm et al., 2019 ; Hu et al., 2017 ; Ji et al., 2019). We highlight the importance of the batch size when using GEMINIs. With the use of mini-batch for training, the complete GEMINI is not evaluated on the entire dataset and hence a bias may rise from the empirical estimate. This bias then has consequences on the gradient, which in turn alters training. To illustrate this point, we generated 1000 samples from a Dirichlet distribution with 10 clusters. These samples are a proxy for the output of any discriminative model $p_\theta(y|\boldsymbol{x})$. We then compute the true GEMINI on all samples before evaluating it 50 times for different randomly sampled batches of increasing size. We report in Figure 3.4 the Mean Squared Error of all GEMINIs upon evaluation. We see that past 200 samples for both the OvA and the OvO models, the mean squared error is already close to or below 10^{-2}, except for OvO Wasserstein- and MMD-GEMINIs. This implies an upper bound of 10^{-2} for the bias of the estimates. We conclude that there is possibly a bias in GEMINIs estimates, but it remains small enough to be negligible, *e.g.* smaller than 10^{-1}.

3.4.3 Complexity considerations

The complexity of evaluating GEMINI increases with the distances previously mentionned depending on the number of clusters K and the number of samples per batch N. It ranges from $\mathcal{O}(NK)$ for the usual MI to $\mathcal{O}(K^2N^3 \log N)$ for OvO Wasserstein-GEMINI. Indeed, the usual MI only consists in two sums over N samples and K clusters. Evaluating one Wasserstein distance

takes $N^3 \log N$ (Shirdhonkar & Jacobs, 2008) which is then summed K^2 times for the OvO version of GEMINI. As an example, we show in Figure 3.5 the average time of GEMINI evaluation as the number of tasked clusters increases for both 10 samples per batch (Figure 3.5a) and 500 samples (Figure 3.5b). The batches consists in randomly generated prediction and distances or kernel between randomly generated data.



(a) 10 samples per batch

(b) 500 samples per batch

Figure 3.5 – Average performance time (in seconds) of GEMINIs as the number of tasked clusters grows for batches of size 10 and 500 samples.

OvO Wasserstein is the most complex, and so its usage should remain for 10 clusters or less overall. The second most time-consuming loss is OvA Wasserstein, which is a good compromise between encompassing all data distribution information and computation complexity. The main difference also to notice their memory complexity. OvA MMD requires only $\mathcal{O}(KN^2)$ while OvO MMD requires $\mathcal{O}(K^2N^2)$. This memory complexity should be the major guide to choosing one MMD-GEMINI or the other. Thus, the minimal time-consuming and resource-demanding GEMINI is OvA MMD if we consider GEMINIs that incorporates knowledge of data through kernels and distances. Other versions involving f-divergences have in fact the same complexity as MI in our implementations, apart from OvO TV which reaches $\mathcal{O}(K^2N)$ in our implementation.

3.4.4 Further speed-ups for the OvO Wasserstein

The complexity of the Wasserstein metric is $\mathcal{O}(N^3 \log N)$ for a batch of size N. Consequently, the complexity of the OvO Wasserstein reaches $\mathcal{O}(K^2N^3 \log N)$ for K clusters. This implies that this GEMINI is hardly usable for a high number of clusters. To tackle this complexity, we propose instead to sample T independent pairs among the $K(K-1)/2$ pairs of clusters to compare. We evaluate the OvO Wasserstein on these pairs and scale it to the same order as if performed on all pairs. This is an unbiased estimate of the OvO Wasserstein for all pairs:

$$\hat{\mathcal{I}}^{\text{ovo}}_{D_{\mathcal{W}}}(\boldsymbol{x}; y) = \frac{2T}{K(K-1)} \sum_{k,k' \in I} p_\theta(y=k) p_\theta(y=k') D_{\mathcal{W}} \left(\sum_{i=1}^{N} m_i^k \delta_{\boldsymbol{x}_i} \| \sum_{i=1}^{N} m_i^{k'} \delta_{\boldsymbol{x}_i} \right), \quad (3.35)$$

with

$$I = \{(k_t, k'_t)\}_{t=1}^{T} \quad ; (k_n, k'_n) \sim \mathcal{U}\left([\![K]\!] \times [\![K]\!]\right), \quad (3.36)$$

```python
from gemclus.linear import LinearMMD
from sklearn.datasets import load_iris
# load data
X, _ = load_iris(return_X_y=True)
# Perform clustering with a logistic regression
y_pred = LinearMMD(n_clusters=3).fit_predict(X)
```

Listing 3.1 – An example of logistic regression model trained with the OvA-MMD-GEMINI on the iris dataset. Default kernel is linear.

```python
from gemclus.mlp import MLPWasserstein
from sklearn.datasets import load_iris
# load data
X, _ = load_iris(return_X_y=True)
# Perform clustering with a MLP
y_pred = MLPWasserstein(n_clusters=3, ovo=True, n_hidden_dim
    =3).fit_predict(X)
```

Listing 3.2 – An example of an MLP model trained with the OvO-Wasserstein-GEMINI on the iris dataset. Default distance is Euclidean.

a set of uniformly drawn pairs of clusters.

This optimisation requires however longer training time as a tradeoff for a controlled complexity of $\mathcal{O}(TN^3)$. Indeed, by comparing fewer pairs of clusters that are uniformly selected, we may need more epochs to ensure that all pairs of clusters are compared enough times. Note that the same optimisation can be applied to the OvA Wasserstein-GEMINI.

3.4.5 The *GemClus* package

Owing to the exact computations of the gradients that we detail in the appendices C and D, we developped a Python package encompassing all GEMINI methods named *GemClus*. Overall, the package is aimed for small datasets as it is implemented on CPU only [*]. We give an example of code in the listings 3.1 and 3.2.

Beyond the scope of learning models for clustering, GEMINIs can be isolated for scoring other model predictions, as shown in Listing 3.3.

We want as well to extend this package to other discriminative clustering methods for potentially small-scale datasets. Therefore, we include an implementation of the regularised mutual information (RIM) model by Krause et al. (2010) as shown in Listing 3.4 because we consider this model to be one of the very first proposed in the domain yet find few satisfying implementations.

This package aims at containing most of the essential code of this thesis and is also expected to grow beyond this specific scope. At the moment of writing, the package is publicly available with the version 1.0.0.

∗. The gemclus package can be found at https://gemini-clustering.github.io/.

```python
from gemclus.gemini import WassersteinGEMINI
# Load some data and fit a model to it
X, y_pred = ... # y_pred has a shape of n_samples x n_clusters
# Create the GEMINI for scoring with default Euclidean metric
gemini = WassersteinGEMINI(ovo=True)
# Compute the distance matrix then evaluate the GEMINI
D = gemini.compute_affinity(X)
score = gemini(y_pred, D)
```

Listing 3.3 – Using GEMINI for scoring the predictions of another model

```python
from gemclus.linear import RIM
y_pred = RIM(n_clusters=3).fit_predict(X)
```

Listing 3.4 – The package *gemclus* incorporates as well the basic logistic regression with mutual information regularised by an ℓ_2 penalty (RIM).

3.5 Experiments

For all experiments below, we report the Adjusted Rand Index (ARI) (Hubert & Arabie, 1985), a common metric in clustering. This metric is external as it requires labels for evaluation. It ranges from 0, when labels are independent of cluster assignments, to 1, when labels are equivalent to cluster assignments up to permutations. An ARI close to 0 is equivalent to the best accuracy when voting constantly for the majority class, *e.g.* 10% on a balanced 10-class dataset. Regarding the MMD- and Wasserstein-GEMINIs, we used by default a linear kernel and the Euclidean distance unless specified otherwise. All discriminative models are trained using the Adam optimiser (Kingma & Ba, 2014).

We start by giving detailed examples where GEMINI overcomes the limitations of MI followed by examples to illustrate the compatibility of the method with various models and metrics, before concluding on a practical example: the Enron email dataset.

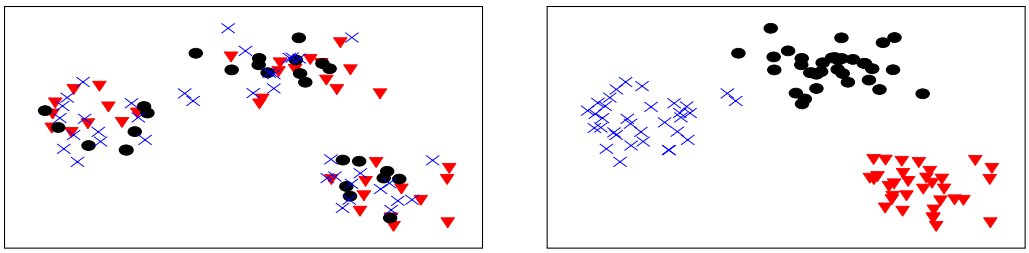
3.5.1 When MI fails because of the modelling

We first took the most simple discriminative clustering model, where each cluster assignment according to the input datum follows a categorical distribution:

$$y|\boldsymbol{x} = \boldsymbol{x}_i \sim \text{Categorical}(\boldsymbol{\theta}_{i1}, \boldsymbol{\theta}_{i2}, \cdots, \boldsymbol{\theta}_{iK}). \tag{3.37}$$

We generated $n = 100$ samples from a simple mixture of $K = 3$ Gaussian distributions. Each model thus only consists in nK parameters to optimise. This is a simplistic way of describing the most flexible deep neural network. We then maximised on the one hand OvA KL (MI) and on the other hand OvA MMD. Both clustering results can be seen in Figure 3.6. We concluded that without any function, *e.g.* a neural network, to link the parameters of the conditional distribution with $\boldsymbol{x}$, MI struggles to find the correct decision boundaries. Indeed, the position of $\boldsymbol{x}$ in the 2D space plays no role and the decision boundary becomes only relevant with regards to cluster entropy

maximisation: a uniform distribution between 3 clusters. However, it plays a major role in the kernel of the MMD-GEMINI thus correctly solving the problem.

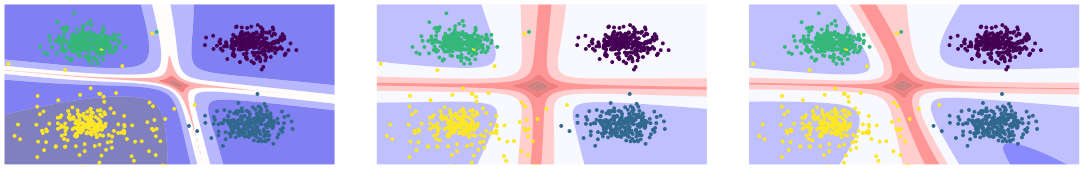


(a) OvA KL (MI) (b) OvA MMD with linear kernel

Figure 3.6 – Clustering of a mixture of 3 Gaussian distributions with MI (left) and GEMINI (right) using categorical distributions. MI does not have insights on the data shape because of the model, and clusters points uniformly between the 3 clusters (black dots, red triangles and blue crosses) whereas MMD-GEMINI is aware of the data shape through its kernel.

3.5.2 Resistance to outliers

To prove the strength of using neural networks for clustering trained with GEMINI, we introduced extreme samples in Gaussian mixtures by replacing a Gaussian distribution with a Student-t distribution for which the degree of freedom ρ is small. We fixed $K = 4$ clusters, 3 being drawn from multivariate Gaussian distributions and the last one from a multivariate Student-t distribution with 1 degree of freedom or 2. Thus, the Student-t distribution produces samples that can be perceived as outliers regarding a Gaussian mixture owing to its heavy tail.



(a) OvA KL (MI) Entropy Map (b) OvO MMD Entropy Map (c) OvO Wasserstein Entropy Map

Figure 3.7 – Entropy maps of the predictions of each MLP trained using a GEMINI or the MI. The bottom-left distribution (yellow) is a Student-t distribution with 1 degree of freedom that produces samples far from the origin. The Rényi entropy of prediction is highlighted from lowest (blue background) to highest (red background). MI on the left has the most confident predictions overall and the smallest uncertainty around the decision boundary, *i.e.* high entropy variations.

Each cluster distribution is centred around a mean μ_i whose proximity is controlled by a scalar α. For simplicity, all covariance matrices are the identity. We define:

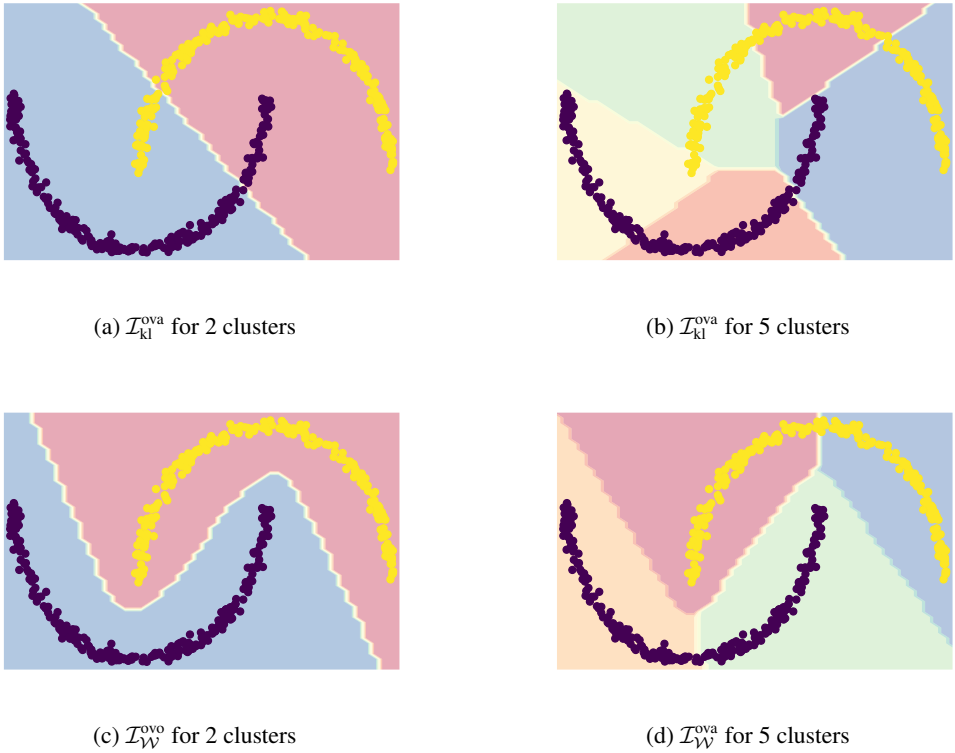
$$\mu_1 = [\alpha, \alpha], \qquad \mu_2 = [\alpha, -\alpha],$$
$$\mu_3 = [-\alpha, \alpha], \qquad \mu_4 = [-\alpha, -\alpha].$$

Table 3.2 – Mean ARI $_{\text{std}}$ of a MLP fitting a mixture of 3 Gaussian and 1 Student-t multivariate distributions compared with Gaussian Mixture Models and K-Means. The Student-t distribution has ρ=1 or 2 degrees of freedom and the model can be designed to find either 4 or 8 clusters at best. Bottom line presents the ARI for the maximum a posteriori of an oracle aware of all parameters of the data.

Model	$\rho = 2$		$\rho = 1$	
	4 clusters	8 clusters	4 clusters	8 clusters
K-Means	0.965 $_{0}$	0.897 $_{0.040}$	0 $_{0}$	0.657 $_{0.008}$
GMM $_{\text{full covariance}}$	0.972 $_{0}$	0.868 $_{0.042}$	0 $_{0}$	0.610 $_{0.117}$
GMM $_{\text{diagonal covariance}}$	**0.973** $_{\mathbf{0}}$	0.862 $_{0.048}$	0.024 $_{0.107}$	0.660 $_{0.097}$
$\mathcal{I}^{\text{ova}}_{\text{KL}}$	0.883 $_{0.182}$	0.761 $_{0.101}$	**0.939** $_{\mathbf{0.006}}$	0.742 $_{0.092}$
$\mathcal{I}^{\text{ovo}}_{\text{KL}}$	0.731 $_{0.140}$	0.891 $_{0.129}$	0.723 $_{0.114}$	0.755 $_{0.163}$
$\mathcal{I}^{\text{ova}}_{\text{H}^2}$	0.923 $_{0.125}$	**0.959** $_{\mathbf{0.043}}$	0.906 $_{0.103}$	0.86 $_{0.087}$
$\mathcal{I}^{\text{ovo}}_{\text{H}^2}$	0.926 $_{0.112}$	0.951 $_{0.059}$	0.858 $_{0.143}$	0.887 $_{0.074}$
$\mathcal{I}^{\text{ova}}_{\text{TV}}$	0.940 $_{0.097}$	0.973 $_{0.004}$	0.904 $_{0.104}$	**0.925** $_{\mathbf{0.103}}$
$\mathcal{I}^{\text{ovo}}_{\text{TV}}$	0.971 $_{0.005}$	0.620 $_{0.053}$	**0.938** $_{\mathbf{0.005}}$	0.595 $_{0.055}$
$\mathcal{I}^{\text{ova}}_{\text{MMD}}$	0.953 $_{0.060}$	0.940 $_{0.033}$	0.922 $_{0.004}$	**0.908** $_{\mathbf{0.016}}$
$\mathcal{I}^{\text{ovo}}_{\text{MMD}}$	0.968 $_{0.001}$	0.771 $_{0.071}$	0.921 $_{0.007}$	0.849 $_{0.048}$
$\mathcal{I}^{\text{ova}}_{\mathcal{W}}$	0.897 $_{0.096}$	0.896 $_{0.021}$	0.915 $_{0.131}$	0.889 $_{0.051}$
$\mathcal{I}^{\text{ovo}}_{\mathcal{W}}$	0.970 $_{0.002}$	0.803 $_{0.067}$	0.922 $_{0.006}$	0.817 $_{0.042}$
Oracle	0.991		0.989	

To sample from a multivariate Student-t distribution, we first draw samples $\boldsymbol{x}$ from a centred multivariate Gaussian distribution. We then sample another variable u from a χ^2-distribution using the degrees of freedom ρ as parameter. Finally, $\boldsymbol{x}$ is multiplied by $\sqrt{\frac{\rho}{u}}$, yielding samples from the Student-t distribution.

We report the ARIs of Multi-Layered Perceptron (MLP) trained 20 times with GEMINIs in Table 3.2. The presence of "outliers" leads K-Means and Gaussian Mixture models to fail at grasping the 4 distributions when trying to find 4 clusters. Meanwhile, GEMINIs perform better. Note that MMD- and Wasserstein-GEMINI present lower standard deviation for high scores compared to f-divergence GEMINIs. We attribute these performances to both the MLP that tries to find separating hyperplanes in the data space and the absence of hypotheses regarding the data. Moreover, MI is best maximised when its decision boundary presents little entropy $\mathbb{H}(y|\boldsymbol{x})$. As neural networks can be overconfident (Guo, Pleiss, Sun, & Weinberger, 2017), MI is likely to yield overconfident clustering by minimizing the conditional entropy. We highlight such behaviour in Figure 3.7 where the Rényi entropy (Rényi, 1961) associated with each sample in MI (Figure 3.7a) is much lower, if not 0, compared to OvO MMD and OvO Wasserstein (figures 3.7b and 3.7c). We conclude that Wasserstein- and MMD-GEMINIs train neural networks to less overconfidence, hence yielding more moderate distributions $p_\theta(y|\boldsymbol{x})$.



(a) $\mathcal{I}_{\mathrm{kl}}^{\mathrm{ova}}$ for 2 clusters

(b) $\mathcal{I}_{\mathrm{kl}}^{\mathrm{ova}}$ for 5 clusters

(c) $\mathcal{I}_{\mathcal{W}}^{\mathrm{ovo}}$ for 2 clusters

(d) $\mathcal{I}_{\mathcal{W}}^{\mathrm{ova}}$ for 5 clusters

Figure 3.8 – Fitting hand-generated moons using the GEMINI on top of an MLP for different amount of clusters. The OvO Wasserstein model with 5 clusters eventually found 4 clusters.

3.5.3 Leveraging a manifold geometry

We highlighted that MI can be maximised without requiring a suitable decision boundary. Here, we show how the provided distance to OvO Wasserstein-GEMINI can leverage appropriate clustering when we have a good a priori on the data.

To that end, we create a dataset with two interlacing moon-shaped clusters. The cluster assignment is drawn according to a Bernoulli distribution with parameter 0.5. The points are then sampled using a uniformly distributed angle in $[0, \pi]$. Axis symmetry is applied depending on the clusters. All points are distributed at the same radius ρ from some circle centre before adding some Gaussian noise. The samples then undergo some offset so that both moons are not linearly separable.

To construct a distance c for the Wasserstein distance, we derived a distance from the Floyd-Warshall algorithm (Roy, 1959 ; Warshall, 1962) on a sparse graph describing neighbourhoods of samples. This distance describes how many neighbours are in between two samples. To compute it, we first use a sub-metric that we note d, in this example the ℓ_2 norm. This allows us to compute all distances d_{ij} between every sample i and j. From this matrix of sub-distances, we can build a graph adjacency matrix W following the rules:

$$W_{ij} = \begin{cases} 1 & d_{ij} \leq \epsilon \\ 0 & d_{ij} > \epsilon \end{cases}, \tag{3.38}$$

where ϵ is a chosen threshold such that the graph has sparse edges. Our typical choice for ϵ is the 5% quantile of all d_{ij}.

We chose the graph adjacency matrix to be unweighted and undirected, owing to the symmetry of d_{ij}. Indeed, solving the all-pairs shortest paths involves the Floyd-Warshall algorithm (Roy, 1959 ; Warshall, 1962) which complexity $\mathcal{O}(n^3)$ is not affordable when the number of samples n becomes large. An undirected and unweighted graph leverages performing n times the breadth-first-search algorithm, yielding a total complexity of $\mathcal{O}(n^2 + ne)$ where e is the number of edges. Consequently, setting a good threshold ϵ controls the complexity of the shortest paths to finds. Our final distance between two nodes i and j is eventually:

$$c_{ij} = \begin{cases} \text{Shortest-path}^W(i,j) & \text{if it exists.} \\ n & \text{otherwise} \end{cases} . \tag{3.39}$$

This metric c can then be incorporated inside the Wasserstein-GEMINI.

We report the different decision boundaries in Figure 3.8. We observe that the insight on the neighbourhood provided by our distance c helped the MLP to converge to the correct solution with an appropriate decision boundary, unlike MI. Note that the usual Euclidean distance in the Wasserstein metric would have converged to a solution similar to MI. Indeed for 2 clusters, the optimal transport plan has a larger value using a distribution similar to Figure 3.8a than in Figure 3.8c. This toy example shows how an insightful metric provided to the Wasserstein distance in GEMINIs can lead to correct decision boundaries while only designing a discriminative distribution $p_\theta(y|\boldsymbol{x})$ and a distance c.

In addition, we highlight an interesting behaviour of all GEMINIs. During optimisation, it is possible that the model converges to using fewer clusters than the number to find. For example in Figure 3.8, for 5 clusters, the model can converge to 4 balanced clusters and 1 empty cluster (Figure 3.8d) unlike MI that produced 5 misplaced clusters (Figure 3.8b). Indeed, the entropy on the cluster proportion in MI forces to use the maximum number of clusters.

3.5.4 Fitting MNIST

We trained a neural network using either MI or GEMINIs. Following Hu et al. (2017), we first tried with a MLP with one single hidden layer of dimension 1200. To further illustrate the robustness of the method and its adaptability to other architectures, we also experimented using a LeNet-5 architecture (LeCun, Bottou, Bengio, & Haffner, 1998) since it is adequate to the MNIST dataset. We report our results in Table 3.3. Since we are dealing with a clustering method, we may not know the number of clusters a priori in a dataset. The only thing that can be said about MNIST is that there are *at least* 10 clusters, one per digit. Indeed, the writings of digits could differ leading to more clusters than the number of classes. That is why we further tested the same method with 15 clusters to find in Table 3.3. We first see that the scores of MMD and Wasserstein-GEMINIs are greater than MI. We also observe that no f-divergence-GEMINI always yields the best ARIs. Nonetheless, we observe better performances in the case of the TV-GEMINIs that we attribute to its bounded gradient. This results in controlled stepsize when doing gradient descent contrarily to KL- and squared Hellinger-GEMINIs. Notice that the change of architecture from an MLP to a LeNet-5 unexpectedly halves the scores for the f-divergences. We believe this drop is due to the change in the notion of neighbourhood implied by the network architecture.

Table 3.3 – ARI for deep neural network trained with GEMINIs on MNIST for 500 epochs. Models were trained either with either 10 clusters to find or 15. We indicate in parentheses the number of used clusters by the model after training.

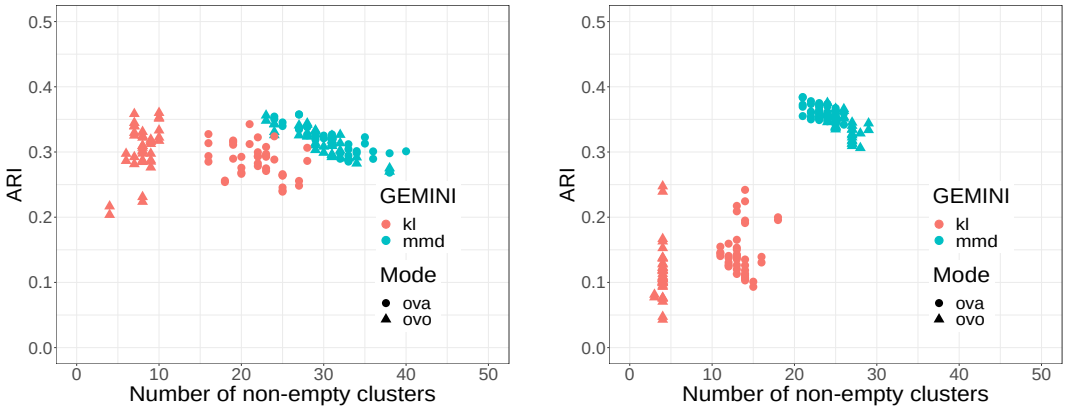
GEMINI		10 clusters		15 clusters	
		MLP	LeNet-5	MLP	LeNet-5
KL	OvA	0.320 (10)	0.138 (8)	0.271 (15)	0.136 (12)
	OvO	0.348 (7)	0.123 (4)	0.333 (8)	0.104 (4)
Squared Hellinger	OvA	0.301 (10)	0.207 (6)	0.224 (13)	0.162 (7)
	OvO	0.287 (10)	0.161 (6)	0.305 (13)	0.157 (7)
TV	OvA	0.299 (10)	0.171 (6)	0.277 (15)	0.140 (6)
	OvO	0.422 (10)	0.161 (9)	0.330 (15)	0.182 (14)
MMD	OvA	0.373 (10)	0.382 (10)	0.345 (15)	0.381 (15)
	OvO	0.361 (10)	0.373 (10)	0.364 (15)	0.379 (15)
Wasserstein	OvA	**0.471** (10)	**0.463** (10)	0.390 (15)	**0.446** (11)
	OvO	0.450 (10)	0.383 (10)	**0.415** (15)	0.414 (15)
K-Means		0.367		0.385	

3.5.5 Number of non-empty clusters and architecture

We repeat our previous experiment on the MNIST dataset from Sec. 3.5.4. We choose this time to get 50 clusters at best for both MI and the MMD-GEMINI and train the models for 100 epochs. We did not choose to test with the Wasserstein-GEMINI because its complexity implies a long training time for 50 clusters. We repeat the experiment 20 times per model and plot the resulting scores in Figure 3.9. We first observe that MMD-GEMINI with linear kernel has a tendency to exploit more clusters than MI. The model converges to approximately 30 clusters in the case of the MLP and 25 for the LeNet-5 model with less variance. We can further observe that for all metrics the choice of architecture impacted the number of non-empty clusters after training. Indeed, by playing a key role in the decision boundary shape, the architecture may limit the number of clusters to be found: the MLP can draw more complex boundaries compared to the LeNet5 model. Moreover, we suppose that the cluster selection behaviour of GEMINI may be due to optimisation processes. Indeed, we optimise estimators of GEMINI rather than the exact GEMINI. Finally, Fig. 3.9 also confirms from Table. 3.3 the stability of MMD-GEMINI regarding the ARI despite the change of architecture whereas MI is affected and shows poor performance with the LeNet-5 architecture.

3.5.6 Number of clusters and training time

We observed in the previous experiments that models trained with GEMINI may not always converge to the desired number of clusters. We go further here by showing that in a general, a longer training time implies better chances of getting the desired number of clusters. To that end, we generate a dataset called *barrel* which consists in a balanced mixture of 10 isotropic Gaussian distributions circularly and evenly distributed around the origin of $\mathbb{R}^2$ with sufficiently small covariance to guarantee a good separation of clusters. We then train MLPs with a hidden



(a) MLP (b) LeNet5

Figure 3.9 – Distributions of the ARI scores given a number of non-empty clusters after 100 epochs of training on MNIST on two different architectures.

size of 20 nodes with all GEMINIs for 10,000 epochs. A model thus contains a greater number of nodes than required to separate correctly the distributions. With Figure 3.10, we conclude that a longer training time implies an increase in the number of non-empty clusters. However, none of the models reached the 10 clusters, even after 10,000 epochs. Still, we note that in general, OvO GEMINIs tend to converge faster to the targeted number of clusters than OvA GEMINIs, owing to their cluster-wise comparative nature. In the specific cases of KL- and Hellinger-GEMINIs, we can respectively explain the failure due to the cancellation of the cluster entropy term emerging from the symmetric KL, and to the unrelated nature of a variance expectation regarding the number of clusters. Hence these objectives do not encourage the model to reach the targeted number of clusters. Finally, we observe a stronger tendency for MMD-GEMINI to stabilise sooner around a fixed number of clusters while others keep on increasing. From these observations, we advise to start any training with an OvA GEMINI for ease of computations, then switch to OvO GEMINI once the training reaches a fixed number of clusters. If possible, a longer training might present chances of a greater number of clusters if the current number is not satisfactory.

3.5.7 Cifar10 clustering using a SIMCLR-derived kernel

To further illustrate the benefits of the kernel or distance provided to GEMINIs, we continue the same experiment as in section 3.5.4. However, we focus this time on the CIFAR10 dataset. As an improved distance, we chose a linear kernel and ℓ_2 norm between features extracted from a pretrained SIMCLR model (T. Chen et al., 2020). We provide results for two different architectures: LeNet-5 and ResNet-18 both trained from scratch on raw images, the latter being a common choice of models in deep clustering literature (Tao et al., 2021 ; Van Gansbeke, Vandenhende, Georgoulis, Proesmans, & Van Gool, 2020). We report the results in Table 3.4 and provide the baseline of MI. We also add the baselines from related works when not using data augmentations to make a fair comparison. Indeed, models trained with GEMINIs do not use data augmentation: only the architecture and the kernel or distance function in the data space plays a role. We observe here that the choice of kernel or distance can be critical in GEMINIs. While the Euclidean norm between images does not provide insights on how images are far as shown by K-Means, features derived from

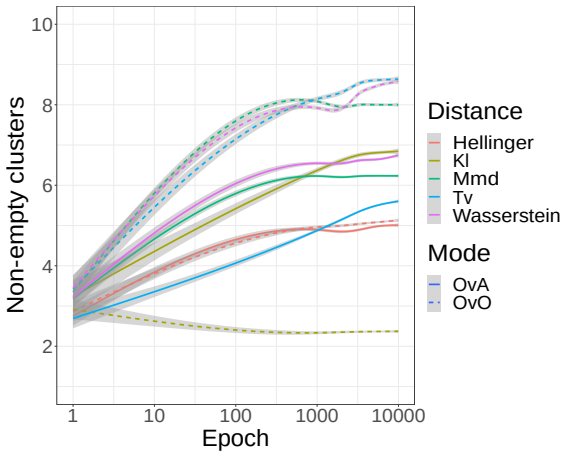


Figure 3.10 – Number of non-empty clusters on average for 30 GEMINI models trained during 10,000 epochs on the barrel dataset. Shaded areas correspond to the 95% confidence intervals.

SIMCLR carry much more insight on the proximity of images. This shows that the performances of GEMINIs depend on the quality of distance functions. Interestingly, we observe that for the Resnet-18 using SIMCLR features to guide GEMINIs was not as successful as it has been on the LeNet-5. We believe that the ability of this network to draw any decision boundary makes it equivalent to a categorical distribution model as in Sec. 3.5.1. Finally, to the best of our knowledge, we are the first to train from scratch a standard discriminative neural network on CIFAR10 raw images without using labels or direct data augmentations, while getting sensible clustering results. However, other recent clustering methods achieve best scores using data augmentations which we do not (Park et al., 2021).

3.5.8 A practical application with Graph Neural Networks: the Enron email dataset

We focus in this experiment on the clustering of nodes in a graph representing email interaction between individuals in the Enron dataset (available at `https://www.cs.cmu.edu/~./enron/`). This famous company was filed for bankruptcy on the 2nd of December 2001 following an investigation for fraud by the Securities and Exchange Commision (SEC). Following Bouveyron, Latouche, et Zreik (2018), we focus on the exchange of emails in the Enron corporation between the 1st of September 2001 and the 31st of December 2001, which corresponds to the peak period at which the company collapsed. We represent the dataset as a graph where each node corresponds to an employee and each edge represents the sending of at least one email. We choose to keep the edges unweighted because the number of emails sent between two persons is not necessarily reflexive of how they can be close to each other. After filtering nodes that do not communicate between September and December, the graph comprises 141 nodes and 872 symmetric edges. In order to use the Wasserstein metric, we choose as a distance the all-pairs-shortest path. Due to the necessity of using a *symmetric* distance for the Wasserstein metric, we left the graph undirected. Thus, an edge between two persons represents the exchange of at least one email anyhow.

To cluster the nodes of the graph, we adopt a simple Graph Convolution Network (GCN) inspired by Kipf et Welling (2016) and Liang, Corneli, Bouveyron, et Latouche (2022) with a hidden size of dimension 64 and 10 clusters to find at best (Bouveyron et al., 2018). We vary the

Table 3.4 – ARI score of models trained for 200 epochs on CIFAR10 with different architectures using GEMINIs. The kernel for the MMD is either a linear kernel or the dot product between features extracted from a pretrained SIMCLR model. Both the Euclidean norm between images and SIMCLR features are considered for the Wasserstein metric. We report the ARI of related works when not using data augmentation for comparison.*: scores reported from Y. Li et al. (2021).

GEMINI	Metric	ARI with LeNet-5	ARI with Resnet-18
$\mathcal{I}^{\text{ova}}_{\text{KL}}$	$\emptyset$	0.02	0.008
$\mathcal{I}^{\text{ova}}_{\text{MMD}}$	Euclidean	0.049	0.047
	SIMCLR	**0.157**	0.122
$\mathcal{I}^{\text{ovo}}_{\text{MMD}}$	Euclidean	0.048	0.044
	SIMCLR	0.145	0.145
$\mathcal{I}^{\text{ova}}_{\mathcal{W}}$	Euclidean	0.043	0.037
	SIMCLR	0.079	0.052
$\mathcal{I}^{\text{ovo}}_{\mathcal{W}}$	Euclidean	0.041	0.036
	SIMCLR	0.138	0.080

Competitor	ARI
K-means on raw images	0.041
K-means on SIMCLR features	0.147
IDFD (Tao et al., 2021)	0.060
CC (Y. Li et al., 2021)	0.030
JULE (J. Yang, Parikh, & Batra, 2016)	0.138

number of hidden layers from 1 to 3. All models were run 30 times. We perform clustering by maximising the OvO Wasserstein during 1000 epochs with a learning rate of 2×10^{-3} for the Adam optimiser. Indeed, with a long training time, we allow the model to find more clusters as seen in Sec 3.5.6. We also experimented using MI with the same models and hyperparameters. Finally, we selected the final model using the highest GEMINI value for each different depth of GCN.

Similarly, we ran 30 times two competitors with a number of clusters to find ranging from 2 to 10: the LPM model (Hoff, Raftery, & Handcock, 2002) and the Deep LPM model (Liang et al., 2022) which are generative methods based on the assumption of a latent position of the nodes in the graph determining their interaction. Their respective best models were selected according to the lowest Bayesian information criterion and the highest evidence lower bound. For Deep LPM, we used the architecture proposed by Liang et al. (2022) with a one-hidden-layer network.

We start by showing the highest Wasserstein-GEMINI clustering in Figure 3.11 where the graph nodes are positioned according to the Fruchterman Reingold algorithm (Fruchterman & Reingold, 1991).

When we look at the interaction matrices of the best models in Figure 3.12, we can observe that GEMINI clustering applied to graph yields dense clusters where nodes intensively connect with each other (Figure 3.12a). The LPM model found fewer clusters that are as well densely connected, while the Deep LPM found 6 clusters including one densely connected and another one which contains multiple nodes sparsely connected. Interestingly, GEMINI managed to isolate

Table 3.5 – Average ARI scores $_{\text{std}}$ between 30 GEMINI-trained models and the best Deep LPM and LPM clustering according to ELBO criterion. H represents the number of hidden layers in the model trained with GEMINI.

GEMINI	H	LPM	Deep LPM
MI	1	$0.47_{0.08}$	$\mathbf{0.31_{0.10}}$
	2	$0.31_{0.07}$	$0.17_{0.09}$
	3	$0.29_{0.04}$	$0.16_{0.05}$
OvO Wasserstein	1	$\mathbf{0.58_{0.04}}$	$0.25_{0.03}$
	2	$0.54_{0.06}$	$0.22_{0.03}$
	3	$0.49_{0.06}$	$0.21_{0.04}$

graph nodes that act as hubs, *e.g.* the 9th cluster in Fig. 3.12a with star-shaped interactions while Deep LPM mixes hubs in its 3rd cluster in Figure 3.12c. A potential explanation for this difference is that LPM models seek to cluster nodes based on supposedly close latent representations whereas GEMINI clustering uses the Wasserstein distance, hence taking into account the flow of information passing through each graph node. In Table 3.5, we compared the average ARI between our 30 GEMINI models with the best-selected LPM and Deep LPM clustering. It appears indeed that GEMINI models have a stronger ARI with the LPM model than with the Deep LPM, as the former concentrated on dense clusters. Interestingly, we can notice in Table 3.5 that MI models with one hidden layer are able to produce satisfying results with an ARI close to or better than the ARI between GEMINI and LPM methods. However, when the models turn deeper, the performances of MI drop whereas the ARI between the Wasserstein-GEMINI decreases more slowly.

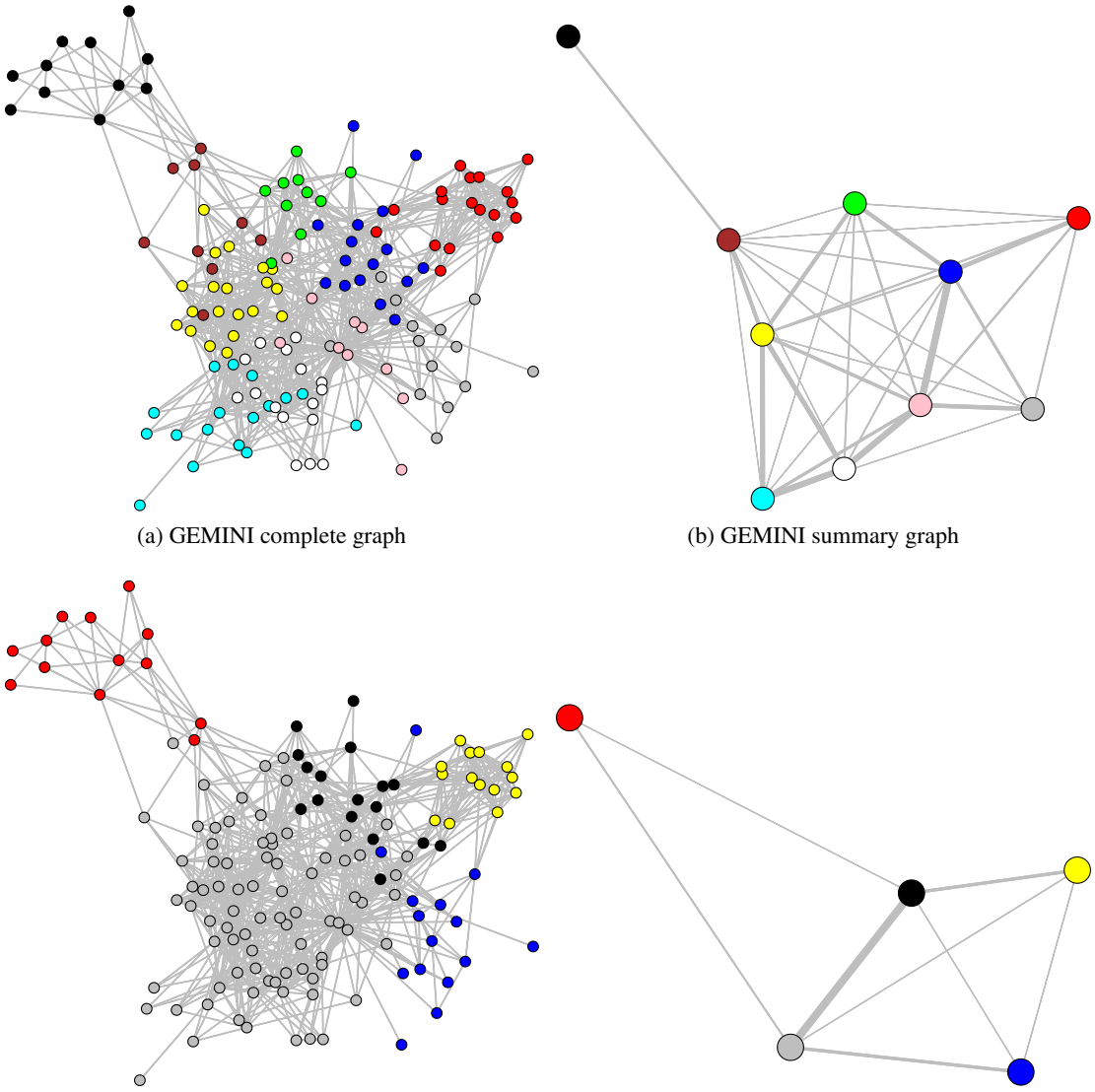
To further compare the performances of these models, we chose to evaluate their proportions of ambiguous clusters (PAC score) (Şenbabaoğlu, Michailidis, & Li, 2014). The PAC score corresponds to the proportion of pairs of samples that have been ambiguously clustered together between 10% and 90% of the runs. Through Figure 3.13, we can see that the models trained with Wasserstein-GEMINI have a lower PAC score compared to MI-trained models which highlights a more consistent clustering assignment through all 30 runs. We can further observe that for both objective functions, a deeper model leads to a higher PAC score that we can explain by the difficulty of repeating the same decision boundary with deep models. Eventually, we may conclude that although MI may have competitive results with shallow models compared to GEMINI and Deep LPM, it is unable to repeat a consistent clustering as illustrated by a high PAC score.

3.6 Conclusion

We highlighted that the choice of distance at the core of MI can alter the performances of deep learning models when used as an objective for deep discriminative clustering. We first showed that MI maximisation does not necessarily reflect the best decision boundary in clustering when the clustering model converges to a Dirac distribution. We introduced GEMINI, a method which only needs the specification of a neural network and a kernel or distance in the data space. Moreover, we showed how the notion of neighbourhood built by the neural network can affect the clustering, especially for MI. To the best of our knowledge, this is the first method that trains single-stage neural networks from scratch using neither data augmentations nor regularisations, yet achieving

good clustering performances. We emphasised that GEMINIs are only searching for a maximum number of clusters: after convergence, some may be empty. Finally, we introduced several versions of GEMINIs and would encourage OvA MMD or OvA Wasserstein as a default choice, since it proves to both incorporate knowledge from the data using a kernel or distance while remaining less complex than OvO MMD and OvO Wasserstein in time and memory. OvO versions could be privileged for fine-tuning steps. Future works could focus on the joint learning of distances or kernels (Wu, Khan, Ioannidis, & Dy, 2020) while maximising the GEMINI to get both meaningful clustering and metrics in the data space. We will also investigate why after convergence some clusters end up empty.

GEMINI constitutes therefore a promising clustering algorithm for the PROGRESSA dataset. However, obtaining the clusters is the first cornerstone of clustering. There still remains to analyse the contents of the clusters. To alleviate this interpretation, we now turn to Chapter 4 where we introduce feature selection along clustering with Sparse GEMINI.



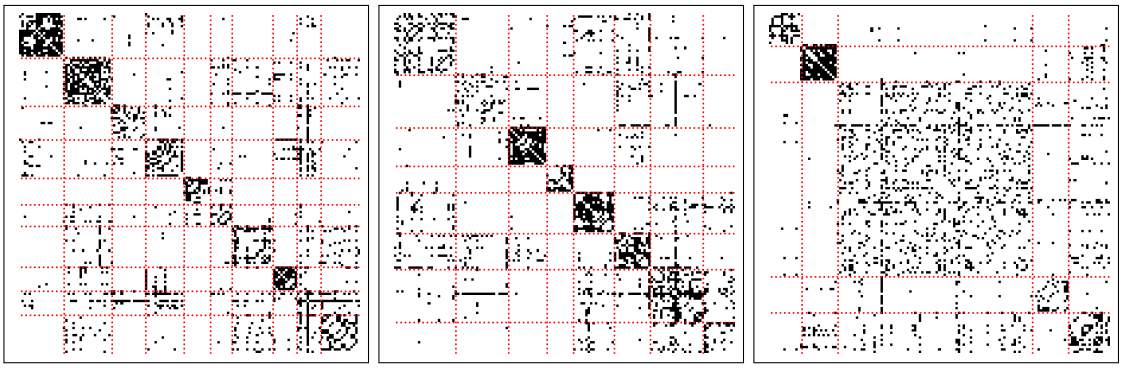
(a) GEMINI complete graph

(b) GEMINI summary graph

(c) Deep LPM complete graph

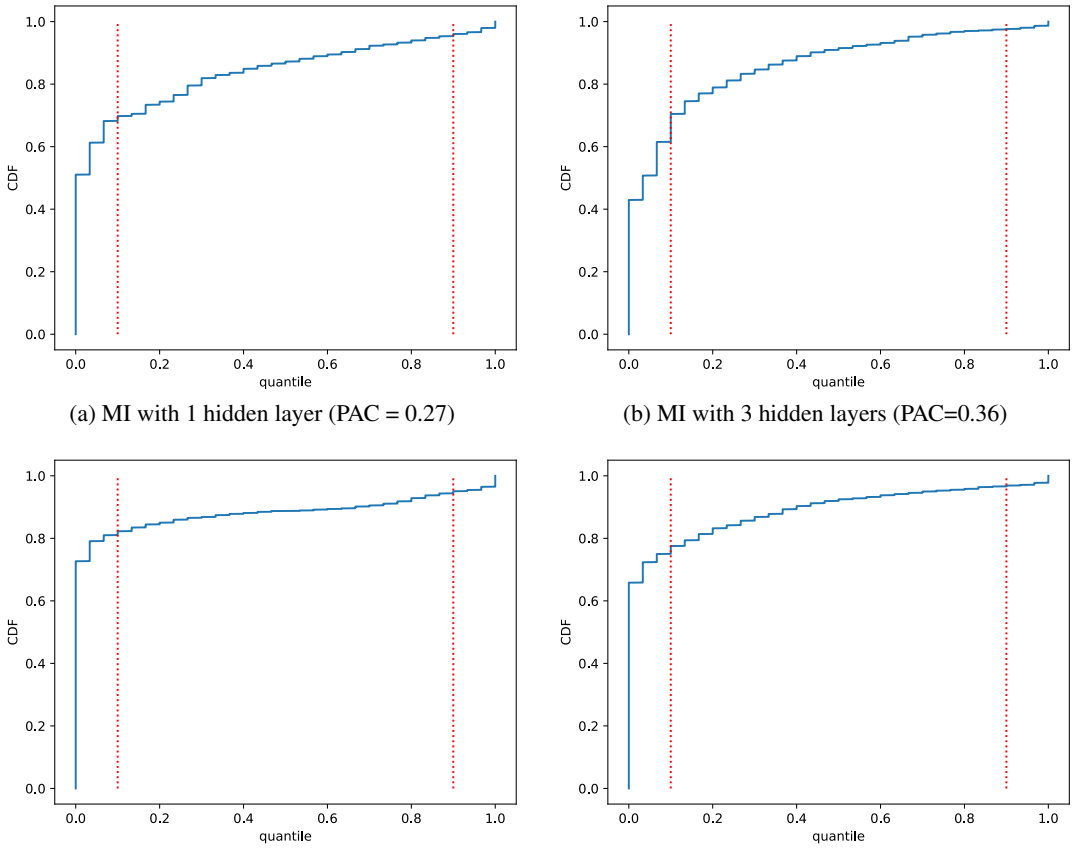
(d) Deep LPM summary graph

Figure 3.11 – Fruchterman Reingold representation of the Enron email interaction graph. Nodes are coloured according to clusters. The summary graph is a cluster-wise average of the positions of the nodes with edges as strong as the number of interactions between two clusters.



(a) GEMINI clustering (b) LPM clustering (c) Deep LPM clustering

Figure 3.12 – Interaction matrices: each black cell encodes an edge. Nodes were reorganised according to their clustering for each model.



(a) MI with 1 hidden layer (PAC = 0.27) (b) MI with 3 hidden layers (PAC=0.36)

(c) GEMINI with 1 hidden layer (PAC=0.13) (d) GEMINI with 3 hidden layers (PAC=0.22)

Figure 3.13 – CDFs of consensus between all clusterings produced by the models on the Enron Email dataset. The indicated PAC score is the difference of cdf between the quantiles 10% and 90%. A lower PAC score is better and highlights more consistent clustering assignments.

CHAPTER 4

Sparsifying the generalised mutual information for joint feature selection and discriminative clustering

5.1 Introduction . . . 79
5.2 Training trees . . . 80
5.2.1 How do we train supervised trees? . . . 81
5.2.2 How do we train unsupervised trees? . . . 81
5.2.3 Related motivating example . . . 82
5.3 Kauri: K-means as unsupervised reward ideal . . . 82
5.3.1 Notations and modelling . . . 83
5.3.2 Objective . . . 83
5.3.2.1 Equivalence to OvA MMD-GEMINI . . . 85
5.3.2.2 Equivalence to OvO MMD-GEMINI . . . 85
5.3.3 Tree branching . . . 87
5.3.4 Gain metrics . . . 87
5.3.4.1 General expression of a gain . . . 87
5.3.4.2 Creating a new cluster: the *star gain* . . . 88
5.3.4.3 Creating two clusters: the *double star gain* . . . 89
5.3.4.4 Merging with another cluster: the *switch gain* . . . 89
5.3.4.5 Reallocating content to different clusters: the *reallocation gain* 90
5.4 A fast implementation for Kauri . . . 91
5.4.1 Pre-computing kernel stocks . . . 91
5.4.2 Optimising split evaluation . . . 91
5.4.3 An iterative rule for split stocks . . . 93
5.4.4 Complete picture . . . 94
5.4.5 Implementation in GemClus . . . 94
5.5 Douglas: DNDTs optimised using GEMINI leverage apprised splits . . . 97

5.5.1 The Douglas model 97
5.5.2 Implementation in GemClus 98
5.6 Experiments 98
5.6.1 Performances with as many leaves as clusters 100
5.6.2 Performances with more leaves than clusters 101
5.6.3 Varying the kernel 108
5.6.4 Pure numpy Douglas performances 108
5.6.5 A qualitative example of the obtained decision tree 109
5.7 Conclusion 110

4.1 Introduction

It is common that clustering algorithms and supervised models rely on all available features for the best performance. However, as datasets become high-dimensional, clustering algorithms tend to break under the curse of dimensionality (Bouveyron & Brunet-Saumard, 2014b), for instance in biological micro-array data where the number of variables outweighs the number of samples (McLachlan et al., 2002). To alleviate this burden, feature selection is a method of choice. Indeed, all features may not always be of interest: some variables can be perceived as relevant or not with respect to the clustering objective. Relevant variables bring information that is useful for the clustering operation, while irrelevant variables do not bring any new knowledge regarding the cluster distribution (Tadesse et al., 2005) and redundant variables look relevant but do not bring beneficial knowledge (Maugis et al., 2009). The challenge of selecting the relevant variables often comes with the burden of combinatorial search in the variable space. Therefore, solutions may be hardly scalable to high-dimensional data (Raftery & Dean, 2006) or to the number of samples (Witten & Tibshirani, 2010) when the selection process is part of the model. Therefore reducing the number of variables for learning to a relevant few is of interest, notably in terms of interpretation (Fop & Murphy, 2018). The necessity of variable selection notably met successful applications in genomics (Marbac et al., 2020), multi-omics (Meng, Helm, Frejno, & Kuster, 2016 ; Ramazzotti, Lal, Wang, Batzoglou, & Sidow, 2018 ; R. Shen et al., 2012).

Often, integrating the selection process as part of the model will lead to either not scaling well (Solorio-Fernández et al., 2020) in terms of number of features (Raftery & Dean, 2006) or number of samples (Witten & Tibshirani, 2010) or imposing too constrained decision boundaries due to the nature of strong parametric assumptions. To alleviate both problems, we present the Sparse GEMINI: a model that combines the logistic regression or the LassoNet architecture (Lemhadri, Ruan, Abraham, & Tibshirani, 2021) and the discriminative clustering objective GEMINI from Chapter 3 for a scalable discriminative clustering with penalised feature selection.

We start by presenting the different choices of clustering distributions $p_\theta(y|\boldsymbol{x})$ for Sparse GEMINI: logistic regression and LassoNet as summarised in Figure 4.1 with their associated objectives. We then detail how the feature elimination is ensured during training and proceed to experiments on both synthetic and real-world datasets.

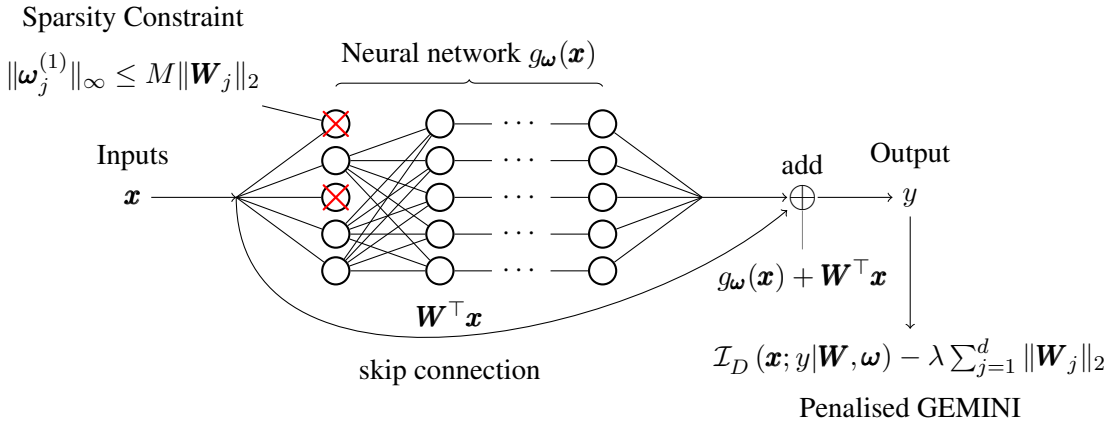


Figure 4.1 – Description of the complete Sparse GEMINI model. Through a proximal gradient, clusters learned by GEMINI drop irrelevant features both in a skip connection and an MLP. Setting $M = 0$ recovers a sparse unsupervised logistic regression.

4.2 Sparse GEMINI

4.2.1 Unsupervised logistic regression architecture

We start with the simplest discriminative model for variable selection: logistic regression. This corresponds to the case where our clustering distribution $p_\theta(y|\boldsymbol{x})$ is characterised by the underlying architecture:

$$\psi_\theta(\boldsymbol{x}) = \text{Softmax}(\theta^\top \boldsymbol{x}), \tag{4.1}$$

with $\theta \in \mathbb{R}^{d \times K}$ for d features and K clusters. Notice the absence of bias as this linear model is a sub-case of the neural network model covered in the next section. To properly ensure that vector weights are eliminated at once, a group-lasso penalty is preferred (Hastie, Tibshirani, & Wainwright, 2015, Section 4.3) also known as ℓ_1/ℓ_2 penalty (F. Bach, Jenatton, Mairal, & Obozinski, 2012). We consider a user-defined partition of the input features into $G \leq d$ groups, each with associated parameter subset θ_j. Note that the dimensions of θ_j vary depending on the number of features within the j-th group. For example, a categorical variable taking M values transformed into a one-hot-encoded vector of dimension M can be associated to a single group. Thus, the optimal parameters should satisfy:

$$\hat{\theta} = \arg\max_\theta \mathcal{I}_D(\boldsymbol{x}; y|\theta) - \alpha \sum_{j=1}^G \|\theta_j\|_2. \tag{4.2}$$

This is exactly the same objective formulation as the supervised multi-class Lasso if we replace GEMINI by the likelihood or any other supervised loss. Notice that α is positive because we seek to simultaneously maximise GEMINI and minimise the ℓ_1/ℓ_2 penalty. During training, the sparse linear parameter will progressively remove variables by setting all grouped parameters to 0. If we set $G = d$, then each variable can be removed on its own as there are no groups of variables. A similar objective without group-lasso and using standard mutual information can be found in (Kong et al., 2015, Eq. (4)), although specific initialisation strategies were required to circumvent the unspecificity of mutual information local maxima.

4.2.2 The LassoNet architecture

We extend this procedure to neural network by adapting the LassoNet (Lemhadri et al., 2021) framework with GEMINIs. The neural network $\psi_\theta : \mathcal{X} \mapsto \mathbb{R}^K$ consists of one multi-layered perceptron (MLP) and a linear skip connection:

$$\psi_\theta(\boldsymbol{x}) = \text{Softmax}\left(g_{\boldsymbol{\omega}}(\boldsymbol{x}) + \boldsymbol{W}^\top \boldsymbol{x}\right), \tag{4.3}$$

with θ containing $\boldsymbol{\omega}$ the parameters of the MLP $g_{\boldsymbol{\omega}}$ and $\boldsymbol{W} \in \mathbb{R}^{K\times d}$ the weights of a linear skip connection penalised by group-lasso. This leads to the same optimisation objective as previously with a focus on the skip connection parameters:

$$\hat{\theta} = \arg\max_\theta \mathcal{I}_D\left(\boldsymbol{x}; y|\theta\right) - \alpha \sum_{j=1}^{G} \|\boldsymbol{W}_j\|_2, \tag{4.4}$$

with $\boldsymbol{W}_j$, the weights of the j-th group of features from $\boldsymbol{W}$. As the sparse skip connection $\boldsymbol{W}$ loses some feature subset we must force the MLP g_ω to drop this same subset of features as well. Therefore the weights of the first layer $\boldsymbol{\omega}^{(1)}$ are constrained such that:

$$\|\boldsymbol{\omega}_j^{(1)}\|_\infty \leq M\|\boldsymbol{W}_j\|_2, \forall j \leq G. \tag{4.5}$$

where M is called the hierarchy coefficient. Thus, when the j-th set of features is eliminated, all weights starting from these features in the MLP will be equal to 0 as well. When $M = 0$, the method is equivalent to the penalised logistic regression from the previous section because all entry weights of the MLP are equal to zero, hence passing no information.

Interestingly, in addition to the constraints that are designed to specifically select features, dimension reduction can be performed as well by extracting representations from lower-dimension layers in the network $g_{\boldsymbol{\omega}}$. However, this intermediate representation would not be complete as it misses the information from the skip connection.

4.3 Optimisation

4.3.1 Training and model selection

We follow Lemhadri et al. (2021) in proposing a *dense-to-sparse* training strategy for the penalty coefficient. Training is carried along a path where the ℓ_1 penalty parameter α is geometrically increased:

$$\alpha = \alpha_0 \rho^t, \tag{4.6}$$

with $\rho > 1$ at time step t after an initial step without ℓ_1 penalty. We stop when the number of remaining features used by the model is below a user-defined threshold $0 < d_{\text{thres}} < d$ which can be thought of as the minimum number of useful variables required. Each time the number of features decreases during training, we save its associate intermediate model

Once the training is finished, we look again at all GEMINI scores during the feature decrease and select the model with the minimum of features that managed to remain in the arbitrary range of 90% of the best GEMINI value. This best value is most of the time the loss evaluated with the model exploiting all features.

Interestingly, as MMD-GEMINI is equivalent to a kernel K-means objective, as shown by França et al. (2020), a subset of the method resembles to a sparse *kernel* K-means. However, unlike related works (França et al., 2020 ; Witten & Tibshirani, 2010), the selection process is done through gradient descent, *i.e.* without pre-selecting variables, and without the definition of explicit centroids, thus being less strict regarding the number of clusters to find.

4.3.2 Extension proposal: the dynamic training regime

As features get eliminated during the training, the notion of affinity (distance c or kernel κ) and clustering with respect to GEMINI between two samples changes. Indeed, GEMINI aims at maximising a distance between two related distributions using an affinity computed between samples, yet removing features from the inference implies we do not cluster any longer the same original data space, but rather a subspace at step t: $\mathcal{X}_t = \prod_{j \in I_t} \mathcal{X}_j$, where I_t is the set of remaining groups of features. If we still compute our affinity function using all features from $\mathcal{X}$, the extra removed features may bring noise compared to the affinity between the relevant features, and thus bring confusion with regards to the ideal decision boundary.

To respect the original notion of GEMINI in clustering, we introduce the dynamic training regime, where at each time step t, the affinity function is computed using only the subset of relevant group of features I_t. We call *static* regime the training with usage of all features in the affinity function as described in section 4.2. The advantage of the dynamic training regime is that it respects the notion of GEMINI with regard to the decision boundary, while the static regime yields comparable values of GEMINI independently of the number of selected features. However, the dynamic regime is incompatible with the selection process described in section 4.3.1 because any change of data space implies a change of values for kernels or distances and thus for GEMINI, making models incomparable. Moreover, we may have more theoretical guarantees of convergence for the usual static regime than in the dynamic regime, which may seem unstable.

4.3.3 Gradient considerations

To ensure the convergence of the parameters to 0 upon elimination, we adopt a proximal gradient strategy (Hastie et al., 2015, Chapter 5). In the case of sparse logistic regression, the gradient *ascent* hence follows the two classical steps:

$$\beta = \theta^t + \eta^t \nabla_\theta \mathcal{I}_D \left(\boldsymbol{x}; y | \theta \right), \tag{4.7}$$

$$\theta_j^{t+1} = \mathcal{S}_{\rho^t \alpha \|\beta_j\|_2}(\beta_j), \forall j \leq G, \tag{4.8}$$

where η is the learning rate at timestep t. The soft-thresholding operation $\mathcal{S}_\alpha$:

$$\mathcal{S}_\alpha(x) = \text{sign}(x) \max\{0, |x| - \alpha\}, \tag{4.9}$$

is the closed-form solution of the proximal operator to project the parameters on the constrained space due to the group-lasso penalty (Hastie et al., 2015, Section 5.3.3). Consequently, we are sure to obtain true zeroes in the linear weights of the logistic regression or the weights of the skip connection for the neural network. For the case of the complete neural network model, Lemhadri et al. (2021) gracefully provide a proximal gradient operation to satisfy inequality constraints during training time which guarantees true zeros in the first MLP layer as well.

```python
from gemclus.sparse import SparseLinearMMD
from sklearn.datasets import load_breast_cancer
# load data
X, _ = load_breast_cancer(return_X_y=True)
# Create a simple sparse logistic regression model
model = SparseLinearMMD(n_clusters=2, alpha=1)
# Perform the path for eliminating the variables
path_results = model.path(X)
# Conclude with clustering
y_pred = model.predict(X)
```

Listing 4.1 – An example of *gemclus* doing clustering with variable selection with the `path` method.

Table 4.1 – Brief description of datasets involved in experiments

Name	Samples	Features	#Classes
US-Congress	435	16	2
Heart-statlog	270	13	2
MNIST	12000	784	10
MNIST-BR	12000	784	10
Prostate-BCR	171	25904	2

4.3.4 Implementation in GemClus

The Sparse GEMINI method is implemented in GemClus. The available models comprises logistic regression and LassoNet models with any GEMINI, including both static and dynamic training regimes. For instance, Listing 4.1 gives an example of a path performed with the unsupervised LASSO model for MMD-GEMINI. The code of this chapter was written prior to GemClus with PyTorch's automatic differentiation.

In the specific case of the logistic regression, we added a bias to the model from Eq. (4.1) in the GemClus implementation. However, that bias is not regularised by the ℓ_1 penalty.

4.4 Experiments

A brief summary of the datasets used in these experiments can be found in table 4.1.

4.4.1 Metrics

Depending on the experiments for comparison purposes, we report 3 different metrics. The adjusted rand index (ARI, Hubert & Arabie, 1985) describes how close the clustering is to the classes, with a correction to random guesses. The variable selection error rate (VSER), for instance used by Celeux, Martin-Magniette, Maugis-Rabusseau, et Raftery (2014), describes the percentage of variables that the model erroneously omitted or accepted, therefore the lower the better. We

```
from gemclus.sparse import SparseMLPMMD
from gemclus.data import celeux_one

# Generate the data according to the 5th scenario
X,y = celeux_one(n=300, p=95, mu=1.7)
# Prepare the model: MLP with OvA MMD-GEMINI for 3 clusters
model = SparseMLPMMD(n_clusters=3)
# Progressively increase the penalty until all features are removed
# res contains the history of feature selection and best model weights
res = model.path(X)
```

Listing 4.2 – An example of sparse GEMINI model fitting the 5th scenario of the synthetic datasets

finally report the correct variable rate (CVR) which describes how many of the expected variables were selected: higher is better. For example, a model selecting all variables of a dataset with d variables and d' good variables will get a CVR of 100% and a VSER of $1 - \frac{d'}{d}$.

4.4.2 Default hyperparameters

We set the hierarchy coefficient to $M = 10$, as Lemhadri et al. (2021) report that this value seems to "work well for a variety of datasets". We also report the performances for the logistic regression mode when $M = 0$. The optimiser for the initial training step with $\alpha = 0$ is Adam (Kingma & Ba, 2014) with a learning rate of 10^{-3} while other steps are done with SGD with momentum 0.9 and the same learning rate. Most of our experiments are done with 100 epochs per step with early stopping as soon as the global objective does not improve by 1% for 10 consecutive epochs. The early stopping criterion is evaluated on the same training set since we do not seek to separate the dataset in train and validation sets in clustering. All activation functions are ReLUs. The default starting penalty is $\alpha_0 = 1$ with a 5% increase per step. We keep the linear kernel and the Euclidean distance respectively in conjunction with the MMD and Wasserstein distances when evaluating GEMINI. Finally, we evaluate in most experiments the method with the exact same number of clusters as the number of known (supervised) labels.

4.4.3 Numerical experiments

We experimented Sparse GEMINI on two synthetic datasets proposed by Celeux et al. (2014) and also used by Bouveyron et Brunet-Saumard (2014a) to first highlight some properties of the algorithm and compare it with competitors.

The first synthetic dataset consists of a few informative variables amidst noisy independent variables. The first 5 variables are informative and drawn from an equiprobable multivariate Gaussian mixture distribution of 3 components. All covariances are set to the identity matrix. The means are $\boldsymbol{\mu}_1 = -\boldsymbol{\mu}_2 = \alpha\mathbf{1}$ and $\boldsymbol{\mu}_3 = \mathbf{0}$. All remaining p variables follow independent noisy centred Gaussian distributions. The number of samples n, the mean proximity α and the number of non-informative variables p vary over 5 scenarios. For the 2 first scenarios, we use $n = 30$ samples

and $n = 300$ for others. The scenarios 1 and 3 present the challenge of close Gaussian distributions with $\alpha = 0.6$ while others use $\alpha = 1.7$. Finally, we add $p = 20$ noisy variables, except for the fifth scenario which takes up to $p = 95$ uninformative variables. This experiment can typically be done by running the algorithm from Listing 4.2 using *gemclus*.

The second dataset consists of $n = 2000$ samples of 14 variables, 2 of them being informative and most others being linearly dependent on the former. The Gaussian mixture is equiprobable with 4 Gaussian distributions of means $[0, 0]$, $[4, 0]$, $[0, 2]$ and $[4, 2]$ with identity covariances. The 9 following variables are sampled as follows:

$$\boldsymbol{x}^{3-11} = [0, 0, 0.4, 0.8, 1.2, 1.6, 2.0, 2.4, 2.8]^\top + \boldsymbol{x}^{1-2\top} \begin{bmatrix} 0.5 & 2 & 0 & -1 & 2 & 0.5 & 4 & 3 & 2 \\ 1 & 0 & 3 & 2 & -4 & 0 & 0.5 & 0 & 1 \end{bmatrix} + \boldsymbol{\epsilon}, \quad (4.10)$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega})$ with the covariance:

$$\boldsymbol{\Omega} = \text{diag}\left(\boldsymbol{I}_3, 0.5\boldsymbol{I}_2, \text{diag}([1, 3])\text{Rot}(\frac{\pi}{3}), \text{diag}[2, 6]\text{Rot}(\frac{\pi}{6})\right). \quad (4.11)$$

Finally, the last 3 variables are independently sampled from $\mathcal{N}([3.2, 3.6, 4], \boldsymbol{I}_3)$.

For all synthetic datasets, we asked training to stop with d_{thres} set to the expected quantity of variables. We report the results of Sparse GEMINI in Table 4.2 after 20 runs. We compare our results against our own runs of other methods using their R package: Sparse K-means (Witten et al., 2013), ClustVarSel (Scrucca & Raftery, 2018), vscc (Andrews & McNicholas, 2014) and SparseFisherEM (Bouveyron & Brunet, 2012). Due to the lack of space, we only report the scores for OvO GEMINI in Table 4.2. The results for OvA GEMINI can be found separately in Table 4.3.

It appears that Sparse GEMINI is efficient in selecting relevant variables when several others are noisy, especially with OvO MMD-GEMINI while maintaining a high ARI. Moreover, while we do not systematically get the best ARI, our performances never fall far behind the most competitive method. We can also observe that MMD-GEMINI learns well despite the presence of few samples in scenarios 2 and 3 and that the usage of an MLP leads to a trade-off between ARI and VSER when we have enough samples. Additionally, the selection strategy often leads to selecting the correct number of variables for MMD-GEMINI, except in scenarios 1 and 3 where the Gaussian distributions are close to each other, which is hard given the large variance. For Wasserstein-GEMINI, we notice that the performances in selection are improved with the presence of more samples. However, the clustering performances are worse than MMD-GEMINI's, which we can attribute to the contribution of the noisy variables to the computation of the distances between samples, thus troubling the holistic perspective of the Wasserstein distance on the cluster distribution. It also appears that we performed poorly at selecting the correct variables in the presence of redundancy in the second dataset. However, since all variables except 3 are correlated to the informative variables, we still managed to get a correct ARI on the dataset while using other variables. On average, the variables selected by our models were the 6th and the 8th variables. We focus on this difference of convergence in Figure 4.2 where we plot the norm of the skip connection per feature $\boldsymbol{W}_j$. In the case of noisy variables, we were able to recover them as the number of selected features decreased, whereas we eliminated the informative variable of the second dataset during the first steps. In general, Clustvarsel (Scrucca & Raftery, 2018) performed better on this type of synthetic dataset in terms of variable selection because it explicitly assumes a linear dependency between relevant variables and others.

Table 4.2 – Performances of Sparse GEMINI (OvO only) on synthetic datasets after 20 runs. We compare our performances against other methods. SX stands for a scenario of the first synthetic dataset and D2 stands for the second synthetic dataset. Standard deviation is reported in subscript.

(a) ARI scores (greater is better)

	Sparse K-means	Clust VarSel	VSCC	SFEM	MMD Logistic	MMD MLP	Wasserstein Logistic	Wasserstein MLP
S1	$0.09_{0.08}$	$0.05_{0.07}$	$0.00_{0.02}$	$\mathbf{0.13_{0.11}}$	$\mathbf{0.14_{0.11}}$	$0.08_{0.07}$	$0.09_{0.12}$	$0.04_{0.07}$
S2	$\mathbf{0.80_{0.15}}$	$0.20_{0.22}$	$0.02_{0.03}$	$0.72_{0.17}$	$0.59_{0.17}$	$0.52_{0.23}$	$0.44_{0.18}$	$0.43_{0.15}$
S3	$0.11_{0.03}$	$0.04_{0.10}$	$0.15_{0.10}$	$\mathbf{0.22_{0.05}}$	$\mathbf{0.21_{0.05}}$	$\mathbf{0.21_{0.04}}$	$0.14_{0.05}$	$0.10_{0.05}$
S4	$\mathbf{0.87_{0.04}}$	$\mathbf{0.88_{0.04}}$	$0.84_{0.12}$	$\mathbf{0.87_{0.04}}$	$0.74_{0.07}$	$\mathbf{0.86_{0.05}}$	$0.73_{0.19}$	$0.82_{0.11}$
S5	$\mathbf{0.87_{0.03}}$	$0.65_{0.38}$	$0.00_{0.00}$	$0.83_{0.03}$	$0.76_{0.05}$	$\mathbf{0.86_{0.03}}$	$0.59_{0.20}$	$0.67_{0.20}$
D2	$0.31_{0.03}$	$\mathbf{0.60_{0.02}}$	$0.58_{0.02}$	$0.58_{0.01}$	$0.57_{0.01}$	$0.54_{0.03}$	$0.57_{0.01}$	$0.55_{0.02}$

(b) VSER scores (lower is better)

	Sparse K-means	Clust VarSel	VSCC	SFEM	MMD Logistic	MMD MLP	Wasserstein Logistic	Wasserstein MLP
S1	$0.31_{0.22}$	$0.28_{0.06}$	$0.72_{0.15}$	$\mathbf{0.24_{0.07}}$	$0.44_{0.15}$	$0.43_{0.11}$	$0.51_{0.11}$	$0.56_{0.13}$
S2	$0.75_{0.18}$	$0.29_{0.07}$	$0.73_{0.09}$	$0.27_{0.08}$	$\mathbf{0.06_{0.05}}$	$0.11_{0.07}$	$0.15_{0.10}$	$0.24_{0.12}$
S3	$0.47_{0.34}$	$0.25_{0.07}$	$0.65_{0.22}$	$0.20_{0.04}$	$\mathbf{0.07_{0.05}}$	$0.20_{0.11}$	$0.20_{0.12}$	$0.56_{0.14}$
S4	$0.80_{0.00}$	$\mathbf{0.01_{0.03}}$	$0.64_{0.29}$	$0.23_{0.08}$	$\mathbf{0.00_{0.00}}$	$\mathbf{0.00_{0.00}}$	$\mathbf{0.01_{0.03}}$	$\mathbf{0.00_{0.02}}$
S5	$0.95_{0.00}$	$0.05_{0.03}$	$0.95_{0.00}$	$0.10_{0.02}$	$\mathbf{0.00_{0.00}}$	$\mathbf{0.00_{0.00}}$	$\mathbf{0.01_{0.01}}$	$\mathbf{0.01_{0.01}}$
D2	$0.84_{0.06}$	$0.00_{0.00}$	$0.74_{0.13}$	$0.52_{0.08}$	$0.29_{0.00}$	$0.31_{0.04}$	$0.29_{0.00}$	$0.29_{0.00}$

(c) CVR scores (greater is better)

	Sparse K-means	Clust VarSel	VSCC	SFEM	MMD Logistic	MMD MLP	Wasserstein Logistic	Wasserstein MLP
S1	$0.53_{0.31}$	$0.11_{0.10}$	$\mathbf{0.87_{0.23}}$	$0.28_{0.18}$	$0.60_{0.26}$	$0.64_{0.19}$	$0.63_{0.23}$	$0.59_{0.17}$
S2	$\mathbf{1.00_{0.00}}$	$0.14_{0.17}$	$0.66_{0.39}$	$0.39_{0.17}$	$0.93_{0.10}$	$0.82_{0.22}$	$0.83_{0.13}$	$0.78_{0.14}$
S3	$\mathbf{1.00_{0.00}}$	$0.19_{0.30}$	$0.97_{0.13}$	$0.27_{0.15}$	$0.95_{0.09}$	$0.99_{0.04}$	$0.68_{0.25}$	$0.92_{0.12}$
S4	$\mathbf{1.00_{0.00}}$	$0.19_{0.30}$	$0.97_{0.13}$	$0.27_{0.15}$	$1.00_{0.00}$	$1.00_{0.00}$	$0.99_{0.04}$	$0.99_{0.04}$
S5	$\mathbf{1.00_{0.00}}$	$0.75_{0.44}$	$\mathbf{1.00_{0.00}}$	$0.66_{0.18}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$0.94_{0.11}$	$0.96_{0.10}$
D2	$0.98_{0.11}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$0.00_{0.00}$	$0.00_{0.00}$	$0.00_{0.00}$	$0.00_{0.00}$

Table 4.3 – Performances of Sparse GEMINI using the OvA objectives on the synthetic datasets. SX stands for a scenario of the first synthetic dataset and D2 stands for the second synthetic dataset.

(a) ARI scores (greater is better)

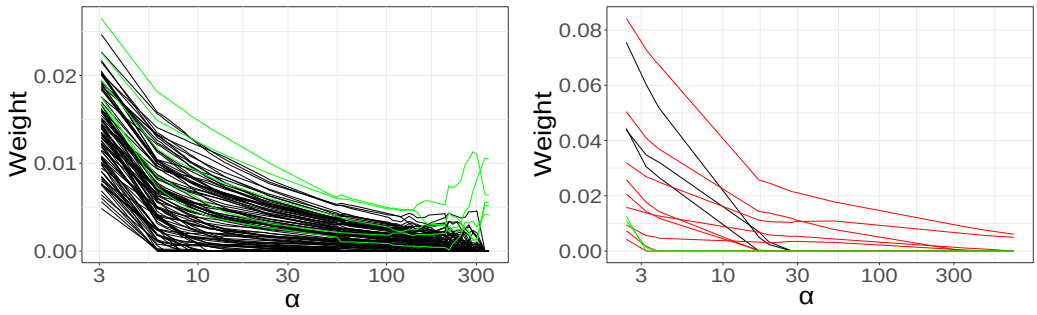
	MMD-GEMINI		Wasserstein-GEMINI	
	Logistic	MLP	Logistic	MLP
S1	$0.08_{0.08}$	$\mathbf{0.11_{0.12}}$	$0.03_{0.05}$	$0.08_{0.09}$
S2	$\mathbf{0.46_{0.11}}$	$\mathbf{0.47_{0.11}}$	$\mathbf{0.46_{0.13}}$	$0.41_{0.15}$
S3	$\mathbf{0.23_{0.07}}$	$\mathbf{0.22_{0.05}}$	$0.13_{0.07}$	$0.08_{0.06}$
S4	$0.43_{0.05}$	$0.45_{0.05}$	$0.56_{0.17}$	$\mathbf{0.74_{0.22}}$
S5	$0.44_{0.05}$	$\mathbf{0.58_{0.11}}$	$0.52_{0.16}$	$0.47_{0.18}$
D2	$0.53_{0.06}$	$0.55_{0.03}$	$\mathbf{0.57_{0.02}}$	$0.54_{0.03}$

(b) VSER scores (lower is better)

	MMD-GEMINI		Wasserstein-GEMINI	
	Logistic	MLP	Logistic	MLP
S1	$\mathbf{0.38_{0.14}}$	$\mathbf{0.37_{0.11}}$	$0.58_{0.13}$	$0.55_{0.10}$
S2	$\mathbf{0.04_{0.04}}$	$0.10_{0.06}$	$0.16_{0.08}$	$0.25_{0.11}$
S3	$\mathbf{0.06_{0.08}}$	$0.19_{0.11}$	$0.20_{0.13}$	$0.68_{0.16}$
S4	$\mathbf{0.00_{0.00}}$	$\mathbf{0.00_{0.00}}$	$0.03_{0.03}$	$\mathbf{0.00_{0.00}}$
S5	$\mathbf{0.00_{0.00}}$	$\mathbf{0.00_{0.00}}$	$0.02_{0.02}$	$0.08_{0.07}$
D2	$0.29_{0.00}$	$0.30_{0.04}$	$0.29_{0.00}$	$0.29_{0.03}$

(c) CVR scores (greater is better)

	MMD-GEMINI		Wasserstein-GEMINI	
	Logistic	MLP	Logistic	MLP
S1	$0.56_{0.20}$	$0.63_{0.31}$	$0.65_{0.21}$	$\mathbf{0.71_{0.17}}$
S2	$\mathbf{0.93_{0.10}}$	$0.84_{0.12}$	$\mathbf{0.91_{0.10}}$	$0.81_{0.12}$
S3	$0.94_{0.16}$	$\mathbf{0.98_{0.06}}$	$0.66_{0.23}$	$\mathbf{0.96_{0.08}}$
S4	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$0.95_{0.09}$	$0.99_{0.04}$
S5	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$0.94_{0.09}$	$0.95_{0.09}$
D2	$0.00_{0.00}$	$0.00_{0.00}$	$0.00_{0.00}$	$0.00_{0.00}$



(a) Dataset 1 Scenario 5

(b) Dataset 2

Figure 4.2 – Example of convergence of the norm of the weights of the skip connection for every feature during training with OvA Wasserstein. Green lines are the informative variables, black lines are the noise and red are the correlated variables. (a) In the case of noisy variables, Sparse GEMINI can recover the informative variables. (b) In the presence of redundant variables, Sparse GEMINI eliminates informative variables to keep the redundant ones.

Table 4.4 – ARI scores on the synthetic datasets with the dynamic regime of training for the Sparse GEMINI using MLPs

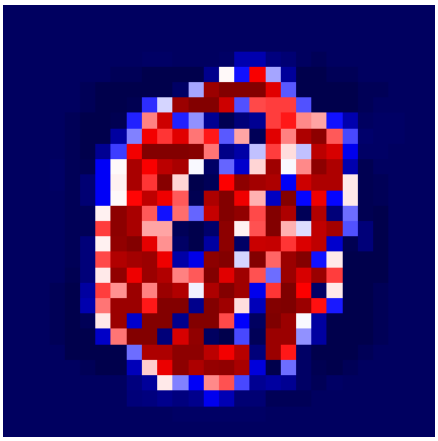
Method	MMD		Wasserstein	
	OvA	OvO	OvA	OvO
Scenario 1	$0.12_{0.13}$	$\mathbf{0.15_{0.12}}$	$0.05_{0.09}$	$0.07_{0.06}$
Scenario 2	$0.48_{0.11}$	$\mathbf{0.63_{0.21}}$	$0.37_{0.11}$	$0.30_{0.13}$
Scenario 3	$\mathbf{0.23_{0.04}}$	$0.20_{0.03}$	$0.11_{0.05}$	$0.11_{0.06}$
Scenario 4	$0.45_{0.07}$	$\mathbf{0.88_{0.03}}$	$0.82_{0.13}$	$0.85_{0.10}$
Scenario 5	$0.62_{0.09}$	$\mathbf{0.84_{0.05}}$	$0.56_{0.20}$	$0.48_{0.17}$
Dataset 2	$0.54_{0.04}$	$0.54_{0.05}$	$0.50_{0.08}$	$\mathbf{0.56_{0.01}}$

4.4.3.1 Specific case of the dynamic training regime

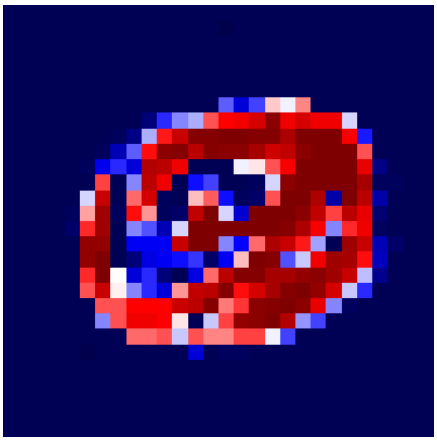
We experimented the dynamic approach on the same synthetic datasets and report the results in Table 4.4. For this experiment, we only evaluated the performances on the final subset of selected features. However, since the Sparse GEMINI is trained until a user-defined number of features is reached, we avoid unfair comparisons with other variable selections methods and do not report the VSER and the CVR. Our main observation on the introduction of the dynamic regime is that it greatly improves the clustering performances of the Wasserstein-GEMINI while not affecting MMD-GEMINI. This success can be explained by the removal of variables as the removal of noise in the distance computation which is crucial for the Wasserstein distance because it takes a global point of view on the complete distribution. In contrast, the MMD only considers expectations, which helps getting rid of noisy variations of the distance around informative variables.

4.4.4 Examples on MNIST and variations

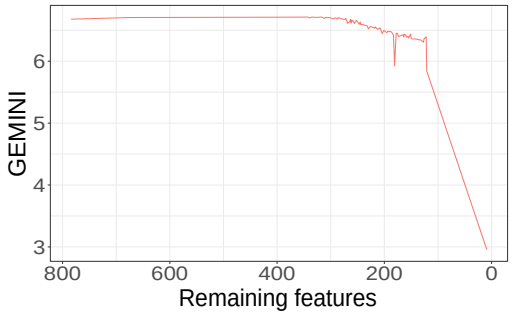
We demonstrate as well performances of the Sparse GEMINI algorithm by running it on the MNIST dataset. The initial α_0 was set to 40. Following Lemhadri et al. (2021), we chose to stop training after finding 50 features. We also use 5% of dropout inside an MLP with 2 hidden layers of 1200 dimensions each (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012), *i.e.* dropping uniformly 5% of the neuron activations. We report in Figure 4.3 the selected features by the clustering algorithms and the evolution of the ARI. We extended this experiment to the variations of MNIST proposed by Larochelle, Erhan, Courville, Bergstra, et Bengio (2007) by showing the performances on the MNIST-BR dataset [*], a challenging dataset for unsupervised variable selection (Mattei, Bouveyron, & Latouche, 2016). This variation consists in samples of MNIST with the black background being replaced by uniform noise hence displaying conditional noise on the data. To be fair, we reduced MNIST to the first 12,000 samples of the training set in order to match the number of samples in MNIST-BR.



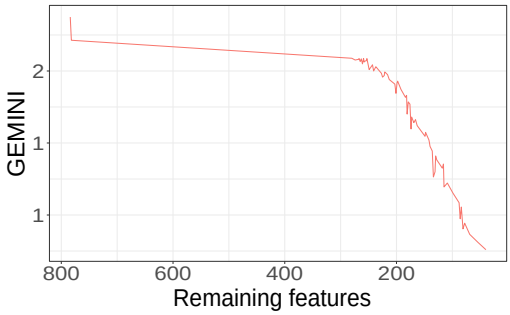
(a) MNIST importance map



(b) MNIST-BR importance map



(c) MNIST GEMINI



(d) MNIST-BR GEMINI

Figure 4.3 – Relative importance of MNIST features after training of Sparse GEMINI with a log-scale color map. Blue features were eliminated at the first steps of α and red features were eliminated last. On the right: evolution of the GEMINI depending on α.

∗. Datasets available at https://web.archive.org/web/20180519112150/http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations

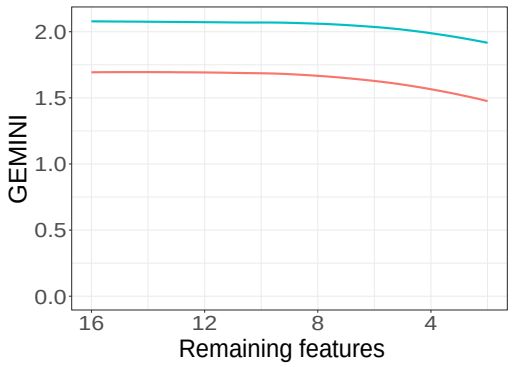
We observed in Figure 4.3 that for both the default MNIST dataset and the MNIST-BR dataset, the feature map concentrates precisely on the good location of the digits in the picture. Following the GEMINI curves in the figures 4.3c and 4.3d despite the presence of noise, the respective selected numbers of features were 122 for MNIST and 243 for MNIST-BR. These chosen models also have a respective ARI of 0.34 for 7 clusters and 0.28 for 8 clusters. The presence of empty clusters is a possible outcome with GEMINI that contributed here to lowering the ARI when evaluating with the true digits targets.

4.4.5 Real datasets

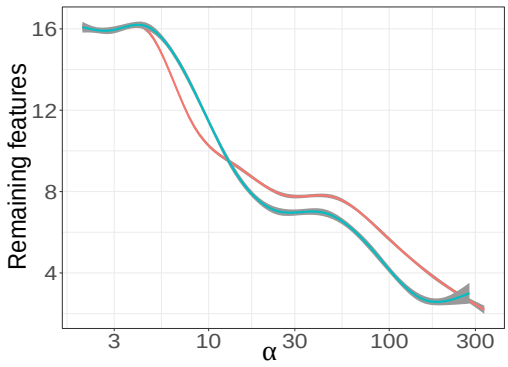
4.4.5.1 OpenML datasets

We ran Sparse GEMINI on two OpenML datasets that are often shown in related works: the US Congress dataset (Quarterly, 1985) and the Heart-statlog dataset (Brown, 2004). The US congress dataset describes the choice of the 435 representatives on 16 key votes in 1984. The labels used for evaluation are the political affiliations: 164 Republican against 267 Democrats. We replaced the missing values with 0 and converted the yes/no answers to 1, -1. Thus, an unknown label is equidistant from both answers. The Heart-statlog dataset describes 13 clinical and heart-related features with labels describing the presence or absence of cardiac disease among patients. We preprocessed it with standard scaling. For the US Congress dataset, we used one hidden layer of 20 nodes and a batch size of 87 samples. For the Heart-statlog dataset, we used 10 nodes and 90 samples. As we seek only two clusters, we only ran the OvA versions of GEMINI because it is strictly equal to the OvO in binary clustering. Both datasets had a penalty increase of $\rho = 10\%$. We first show the number of selected features evolving with α as well as the evolution of the GEMINI score as the number of features decreases respectively in Figure 4.4 for the US Congress dataset and in Figure 4.5 for Heart-statlog. Table 4.5 contains the performances for the both datasets, reporting the average number of selected variables over 20 runs according to our postprocessing selection criterion. We also added the performances of competitors from the previous section. However, we did not manage to run Sparse Fisher EM on the US Congress dataset. For comparison purposes, the best unsupervised accuracy reported on the Heart-statlog dataset by Solorio-Fernández et al. (2020) is 75.3% while Sparse GEMINI achieves 79% with the MMD. The best score for all methods in the review (Solorio-Fernández et al., 2020) is 79.6%, but this encompasses filter methods, which Sparse GEMINI is not. We also get similar results to the best performances of Marbac et al. (2020) who report 33% of ARI. Since most competitors retained all variables in the dataset, we chose to show as well the clustering performances without selection and hence with the greatest GEMINI score.

We averaged the number of times each feature was selected according to the model over the 20 runs and sorted them decreasingly. This post-process revealed that Wasserstein-GEMINI consistently selected the El Salvador Aid and the Aid to Nicaraguan contras votes as sufficient to perform clustering. Indeed, these two votes are among the most discriminating features between Republicans and Democrats and were often chosen by other model-based methods (Fop & Murphy, 2018). The MMD-GEMINI objective only added the Physician fee freeze vote to this subset. Regarding the Heart-Statlog dataset, MMD-GEMINI consistently picked a subset of 8 features out of 13, including for example age or chest pain type as relevant variables. In contrast, Wasserstein-GEMINI did not consistently choose the same subset of variables, yet its top variables, which were selected more than 80% of the runs, agree with the MMD-GEMINI selection as well.

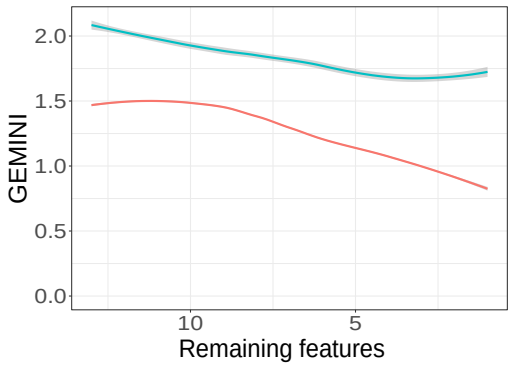


(a) GEMINI Loss depending on selected feature number

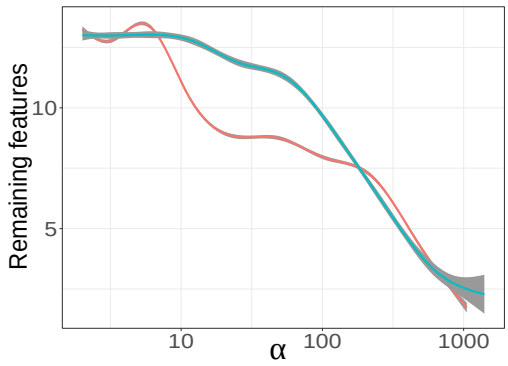


(b) Selected features depending on α

Figure 4.4 – Average training curves of Sparse GEMINI on the US Congress dataset over 50 runs. Blue lines correspond to Wasserstein-GEMINI, red lines to MMD-GEMINI.



(a) GEMINI depending on selected feature number



(b) Selected features depending on α

Figure 4.5 – Average training curves of Sparse GEMINI on the Heart Statlog dataset over 20 runs. Blue lines correspond to Wasserstein-GEMINI, red lines to MMD-GEMINI.

4.4.5.2 Scalability example with the Prostate-BCR dataset

To show the scalability of Sparse GEMINI, we demonstrate its performance on the Prostate-BCR dataset, taken from Vittrant et al. (2020) [†]. This dataset is a combination of transcriptomics data from 3 different sources: the Cancer Genom atlas (Abeshouse et al., 2015), the GSE54460 dataset from the NCBI website, and the PRJEB6530 project of the European Nucleotide Archive. The combined dataset contains 25,904 transcripts, *i.e.* variables, over 171 filtered patients with long-term follow-up, counting 52, 96 and 23 patients from the respective sources. The objective is to find biochemical recurrences (BCR) of prostate cancer through transcriptomic signature, hence binary targets.

To carefully eliminate the variables, we increase α gradually by 2%. We took a simple MLP with only one hidden layer of 100 neurons. We chose to run until converging to 400 features or less, following Vittrant et al. (2020). We trained Sparse GEMINI with OvA objectives 5 times to find

†. Available at https://github.com/ArnaudDroitLab/prostate_BCR_prediction

Table 4.5 – ARI of Sparse GEMINI (OvA) on the Heart-statlog and US Congress datasets with the average number of selected features. Standard deviation in subscript. Scores with an asterisk are the initial performances when using all features.

		Heart-statlog		US Congress	
		ARI	# Variables	ARI	# Variables
Sparse K-means		0.18 $_{0.00}$	13 $_{0.00}$	**0.54** $_{\mathbf{0.00}}$	16 $_{0.0}$
Clustvarsel		0.03 $_{0.00}$	2 $_{0.00}$	0.00 $_{0.00}$	2 $_{0.00}$
VSCC		0.27 $_{0.00}$	13 $_{0.00}$	0.40 $_{0.00}$	11 $_{0.00}$
Sparse Fisher EM		0.19 $_{0.00}$	1 $_{0.00}$	-	-
Logistic regression	MMD	**0.37** $_{\mathbf{0.03}}$	7.5 $_{0.51}$	**0.53** $_{\mathbf{0.02}}$	8.3 $_{0.81}$
	Wasserstein	0.33 $_{0.08}$	5.8 $_{2.09}$	0.48 $_{0.00}$	8.0 $_{0.92}$
MLP	MMD	0.32 $_{0.01}$	8.0 $_{0.00}$	0.48 $_{0.00}$	3.1 $_{0.37}$
	Wasserstein	0.32 $_{0.09}$	8.4 $_{2.70}$	0.47 $_{0.00}$	2.0 $_{0.00}$
MLP	MMD*	0.37 $_{0.02}$	13 -	0.55 $_{0.01}$	16 -
	Wasserstein*	0.33 $_{0.09}$	13 -	0.55 $_{0.02}$	16 -

either 2 clusters or 3 clusters in order to break down possible substructures among the supervised targets.

Interestingly, we observed in Table 4.6 that the clustering results did not catch up with the actual BCR targets, with an ARI close to 0 most of the time. However, upon evaluation of the clusters with respect to the original source of each sample, we found scores close to 1 of ARI in the case of MMD-GEMINI. Thus, the unsupervised algorithm was able to find sufficient differences in distribution between each source of data to discriminate them. Additionally, consistent subsets of features were always selected as the final subset on all 5 runs depending on the GEMINI. This implies that even without the best GEMINI within a range for feature selection, several runs can lead to identifying subsets of relevant data. This example illustrates how even in the presence of potentially legitimate labels, there exist other valid cluster structures in the data (Hennig, 2015)

These results can be viewed as discovering batch effect in the data. Batch effect, also known as batch variation, is a phenomenon that occurs in biological experiments where the results are affected by factors unrelated to the experimental variables being studied. These factors can include variations in sample processing, measurement conditions, people manipulating the samples, or equipment used. One common example of a batch effect is observed in microarray or RNA sequencing experiments, where the samples are processed in different batches and the results are affected by variations in the reagents or protocols used. It has been demonstrated that batch effects in microarray experiments originated from multiple causes, including variations in the labelling and hybridisation protocols used, which led to differences in the intensity of gene expression signals (Luo et al., 2010).

To minimise batch effects, it is important to control for variables such as reagents, protocols, and equipment used, and to use appropriate normalisation and data analysis methods to account for these variations. Several approaches can be used to detect batch effects in RNA-seq experiments, including PCA (Reese et al., 2013) and clustering. For this latter, Hierarchical clustering is often

Table 4.6 – ARI scores of the Prostate BCR dataset for various numbers of clusters depending on the chosen type of targets. We either use the expected targets (BCR) regarding cancer prediction, or data source targets that identify the data origin of each sample. The indicated GEMINIs are in the one-vs-all setting.

Model			#Var	ARI	
Architecture	GEMINI	K		BCR targets	Data source targets
Logistic regression	MMD	2	810 $_{590}$	-0.01 $_{0.00}$	0.79 $_{0.01}$
		3	1229 $_{2270}$	0.04 $_{0.00}$	**1.00** $_{\mathbf{0.01}}$
	Wasserstein	2	1334 $_{2561}$	0.01 $_{0.02}$	0.60 $_{0.13}$
		3	1430 $_{3127}$	0.04 $_{0.01}$	0.96 $_{0.06}$
MLP	MMD	2	4013 $_{6541}$	-0.01 $_{0.00}$	0.78 $_{0.01}$
		3	4287 $_{6590}$	0.03 $_{0.02}$	0.93 $_{0.11}$
	Wasserstein	2	4403 $_{6843}$	0.00 $_{0.00}$	0.65 $_{0.04}$
		3	4331 $_{6742}$	0.02 $_{0.02}$	0.80 $_{0.20}$

used as a method that groups samples based on their similarity in gene expression patterns, and batch effects can be identified based on dendrogram analysis (Leek et al., 2010).

4.4.6 Discussion

Our first observation from Table 4.2 is that Sparse GEMINI can reach performances close to some competitors in terms of ARI while performing better in variable selection, especially with OvO MMD-GEMINI. The MMD is a distance computed between expectations making it thus insensitie to small variations of the kernel, typically when noisy variables are introduced contrary to the Wasserstein distance which takes a global point of view on the distribution. Specifically, the algorithm is good at discarding noisy variables, but less competitive regarding redundant variables as illustrated with the second synthetic dataset. Nonetheless, the ARI remains competitive even though the model failed to give the correct ground for the clustering.

Additionally, the training path produces critical values of α at which features disappear. Thus, the algorithm produces an explicit unsupervised metric of the relevance of each feature according to the clustering. Typically, plateaus of the number of used variables like in figures 4.4b and 4.5b for the MMD shed light on different discriminating subsets. We also find that the empirical threshold of 90% of the maximal GEMINI to select fewer variables is an efficient criterion. In case of a too sudden collapse of variables, we encourage training new models on iteratively selected subsets of features. Indeed, as α increases during training, the collapse of the number of selected variables will often happen when the geometric increase is too strong, which might lead to unstable selections.

4.5 Conclusion

We presented a novel algorithm named Sparse GEMINI that jointly performs clustering and feature selection by combining GEMINI for objective and an ℓ_1 penalty. The algorithm shows good

performances in eliminating noisy irrelevant variables while maintaining relevant clustering. Owing to the nature of multi-layered perceptrons, Sparse GEMINI is easily scalable to high-dimensional data and provides thus an unsupervised technique to get a projection of the data. However, the limits of the scalability are the number of clusters and samples per batch due to the complex nature of GEMINI. Thus, we believe that Sparse GEMINI is a relevant algorithm for multi-omics data where the number of samples is often little and the number of features large, especially when it is hard to design a good generative model for such data. As a concluding remark, we want to draw again the attention to the discriminative nature of the algorithm: Sparse GEMINI focuses on the design of a decision boundary instead of parametric assumptions.

Although Sparse GEMINI provides a subset of variables with which clustering was done, its nature in the case of an MLP leaves an opaque idea of the relevance of each of these variables to the clusters. As a consequence, we will explore in the next chapter how we can orchestrate the intervention of each variable within the GEMINI framework using intrinsically interpretable models: trees.

CHAPTER 5

From neural networks to trees: explainable discriminative clustering with Kauri and Douglas

6.1 Introduction . . . 115
6.2 Known phenogroups . . . 116
6.3 Methods . . . 117
6.3.1 PROGRESSA modalities and datasets . . . 117
6.3.2 Preprocessing . . . 120
6.3.3 Applying multiple Sparse GEMINI models . . . 120
6.3.4 Variables ranking . . . 121
6.3.5 Accuracy filtering for time stability . . . 121
6.3.6 Consensus clustering . . . 121
6.3.6.1 Building the consensus matrix . . . 121
6.3.6.2 Using the consensus matrix . . . 122
6.3.6.3 Consensus model . . . 122
6.4 From clusters to phenogroups . . . 123
6.4.1 Results and metrics . . . 123
6.4.1.1 Accuracy of models for subsequent visits . . . 123
6.4.1.2 Ambiguity of the filtered models for consensus . . . 124
6.4.2 Identifying phenogroups . . . 124
6.4.2.1 Merging clusters . . . 124
6.4.2.2 Visualising the pipeline outputs . . . 125
6.4.2.3 Characteristics comparisons . . . 125
6.4.2.4 Endpoints of the clusters . . . 131
6.5 Conclusion . . . 133

5.1 Introduction

We explored in Chapter 3 the development of an objective function, the GEMINI, to train any discriminative model for clustering, then introduced sparsity in the models in Chapter 4 for

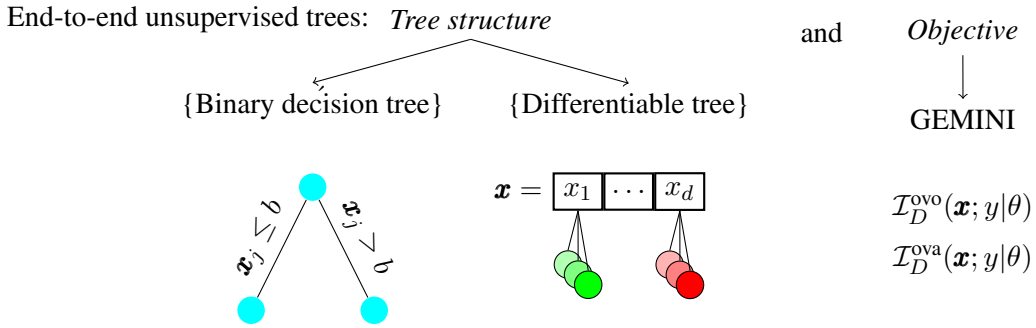


Figure 5.1 – Summary of the proposed framework for learning end-to-end unsupervised trees. The framework concatenates a tree structure with an objective to maximise: the generalised mutual information. The Kauri model corresponds to a binary decision tree with the squared-MMD-GEMINI whereas the Douglas model corresponds to a differentiable tree and Wasserstein-GEMINI.

enhancing interpretability. However, in both cases we mainly focused on neural networks, which may not be instrinsically interpretable. Although we included sparse logistic regression in Chapter 4, we can train simpler and/or non-differentiable models to maximise the GEMINI. We focus in this chapter on the training of trees.

Decision tree classifiers are one of the most intuitive models in machine learning owing to their intrinsic interpretability (Molnar, 2020, Section 3.2). Decision trees consist of a set of hierarchically sorted nodes starting from one single root node. Each node comprises two or more conditions called rules, each of which leading to a different child node. Once a node does not have any child, a decision is returned. A childless node is named a leaf.

While the end model is eventually interpretable, building it implies some questions to be addressed, notably regarding the number of nodes, the feature (or set of features) on which to apply a decision rule, the construction of a decision rule *i.e.* the number of thresholds and hence the number of children per node. Learning the structure is easier in the case of supervised learning, whereas the absence of labels makes the construction of unsupervised trees more challenging. In recent related works, the problem was oftentimes addressed with twofold methods (Laber et al., 2023 ; Tavallali et al., 2021): first learning clusters using another algorithm *e.g.* K-means, then applying a supervised decision tree to uncover explanations of the clusters. However, such *unsupervised trees* are not fully unsupervised in fact since their training still requires the presence of external labels for guidance which are provided by K-means.

To achieve end-to-end unsupervised learning in trees, we propose a framework where we merge the view of trees as statistical models with learnable parameters and a clustering criterion to maximise: GEMINI. We derive two new clustering algorithms from this framework; respectively binary decision trees for datasets with a large number of features (Kauri) and k-ary differentiable trees for datasets with a large number of samples (Douglas). A short description of these methods is provided in Fig. 5.1.

5.2 Training trees

We progressively present in this section the different means for creating a decision tree structure, with supervision or not.

5.2.1 How do we train supervised trees?

In supervised learning, we have access to targets y which guides our tree construction for separating our samples. In this field, we can refer to the well-known classification and regression tree (CART) (Breiman, 1984). At each node, we evaluate the quality of a split, i.e. a proposed rule on a given feature and data-dependent threshold, through gain metrics. We then add to the tree structure the split that achieved the highest possible gain. Common implementations of supervised trees use the Gini criterion developed by the statistician Corrado Gini (1912), which indicates how *pure* a tree node is given the proportion of different labels in its samples (Casquilho & Österreicher, 2018). Later works then proposed different gain metrics like the difference of mutual information in the ID3 (Quinlan, 1986) and C4.5 (Quinlan, 2014) algorithms.

When the number of leaves and features to explore is unlimited, these approaches can produce deterministic results. Moreover, their greedy nature can lead to the construction of very deep trees which harms the interpretable nature of the model (Luštrek, Gams, & Martinčić-Ipšić, 2016). This motivates for example the construction of multiple trees that are equivalent in terms of decision, yet different in terms of structure presenting thus an overview of the Rashomon set for interpretations (Xin et al., 2022). Other approaches tried to overcome the deterministic non-differentiable nature of the rule-based tree by introducing differentiable leaves (Fang, Jennings, Wen, Li, & Li, 1991 ; Y. Yang, Morillo, & Hospedales, 2018) which allows to train trees through gradient descent. We will later come back to the definition of one such model for our method, the deep neural decision tree (Y. Yang et al., 2018).

Whether differentiable or not, we choose to describe the decision trees as statistical models $p_\theta(y|\boldsymbol{x})$ which assign the data sample $\boldsymbol{x}$ to a discrete variable y, the cluster membership, according to some parameters θ. These parameters can be for example the set of thresholds and features on which decisions are carried at each node or matrix weights in differentiable trees as we will see in the next sections.

5.2.2 How do we train unsupervised trees?

In clustering, we do not have access to labels making all previous notions of gains unusable, so we need other tools for guiding the splitting procedure of the decision trees. A common approach is then to keep the algorithm supervised as described in the previous section, yet providing labels that were derived from a clustering algorithm e.g. K-means (Held & Buhmann, 1997 ; Laber et al., 2023). In this sense, centroids derived from K-means can also be involved in split procedures (Tavallali et al., 2021), even to the point that the data from which the centroids are derived do not need to be collected (Gamlath et al., 2021). However, such methods do not properly construct the tree *from scratch* in an unsupervised way despite potential changes in the gain formulations. We are interested in a method that can provide a directly integrated objective to optimise tree training. Other gains derived from entropy formulations can also be proposed (Basak & Krishnapuram, 2005 ; Bock, 1994). We even note the usage of recursive writing of mutual information to achieve deeper and deeper refinements of binary clusters (Karakos, Khudanpur, Eisner, & Priebe, 2005).

Oftentimes, these approaches assume that a leaf describes fully a cluster, e.g. Blockeel, Raedt, et Ramon (1998). Combining leaves into a single cluster requires then post hoc methods (Fraiman, Ghattas, & Svarc, 2013). In such a case, an elegant approach for constructing an unsupervised tree was proposed by Liu, Xia, et Yu (2000) by adding uniform noise to the data and assigning a

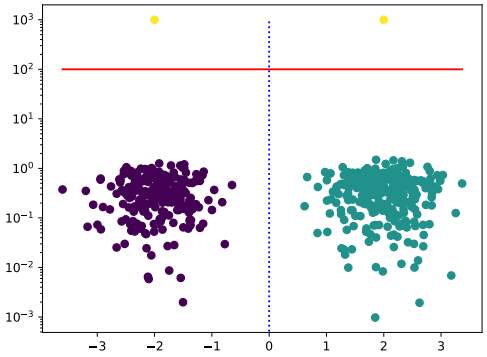


Figure 5.2 – A dataset proposed by Moshkovitz et al. (2020) consisting in two isotropic Gaussian distributions and a cluster of two points distant on the y-axis. In order to split optimally the clusters, a decision tree should start with a y-axis split (solid red line) then use an x-axis split (dashed blue line) to separate the two Gaussian distributions.

decision tree to separate the noise from the true data. Such trees put in different leaves dense areas of the data, which can then be labelled manually.

To ensure that several leaves can be assigned to a single cluster, related work also focused on the complete initialisation of a tree and refinement according to a global objective function. For example, Bertsimas et al. (2021) directly maximise the silhouette score or the Dunn index, which are internal clustering metrics and require the initialisation of the tree through greedy construction or K-means labels. The objective is optimised using a mixed integer optimisation formulation of the tree structure. Lately, Gabidolla et Carreira-Perpinan (2022) proposed to optimise an oblique tree, a more difficult tree structure through the alternative optimisation of a distance-based objective, e.g. K-means, providing pseudo-labels to a tree alternating optimisation problem (Carreira-Perpinan & Tavallali, 2018).

5.2.3 Related motivating example

To justify the interest in seeking novel algorithms for unsupervised clustering trees instead of naively applying K-means then a supervised CART tree on the obtained clusters, Moshkovitz, Dasgupta, Rashtchian, et Frost (2020, Figure 2b) created a simple dataset where this naive combination would solve the task with excellent accuracy, yet with non-optimal splits.

This dataset consists in 3 clusters. The first two ones are respectively drawn from two Gaussian distributions: $\mathcal{N}([2,0]^\top, \epsilon \boldsymbol{I}_2)$ and $\mathcal{N}([-2,0], \epsilon \boldsymbol{I}_2)$ with ϵ small enough. The last cluster contains two points located at $(-2, v)$ and $(2, v)$. We plot in Figure 5.2 a sample of such dataset for $v = 1000$.

A decision tree learning from K-means labels will start by separating the samples along the x-axis. This non-optimal choice then requires two splits on the left- and right-hand sides to then separate the Gaussian distributions from the third cluster. The optimal choice, achieved by ExKMC (Moshkovitz et al., 2020), starts by cutting on the y-axis, separating thus all Gaussian distributions from the third cluster. A single split afterwards is sufficient for separating the two Gaussian distributions.

The Kauri model that we now introduce is also able to find such optimal splits.

5.3 Kauri: K-means as unsupervised reward ideal

The Kauri tree is a non-differentiable binary decision tree that looks in many ways alike the CART algorithm. It constructs from scratch a binary tree giving hard clustering assignments to the data by using an objective equivalent to both the optimisation of kernel K-means and MMD-GEMINI. In the Kauri structure, a cluster can be described by several leaves.

5.3.1 Notations and modelling

We consider that we have a dataset of n samples: $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$. We can model the classification/clustering distribution associated with decision trees as a delta Dirac:

$$p_\theta(y = k|\boldsymbol{x}) = \mathbb{1}[\boldsymbol{x} \in \mathcal{X}_k]\,, \tag{5.1}$$

with $\{\mathcal{X}_k\}_{k=1}^K$ a partition of the data space $\mathcal{X}$. Notice that we use the notation $\mathbb{1}$ because y is discrete. We set $\mathcal{X} \subseteq \mathbb{R}^d$. We write the partition into K clusters as the sets samples that fall in the respective data subspace:

$$\mathcal{C}_k = \{\boldsymbol{x}_i \in \mathcal{X}_k\}, \forall k \leq K. \tag{5.2}$$

We assume that the model sees all the data, *i.e.* we don't use mini-batches, and that $p_{\text{data}}(\boldsymbol{x})$ corresponds to the empirical distribution of the training data. Consequently, the expectations of the model turn to discrete sums. Notably, we have:

$$p_\theta(y = k) = \frac{|\mathcal{C}_k|}{n}. \tag{5.3}$$

We note $\mathcal{T}_p$ the set of samples reaching the p-th node and τ_{pj} its threshold defined for a single feature j. This threshold defines two binnings and produces two child nodes. For example, if $\boldsymbol{x}_{ij} \leq \tau_{pj}$, then this sample goes to the left child of the parent node p, otherwise to the right child.

5.3.2 Objective

The kernel KMeans algorithm minimises the cluster sum of squares in a Hilbert space $\mathcal{H}$ with projection φ and kernel κ with respect to $K_{\max}$ centroids $\boldsymbol{\mu}_k \dots \boldsymbol{\mu}_{K_{\max}}$:

$$\mathcal{L}_{\text{KMeans}} = \sum_{k=1}^{K_{\max}} \sum_{\boldsymbol{x} \in \mathcal{C}_k} \|\varphi(\boldsymbol{x}) - \boldsymbol{\mu}_k\|_{\mathcal{H}}^2. \tag{5.4}$$

Instead of computing the sample-wise distance to the centroid using the kernel trick (Dhillon et al., 2004), Kauri optimises this objective without explicitly computing centroids per cluster. To that end, we use the following simple equality:

$$\sum_{\boldsymbol{x} \in \mathcal{C}_k} \|\varphi(\boldsymbol{x}) - \boldsymbol{\mu}_k\|_{\mathcal{H}}^2 = \frac{1}{2\,|\mathcal{C}_k|} \sum_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{C}_k} \|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})\|_{\mathcal{H}}^2. \tag{5.5}$$

This equality can be briefly shown using the kernel trick and the bilinearity of the kernel:

$$\sum_{\boldsymbol{x}\in\mathcal{C}_k} \|\varphi(\boldsymbol{x}) - \frac{1}{|\mathcal{C}_k|}\sum_{\boldsymbol{y}\in\mathcal{C}_k}\varphi(\boldsymbol{y})\|^2_{\mathcal{H}} = \sum_{\boldsymbol{x}\in\mathcal{C}_k}\Bigg(\langle\varphi(\boldsymbol{x}),\varphi(\boldsymbol{x})\rangle + \frac{1}{|\mathcal{C}_k|^2}\langle\sum_{\boldsymbol{y}\in\mathcal{C}_k}\varphi(\boldsymbol{y}),\sum_{\boldsymbol{y}\in\mathcal{C}_k}\varphi(\boldsymbol{y})\rangle \tag{5.6}$$

$$-\frac{2}{|\mathcal{C}_k|}\langle\varphi(\boldsymbol{x}),\sum_{\boldsymbol{y}\in\mathcal{C}_k}\varphi(\boldsymbol{y})\rangle\Bigg) \tag{5.7}$$

$$= \sum_{\boldsymbol{x}\in\mathcal{C}_k}\langle\varphi(\boldsymbol{x}),\varphi(\boldsymbol{x})\rangle - \frac{1}{|\mathcal{C}_k|}\sum_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}_k}\langle\varphi(\boldsymbol{x}),\varphi(\boldsymbol{y})\rangle. \tag{5.8}$$

This demonstration is finished by summing $\mathcal{C}_k$ times the first term on the unused variable $\boldsymbol{y}$ to retrieve the expression of the norm using kernels between $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\sum_{\boldsymbol{x}\in\mathcal{C}_k}\|\varphi(\boldsymbol{x}) - \boldsymbol{\mu}_k\|^2_{\mathcal{H}} = \sum_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}_k}\left(\frac{1}{|\mathcal{C}_k|}\langle\varphi(\boldsymbol{x}),\varphi(\boldsymbol{x})\rangle - \frac{1}{|\mathcal{C}_k|}\langle\varphi(\boldsymbol{x}),\varphi(\boldsymbol{y})\rangle\right) \tag{5.9}$$

$$= \frac{1}{|\mathcal{C}_k|}\sum_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}_k}\left(\frac{1}{2}\langle\varphi(\boldsymbol{x}),\varphi(\boldsymbol{x})\rangle + \frac{1}{2}\langle\varphi(\boldsymbol{y}),\varphi(\boldsymbol{y})\rangle - \langle\varphi(\boldsymbol{x}),\varphi(\boldsymbol{y})\rangle\right) \tag{5.10}$$

$$= \frac{1}{2\,|\mathcal{C}_k|}\sum_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}_k}\|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})\|^2_{\mathcal{H}}. \tag{5.11}$$

Once inserted into the kernel KMeans objective, we get an alternative formulation without centroids:

$$\mathcal{L}_{\text{KMeans}} = \sum_{k=1}^{K_{\max}}\frac{1}{2\,|\mathcal{C}_k|}\sum_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}_k}\|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})\|^2_{\mathcal{H}}. \tag{5.12}$$

Using the kernel trick, we can rephrase this objective as:

$$\mathcal{L}_{\text{KMeans}} = \sum_{k=1}^{K_{\max}}\frac{1}{2\,|\mathcal{C}_k|}\sum_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}_k}(\kappa(\boldsymbol{x},\boldsymbol{x}) + \kappa(\boldsymbol{y},\boldsymbol{y}) - 2\kappa(\boldsymbol{x},\boldsymbol{y})), \tag{5.13}$$

where the two first kernel terms can be summarised as the size of clusters weighting the diagonal elements of the kernel. Finally, the third term is the grand sum of the kernel of the cluster, and thus:

$$\mathcal{L}_{\text{KMeans}} = \sum_{\boldsymbol{x}\in\mathcal{D}}\kappa(\boldsymbol{x},\boldsymbol{x}) - \sum_{k=1}^{K_{\max}}\frac{1}{|\mathcal{C}_k|}\sum_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}_k}\kappa(\boldsymbol{x},\boldsymbol{y}). \tag{5.14}$$

For the sake of simplicity, we introduce the function σ that sums the kernel values $\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) = \langle\varphi(\boldsymbol{x}_i),\varphi(\boldsymbol{x}_j)\rangle$ of samples indexed by two sets:

$$\sigma(E\times F) = \sum_{\substack{\boldsymbol{x}_i\in E\\ \boldsymbol{x}_j\in F}}\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j). \tag{5.15}$$

We will refer to the σ function as the *kernel stock*. This function is bilinear with respect to the input spaces. We provide in Figure 5.3 a visual explanation of its different usages.

We finally note that the first term of Eq. (5.14) is a constant because it does not depend on the clustering. With only the second term remaining, we can remove the minus sign and hence maximise the objective:

$$\mathcal{L}_{\text{Kauri}} = \sum_{k=1}^{K_{\max}} \frac{\sigma(\mathcal{C}_k^2)}{|\mathcal{C}_k|}, \tag{5.16}$$

Therefore, maximising our objective function $\mathcal{L}$ is equivalent up to a constant to minimising a KMeans objective for any kernel. However, in contrast to Eq. (5.4), it is not a function of centroids, but a function of a partition. This objective can also be connected to both OvA and OvO MMD-GEMINIs from Chaper 3.

5.3.2.1 Equivalence to OvA MMD-GEMINI

We start by recovering the definition of OvA squared-MMD-GEMINI using only the outputs of the outputs of our model $p_\theta(y|\boldsymbol{x})$:

$$\mathcal{I}^{\text{ova}}_{D_{\text{MMD}^2}}(\boldsymbol{x}; y|\theta) = \mathbb{E}_{y\sim p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p(\boldsymbol{x})}\left[\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\left(\frac{p_\theta(y|\boldsymbol{x}_a)p_\theta(y|\boldsymbol{x}_b)}{p_\theta(y)^2} + 1 - 2\frac{p_\theta(y|\boldsymbol{x}_a)}{p_\theta(y)}\right)\right]\right]. \tag{5.17}$$

We can replace the expectations with discrete sums for both the clusters and the data. Notice the factor $\frac{1}{n}$ in front of the kernel as we are simply doing a Monte Carlo estimate using the empirical data distribution. We replace at the same time the values of the distributions by either the indicator functions or cluster sizes:

$$\mathcal{I}^{\text{ova}}_{D_{\text{MMD}^2}}(\boldsymbol{x}; y|\theta) = \sum_{k=1}^{K}\frac{|\mathcal{C}_k|}{n}\sum_{\substack{i=1\\j=1}}^{n}\frac{\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)}{n^2}\left(\frac{\mathbb{1}[\boldsymbol{x}_i\in\mathcal{X}_k]\,\mathbb{1}[\boldsymbol{x}_j\in\mathcal{X}_k]\,n^2}{|\mathcal{C}_k|^2} + 1 - 2\frac{\mathbb{1}[\boldsymbol{x}_i\in\mathcal{X}_k]\,n}{|\mathcal{C}_k|}\right). \tag{5.18}$$

By applying the indicator functions, we see that we sum the terms of a kernel on condition that the respective samples belong specifically to some subset of data. We can consequently rewrite the inner sum as a combination of *kernel stocks* σ:

$$\mathcal{I}^{\text{ova}}_{D_{\text{MMD}^2}}(\boldsymbol{x}; y|\theta) = \sum_{k=1}^{K}\frac{|\mathcal{C}_k|}{n}\left(\frac{\sigma(\mathcal{C}_k^2)}{|\mathcal{C}_k|^2} + \frac{\sigma(\mathcal{D}^2)}{n^2} - 2\frac{\sigma(\mathcal{C}_k\times\mathcal{D})}{n\,|\mathcal{C}_k|}\right). \tag{5.19}$$

Then, we develop all 3 terms and get:

$$\mathcal{I}^{\text{ova}}_{D_{\text{MMD}^2}}(\boldsymbol{x}; y|\theta) = \sum_{k=1}^{K}\frac{\sigma(\mathcal{C}_k^2)}{n\,|\mathcal{C}_k|} + \frac{|\mathcal{C}_k|\,\sigma(\mathcal{D}^2)}{n^3} - 2\frac{\sigma(\mathcal{C}_k\times\mathcal{D})}{n^2}. \tag{5.20}$$

We can obtain the final form of the GEMINI by summing constant terms. Observing that $\sum_k |\mathcal{C}_k| = n$ for the first term and using the bilinearity of σ for the second term, we have:

$$\mathcal{I}^{\text{ova}}_{D_{\text{MMD}^2}}(\boldsymbol{x}; y|\theta) = -\frac{\sigma(\mathcal{D}^2)}{n^2} + \sum_{k=1}^{K}\frac{\sigma(\mathcal{C}_k^2)}{n\,|\mathcal{C}_k|}. \tag{5.21}$$

As we are interested in optimising the clustering assignments in the tree, we can remove all constant terms and factors that do not bring extra information. Hence, the final objective $\mathcal{L}$ to maximise is:

$$\mathcal{L} = \sum_{k=1}^{K} \frac{\sigma(\mathcal{C}_k^2)}{|\mathcal{C}_k|}. \tag{5.22}$$

5.3.2.2 Equivalence to OvO MMD-GEMINI

We now prove that the objective function $\mathcal{L}$ obtained in the previous section is also equivalent to maximising the OvO squared MMD-GEMINI in case of delta Dirac classifiers. We start by expressing the complete squared MMD-GEMINI:

$$\mathcal{I}^{\text{ovo}}_{D_{\text{MMD}^2}}(\boldsymbol{x}; y|\theta) = \mathbb{E}_{y_a, y_b \sim p_\theta(y)} \left[\mathbb{E}_{\boldsymbol{x}_a, \boldsymbol{x}_b \sim p(\boldsymbol{x})} \left[\kappa(\boldsymbol{x}_a, \boldsymbol{x}_b) \left(\frac{p_\theta(y_a|\boldsymbol{x}_a) p_\theta(y_a|\boldsymbol{x}_b)}{p_\theta(y_a)^2} + \frac{p_\theta(y_b|\boldsymbol{x}_a) p_\theta(y_b|\boldsymbol{x}_b)}{p_\theta(y_b)^2} - 2\frac{p_\theta(y_a|\boldsymbol{x}_a) p_\theta(y_b|\boldsymbol{x}_b)}{p_\theta(y_a) p_\theta(y_b)} \right) \right] \right]. \tag{5.23}$$

As we exactly did for the OvA MMD-GEMINI demonstration, we apply the following tricks: discretising the expectations on the dataset $\mathcal{D}$, re-expressing the cluster proportions, simplifying the sums. Our discrete version is:

$$\mathcal{I}^{\text{ovo}}_{D_{\text{MMD}^2}}(\boldsymbol{x}; y|\theta) = \sum_{k,k'}^{K} \frac{|\mathcal{C}_k|\,|\mathcal{C}_{k'}|}{n^2} \sum_{\substack{x_i \in \mathcal{D} \\ x_j \in \mathcal{D}}} \frac{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)}{n^2} \left(\frac{n^2 \mathbb{1}[\boldsymbol{x}_i \in \mathcal{X}_k]\, \mathbb{1}[\boldsymbol{x}_j \in \mathcal{X}_k]}{|\mathcal{C}_k|^2} + \frac{n^2 \mathbb{1}[\boldsymbol{x}_i \in \mathcal{X}_{k'}]\, \mathbb{1}[\boldsymbol{x}_j \in \mathcal{X}_{k'}]}{|\mathcal{C}_{k'}|^2} - 2\frac{n^2 \mathbb{1}[\boldsymbol{x}_i \in \mathcal{X}_k]\, \mathbb{1}[\boldsymbol{x}_j \in \mathcal{X}_{k'}]}{|\mathcal{C}_k|\,|\mathcal{C}_{k'}|} \right). \tag{5.24}$$

We can cancel the factors in n^2 and replace our sum over indicator functions by the *kernel stock* function σ. Thus, we obtain:

$$\mathcal{I}^{\text{ovo}}_{D_{\text{MMD}^2}}(\boldsymbol{x}; y|\theta) = \sum_{k,k'}^{K} \frac{|\mathcal{C}_k|\,|\mathcal{C}_{k'}|}{n^2} \left(\frac{\sigma(\mathcal{C}_k \times \mathcal{C}_k)}{|\mathcal{C}_k|^2} + \frac{\sigma(\mathcal{C}_{k'} \times \mathcal{C}_{k'})}{|\mathcal{C}_{k'}|^2} - 2\frac{\sigma(\mathcal{C}_k \times \mathcal{C}_{k'})}{|\mathcal{C}_k|\,|\mathcal{C}_{k'}|} \right). \tag{5.25}$$

For the first two terms, we can cancel one part of the summation. Indeed, the *kernel stock* of $\mathcal{C}_k$ does not depend on k', consequently, the sum over k' just multiplies this *kernel stock* up to a factor n that will be cancelled by the denominator n at the very start. The same reasoning goes for the second term. Last but not least, we can cancel cluster sizes on the last term. Our expression is then:

$$\mathcal{I}^{\text{ovo}}_{D_{\text{MMD}^2}}(\boldsymbol{x}; y|\theta) = 2\sum_{k=1}^{K} \frac{\sigma(\mathcal{C}_k \times \mathcal{C}_k)}{n\,|\mathcal{C}_k|} - 2\sum_{k,k'}^{K} \frac{\sigma(\mathcal{C}_k \times \mathcal{C}_{k'})}{n^2}. \tag{5.26}$$

As we now look forward to maximising this expression, we can realise that the last term is simply the *kernel stock* of the dataset, in other words, a constant with respect to the clustering. We

can then discard this term. For the first term, we simply remove the constant factor $2/n$ to obtain the equivalent:

$$\mathcal{I}^{\text{OVO}}_{D_{\text{MMD}^2}}(\boldsymbol{x}; y|\theta) = \frac{2}{n}\left(\sum_{k=1}^{K}\frac{\sigma(\mathcal{C}_k \times \mathcal{C}_k)}{|\mathcal{C}_k|} - \frac{\sigma(\mathcal{D}\times\mathcal{D})}{n}\right) \propto \mathcal{L} + \text{constant}. \tag{5.27}$$

This concludes our proof. Consequently, we that Eq. (5.21) is equal up to a factor 2 to Eq. (5.27) and this equality holds for any non-null proportion of clusters.

Proposition 5.3.1. *Let $p(y = k|\boldsymbol{x}) = \mathbb{1}[\boldsymbol{x} \in \mathcal{X}_k]$ be a non-degenerate clustering distribution on the partition of the data space $\mathcal{X} = \uplus_{k=1}^{K}\mathcal{X}_k$. Then $\mathcal{I}^{OvO}_{MMD^2}(\boldsymbol{x}; y) = 2\mathcal{I}^{OvA}_{MMD^2}(\boldsymbol{x}; y)$.*

5.3.3 Tree branching

For supervised trees like CART or ID3, the types of splits are binary and guided by the labels which tell us to which class each child node should go. For unsupervised trees, we must consider all possibilities: to which cluster goes the left child, to which cluster goes the right child, on which feature to do the split, on what threshold within this feature to split, on which nodes. Assuming to be located at a node p for a split, let $\mathcal{S}_L$ the subset of samples from the node samples $\mathcal{T}_p$ that will go to the left child node and $\mathcal{S}_R$ the complementary subset of samples that will go to the right child node. Each child node will be assigned to a different cluster, whether new, already existing or equal to the parent node's cluster assignment. Let k_p be the current cluster membership of the parent node p, k_L the future cluster membership for the left child node and k_R the future cluster membership of the right child node, then $\mathcal{S}_L \cup \mathcal{S}_R = \mathcal{T}_p \subseteq \mathcal{C}_{k_p}$ and after splitting: $\mathcal{S}_L \subseteq \mathcal{C}_{k_L}$ and $\mathcal{S}_R \subseteq \mathcal{C}_{k_R}$.

We enforce the following constraints: (i) a child node must stay in the parent node's cluster if both children leaving would empty the parent's cluster; (ii) the creation of a new cluster can only be done under the condition that the number of clusters does not exceed a specified limit $K_{\max}$. We also impose a maximum number of leaves $T_{\max}$ which can be equal to at most the number of samples n. It is nonetheless possible that the algorithm stops the splitting procedure if all gains become negative before reaching the maximum number of leaves allowed.

Thus, learning consists in greedily exploring from all nodes the best split and either taking this split to build a new cluster or merging with another cluster. We now present the objective function and related gains depending on the children's cluster memberships.

5.3.4 Gain metrics

5.3.4.1 General expression of a gain

We can derive from the Kauri objective (Eq. 5.16) four gains that evaluate how much score we get by assigning one child node to a new cluster, assigning both child nodes to two new clusters, merging one child node to another cluster or merging both child nodes to different clusters. We denote by $\mathcal{C}'_\bullet$ the clusters after the split operation and $\mathcal{C}_\bullet$ the clusters before the split. Hence, the global gain metric is:

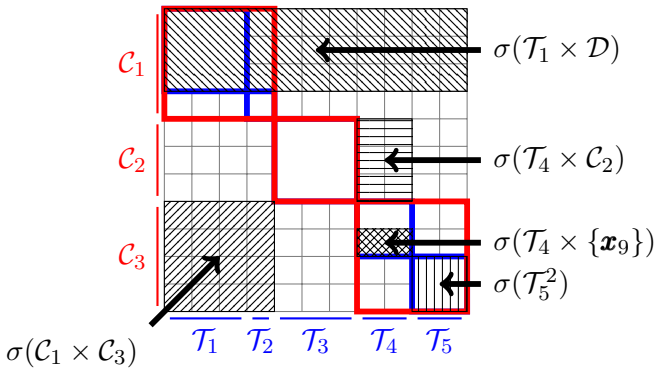


Figure 5.3 – A toy example with a dataset consisting of 11 samples partitioned in 3 clusters using 5 leaves in a tree. The matrix represents the kernel between all pairs of samples and dashed areas correspond to the sum of kernel elements according to the *kernel stock* function σ.

$$\Delta\mathcal{L}(\mathcal{S}_L : k_p \to k_L, \mathcal{S}_R : k_p \to k_R) = \frac{\sigma(\mathcal{C}'^{\,2}_{k_L})}{\left|\mathcal{C}'_{k_L}\right|} + \frac{\sigma(\mathcal{C}'^{\,2}_{k_R})}{\left|\mathcal{C}'_{k_R}\right|} + \frac{\sigma(\mathcal{C}'^{\,2}_{k_p})}{\left|\mathcal{C}'_{k_p}\right|} - \frac{\sigma(\mathcal{C}^2_{k_L})}{|\mathcal{C}_{k_L}|} - \frac{\sigma(\mathcal{C}^2_{k_R})}{|\mathcal{C}_{k_R}|} - \frac{\sigma(\mathcal{C}^2_{k_p})}{\left|\mathcal{C}_{k_p}\right|}, \quad (5.28)$$

which corresponds to subtracting the contribution of the kernel stocks of the former clusters and adding the kernel stocks of the new clusters after splitting. From this global gain metric, we derive four different gains: the *star gain* $\Delta\mathcal{L}^\star$ for assigning either the left or right child of a leaf to a new cluster, the *double star gain* $\Delta\mathcal{L}^{\star\star}$ for assigning the left and right children of a leaf to two new clusters, the *switch gain* $\Delta\mathcal{L}^{\rightleftarrows}$ for assigning either the left or right child of a leaf to another existing cluster and the *reallocation gain* $\Delta\mathcal{L}^{\hookrightarrow}$ for assigning respectively the left and right children to different existing clusters. We provide Figure 5.3 for visual purpose and assistance in the demonstrations.

5.3.4.2 Creating a new cluster: the *star gain*

In this case, we assign one of the splits $\mathcal{S}_L$ or $\mathcal{S}_R$ to a new cluster and let the other split in the same cluster as the parent node, *i.e.* either $k_L = K + 1$ and $k_R = k_p$ or $k_L = k_p$ and $k_R = K + 1$. Taking the case where the left split is given to a new cluster, we derive from the global gain a variation that we call *star gain*:

$$\Delta\mathcal{L}^\star(\mathcal{S}_L : k_p \to k_L) = \frac{\sigma(\mathcal{S}_L^2)}{|\mathcal{S}_L|} + \frac{\sigma(\mathcal{C}'^{\,2}_{k_p})}{\left|\mathcal{C}'_{k_p}\right|} - \frac{\sigma(\mathcal{C}^2_{k_p})}{\left|\mathcal{C}_{k_p}\right|}. \quad (5.29)$$

However, that expression is not convenient since there is a clear dependence: $\mathcal{C}'_{k_p} = \mathcal{C}_{k_p} \setminus \mathcal{S}_L$ and we would be interested in avoiding the evaluation of $\sigma(\mathcal{C}'^{\,2}_{k_p})$. We can use the bilinearity of the σ function and decompose over the new cluster $\mathcal{C}'_{k_p} = \mathcal{C}_{k_p} \setminus \mathcal{S}_L$. Similarly, we can reexpress the cardinal as $\left|\mathcal{C}'_{k_p}\right| = \left|\mathcal{C}_{k_p}\right| - |\mathcal{S}_L|$. Consequently, our term becomes:

$$\Delta\mathcal{L}^{\star}(\mathcal{S}_L : k_p \to k_L) = \frac{\sigma(\mathcal{S}_L^2)}{|\mathcal{S}_L|} + \frac{\sigma(\mathcal{C}_{k_p}^2) - 2\sigma(\mathcal{C}_{k_p} \times \mathcal{S}_L) + \sigma(\mathcal{S}_L^2)}{\left|\mathcal{C}_{k_p}\right| - |\mathcal{S}_L|} - \frac{\sigma(\mathcal{C}_{k_p}^2)}{\left|\mathcal{C}_{k_p}\right|}. \tag{5.30}$$

It is then just a matter of reordering with respect to the *kernel stocks* σ to obtain the final equation:

$$\Delta\mathcal{L}^{\star}(\mathcal{S}_L : k_p \to k_L) = \sigma(\mathcal{S}_L^2)\left(\frac{1}{|\mathcal{S}_L|} + \frac{1}{\left|\mathcal{C}_{k_p}\right| - |\mathcal{S}_L|}\right) + \sigma(\mathcal{C}_{k_p}^2)\left(\frac{1}{\left|\mathcal{C}_{k_p}\right| - |\mathcal{S}_L|} - \frac{1}{\left|\mathcal{C}_{k_p}\right|}\right) - 2\frac{\sigma(\mathcal{C}_{k_p} \times \mathcal{S}_L)}{\left|\mathcal{C}_{k_p}\right| - |\mathcal{S}_L|}, \tag{5.31}$$

which will be the used equation for the *star gain*.

5.3.4.3 Creating two clusters: the *double star gain*

The operation of creating two clusters can be seen as assigning in a first step the complete node p to a new cluster, then taking one of its split and assigning it to a second new cluster. The *double star gain* $\Delta\mathcal{L}^{\star\star}$ can be thus computed by the sum of $\Delta\mathcal{L}^{\star}$ with $\mathcal{T}_p$ replacing the source cluster $\mathcal{C}_{k_p}$ and another $\Delta\mathcal{L}^{\star}$ with $\mathcal{T}_p$ replacing $\mathcal{S}_L$:

$$\Delta\mathcal{L}^{\star\star}(\mathcal{S}_L \to k_L, \mathcal{S}_R \to k_R) = \Delta\mathcal{L}^{\star}(\mathcal{T}_p : k_p \to k_L) + \Delta\mathcal{L}^{\star}(\mathcal{S}_R : k_L \to k_R). \tag{5.32}$$

5.3.4.4 Merging with another cluster: the *switch gain*

This type of split is very similar to the creation of a new one. The main difference is that as one of the child nodes will join another cluster, *e.g.* $k_p \neq k_L \leq K$, we must take into account in the gain that we must subtract the *kernel stock* of the former target cluster. We call this type of gain the *switch gain*:

$$\Delta\mathcal{L}^{\rightleftarrows}(\mathcal{S}_L : k_p \to k_L) = \frac{\sigma({\mathcal{C}'_{k_L}}^2)}{\left|\mathcal{C}'_{k_L}\right|} + \frac{\sigma({\mathcal{C}'_{k_p}}^2)}{\left|\mathcal{C}'_{k_p}\right|} - \frac{\sigma(\mathcal{C}_{k_L}^2)}{|\mathcal{C}_{k_L}|} - \frac{\sigma(\mathcal{C}_{k_p}^2)}{\left|\mathcal{C}_{k_p}\right|}, \tag{5.33}$$

where we arbitrarily chose the left split for the equation. Similarly to the new cluster case, this expression can be completely re-written using only the original clusters and $\mathcal{S}_L$ to remove dependencies in the equation. We start by exploiting the bilinearity of σ. The first new cluster is the source one without the split elements and the second new cluster is the target one with added split elements. Therefore, we have $\mathcal{C}'_{k_p} = \mathcal{C}_{k_p} \setminus \mathcal{S}_L$ and $\mathcal{C}'_{k_L} = \mathcal{C}_{k_L} \cup \mathcal{S}_L$. We can deduce:

$$\Delta\mathcal{L}^{\rightleftarrows}(\mathcal{S}_L : k_p \to k_L) = \frac{\sigma(\mathcal{C}_{k_p}^2) - 2\sigma(\mathcal{C}_{k_p} \times \mathcal{S}_L) + \sigma(\mathcal{S}_L^2)}{\left|\mathcal{C}_{k_p}\right| - |\mathcal{S}_L|} + \frac{\sigma(\mathcal{C}_{k_L}^2) + 2\sigma(\mathcal{C}_{k_L} \times \mathcal{S}_L) + \sigma(\mathcal{S}_L^2)}{|\mathcal{C}_{k_L}| + |\mathcal{S}_L|} - \frac{\sigma(\mathcal{C}_{k_p}^2)}{\left|\mathcal{C}_{k_p}\right|} - \frac{\sigma(\mathcal{C}_{k_L}^2)}{|\mathcal{C}_{k_L}|}. \tag{5.34}$$

Then we finish again the demonstration by reordering the factors according to the respective stocks:

$$\Delta\mathcal{L}^{\rightleftarrows}(\mathcal{S}_L : k_p \to k_L) = \sigma(\mathcal{S}_L^2)\left(\frac{1}{|\mathcal{C}_{k_L}| + |\mathcal{S}_L|} + \frac{1}{\left|\mathcal{C}_{k_p}\right| - |\mathcal{S}_L|}\right) - 2\frac{\sigma(\mathcal{C}_{k_p} \times \mathcal{S}_L)}{\left|\mathcal{C}_{k_p}\right| - |\mathcal{S}_L|}$$
$$+ \sigma(\mathcal{C}_{k_p}^2)\left(\frac{1}{\left|\mathcal{C}_{k_p}\right| - |\mathcal{S}_L|} - \frac{1}{\left|\mathcal{C}_{k_p}\right|}\right) + \sigma(\mathcal{C}_{k_L}^2)\left(\frac{1}{|\mathcal{C}_{k_L}| + |\mathcal{S}_L|} - \frac{1}{\left|\mathcal{C}_{k_p}\right|}\right) + 2\frac{\sigma(\mathcal{C}_{k_L} \times \mathcal{S}_L)}{|\mathcal{C}_{k_L}| + |\mathcal{S}_L|}. \tag{5.35}$$

which is the equation we use in Kauri switch splits.

5.3.4.5 Reallocating content to different clusters: the *reallocation gain*

As the tree grows, it may as well be interesting to reconsider whether samples that are currently in a cluster should all be in a new cluster. We call this process *reallocation* as both splits of a node end up in two different clusters: $k_L \neq k_p$ and $k_R \neq k_p$.

Contrary to the double cluster creation case, we cannot simply sum two *switch gains* $\Delta\mathcal{L}^{\rightleftarrows}$ to compute the *reallocation gain*. Indeed, the *switch gain* assumes that the final state of the original cluster $\mathcal{C}_{k_p}$ still contains the complementary of the chosen split $\mathcal{S}_L$ (or $\mathcal{S}_R$) from the leaf samples $\mathcal{T}_p$ which is not true when both parts of the leaf go to different clusters. Hence, a corrective term ϵ is required.

When we sum two switch gains, the final state of the target clusters is correct: we simply added elements from a split. The corrective term thus only focuses on the state of the source cluster. Let $\mathcal{C}'_k$ the state of the source cluster according to the first switch gain on the left split, $\mathcal{C}''_k$ the state of the source cluster according to the second switch gain on the right split and $\mathcal{C}_k^{\hookleftarrow}$ the true state after reallocating both left and right splits. Notice that we lighten the notation k_p to k. The corrective term must satisfy:

$$\frac{\sigma({\mathcal{C}'_k}^2)}{|\mathcal{C}'_k|} + \frac{\sigma(\mathcal{C}''^2_k)}{|\mathcal{C}''_k|} + \epsilon = \frac{\sigma(\mathcal{C}_k^{\hookleftarrow 2})}{|\mathcal{C}_k^{\hookleftarrow}|}. \tag{5.36}$$

We can rewrite each new definition of the source clusters using the left split $\mathcal{S}_L$ and right split $\mathcal{S}_R$. Thus we get:

$$\frac{\sigma(\mathcal{C}_k \setminus {\mathcal{S}_L}^2)}{|\mathcal{C}_k| - |\mathcal{S}_L|} + \frac{\sigma(\mathcal{C}_k \setminus {\mathcal{S}_R}^2)}{|\mathcal{C}_k| - |\mathcal{S}_R|} + \epsilon = \frac{\sigma(\mathcal{C}_k \setminus {\mathcal{T}_p}^2)}{|\mathcal{C}_k| - |\mathcal{T}_p|}, \tag{5.37}$$

which allows us to use the bilinearity of σ:

$$\frac{\sigma(\mathcal{C}_k^2) - 2\sigma(\mathcal{C}_k \times \mathcal{S}_L) + \sigma(\mathcal{S}_L^2)}{|\mathcal{C}_k| - |\mathcal{S}_L|} + \frac{\sigma(\mathcal{C}_k^2) - 2\sigma(\mathcal{C}_k \times \mathcal{S}_R) + \sigma(\mathcal{S}_R^2)}{|\mathcal{C}_k| - |\mathcal{S}_R|} + \epsilon$$
$$= \frac{\sigma(\mathcal{C}_k^2) - 2\sigma(\mathcal{C}_k \times \mathcal{T}_p) + \sigma({\mathcal{T}_p}^2)}{|\mathcal{C}_k| - |\mathcal{T}_p|}. \tag{5.38}$$

Then, by reordering the terms and simplifying for the factor $\sigma(\mathcal{C}_k^2)$, we get the expression of ϵ:

$$\epsilon = \frac{\sigma(\mathcal{C}_{k_p}^2) + \sigma(\mathcal{T}_p^2) - 2\sigma(\mathcal{C}_{k_p} \times \mathcal{T})}{\left|\mathcal{C}_{k_p}\right| - |\mathcal{T}_p|} + \frac{\sigma(\mathcal{C}_{k_p}^2)}{\left|\mathcal{C}_{k_p}\right|} - \frac{\sigma(\mathcal{C}_{k_p}^2) + \sigma(\mathcal{S}_L^2) - 2\sigma(\mathcal{C}_{k_p} \times \mathcal{S}_L)}{\left|\mathcal{C}_{k_p}\right| - |\mathcal{S}_L|} - \frac{\sigma(\mathcal{C}_{k_p}^2) + \sigma(\mathcal{S}_R^2) - 2\sigma(\mathcal{C}_{k_p} \times \mathcal{S}_R)}{\left|\mathcal{C}_{k_p}\right| - |\mathcal{S}_R|}. \quad (5.39)$$

Thus, we can express the *reallocation gain* $\Delta\mathcal{L}^{\hookrightarrow}$ as the sum of two switch gains assigning both left and right children nodes to different clusters plus the corrective term ϵ.

$$\Delta\mathcal{L}^{\hookrightarrow}(\mathcal{S}_L : k_p \to k_L, \mathcal{S}_R : k_p \to k_R) = \Delta\mathcal{L}^{\rightleftarrows}(\mathcal{S}_L : k_p \to k_L) + \Delta\mathcal{L}^{\rightleftarrows}(\mathcal{S}_R : k_p \to k_R) + \epsilon. \quad (5.40)$$

5.4 A fast implementation for Kauri

Upon looking at one leaf containing a small subset of samples, we need to find the best possible split according to a given threshold on a specified feature. As each feature specifies a different ordering and offers little space for optimisation, computing all possible gains may be time-consuming. Indeed, computing $\sigma(E \times F)$ is done in $\mathcal{O}(|E||F|)$, so evaluating gains for a proposal split $\mathcal{S}$ on a single feature for node samples $\mathcal{T}_p$ contributing to a cluster of $\mathcal{C}_k$ has a naive complexity of: $\mathcal{O}(|\mathcal{S}|^2 + |\mathcal{C}_k|^2 + |\mathcal{C}_k||\mathcal{S}|)$ for $\Delta\mathcal{L}^{\star}$, $\mathcal{O}(|\mathcal{S}|^2 + |\mathcal{C}_k|^2 + |\mathcal{T}_p|^2 + |\mathcal{C}_k|(|\mathcal{S}| + |\mathcal{T}_p|))$ for $\Delta\mathcal{L}^{\star\star}$, $\mathcal{O}(|\mathcal{T}_p|^2 + |\mathcal{C}_k|^2 + \mathcal{C}_k||\mathcal{T}_p| + |\mathcal{C}_{k'}|^2 + |\mathcal{C}_{k'}||\mathcal{T}_p|)$ for each $k' \neq k$ for $\Delta\mathcal{L}^{\rightleftarrows}$, and at worst K times the previous complexity again for all pairs of assignable new clusters k', k'' in the *reallocation gain* $\Delta\mathcal{L}^{\hookrightarrow}$. Therefore, iterating over all features and all possible splits needs to be optimised as this operation is the core of the tree construction.

5.4.1 Pre-computing kernel stocks

Most of the kernel stocks can be computed ahead in fact, and then the splitting choice would just need to access the value of the kernel stocks instead. To that end, we choose to formulate two matrices that will store all structural information. The matrix $\boldsymbol{Z} \in \{0,1\}^{T_{\max} \times n}$ describes the membership of samples to leaves where $T_{\max}$ is the maximal number of leaves allowed ($T_{\max} \leq n$). As a sample can only belong to 1 leaf, each column of $\boldsymbol{Z}$ has a single 1. Similarly, the matrix $\boldsymbol{Y} \in \{0,1\}^{K_{\max} \times T_{\max}}$ describes the membership of leaves to clusters, and only one cluster is allowed per leaf. We can then compute most of the kernel stocks required for split computations, as:

$$\boldsymbol{\Lambda} = [\sigma(\mathcal{T}_i \times \{\boldsymbol{x}_j\})] = \boldsymbol{Z}\boldsymbol{\kappa}, \quad (5.41)$$

is the matrix containing all stocks between leaves and single samples requiring $\mathcal{O}(n^2 T_{\max})$ to compute, and:

$$\boldsymbol{\gamma} = [\sigma(\mathcal{C}_i \times \mathcal{C}_j)] = \boldsymbol{Y}\boldsymbol{\Lambda}\boldsymbol{Z}^\top\boldsymbol{Y}^\top, \quad (5.42)$$

is the matrix with cluster-cluster stocks, requiring $\mathcal{O}(n^2 T_{\max} + T_{\max}^2 K_{\max})$ for computations.

Algorithm 1 Computes the best possible type of split given kernel stocks of a currently split node

Require: $\sigma(\mathcal{S}_L^2)$ the stock of the left split
Require: $|\mathcal{S}_L|$ the size of the left split
Require: $\sigma(\mathcal{S}_R^2)$ the stock of the right split
Require: $|\mathcal{S}_R|$ the size of the right split
Require: $\sigma(\mathcal{C}_k) \forall k$ the stock of all clusters
Require: $|\mathcal{C}_k| \forall k$ the size of all clusters
Require: $\sigma(\mathcal{C}_k \times \mathcal{S}_L) \forall k$ the adversarial stocks of the left split
Require: $\sigma(\mathcal{C}_k \times \mathcal{S}_R) \forall k$ the adversarial stocks of the right split
Require: $\sigma(\mathcal{T}^2)$ the leaf stock
Require: k the indicator of the current cluster of the split leaf
Require: K the current number of clusters

function COMPUTESPLITS($\sigma(\mathcal{S}_L^2)$, $|\mathcal{S}_L|$, $\sigma(\mathcal{S}_R^2)$, $|\mathcal{S}_R|$, $\{\sigma(\mathcal{C}_k)\}$, $\{|C_k|\}$, $\{\sigma(\mathcal{C}_k \times \mathcal{S}_L)\}$, $\{\sigma(\mathcal{C}_k \times \mathcal{S}_R)\}$, $\sigma(\mathcal{T}^2)$, k, K)
    $\Delta\mathcal{L} \leftarrow 0$
    `Split` $\leftarrow (-1, -1)$ ▷ Stores the cluster targets of left and right splits
    if Creating a new cluster is allowed **then**
        Compute $\Delta\mathcal{L}_L^\star$ using the left split $\mathcal{S}_L$ and Eq. 5.30
        Compute $\Delta\mathcal{L}_R^\star$ using the right split $\mathcal{S}_R$ and Eq. 5.30
        $\Delta\mathcal{L}$, `Split` $\leftarrow$ TAKEBEST($\{\Delta\mathcal{L}$, `Split`$\}$, $\{\Delta\mathcal{L}_L^\star, (K+1, k)\}$, $\{\Delta\mathcal{L}_R^\star, (k, K+1)\}$)
    end if
    if Creating two clusters is allowed **then**
        Compute $\Delta\mathcal{L}^{\star\star}$ using either $\mathcal{S}_L$ or $\mathcal{S}_R$ and Eq. 5.32
        $\Delta\mathcal{L}$, `Split` $\leftarrow$ TAKEBEST($\{\Delta\mathcal{L}$, `Split`$\}$,$\{\Delta\mathcal{L}^{\star\star}, (K+1, K+2)\}$)
    end if
    `TopL`, `SecondL`, `TopR`, `SecondR` $\leftarrow$ 0,0,0,0
    `TopkL`, `SecondkL`, `TopkR`, `SecondkR` $\leftarrow$ -1,-1,-1,-1
    for $k' \in \{1, \cdots, K\} \setminus \{k\}$ **do**
        Compute $\Delta\mathcal{L}_L^{\rightleftarrows}$ using the left split $\mathcal{S}_L$ and Eq. 5.35
        Compute $\Delta\mathcal{L}_R^{\rightleftarrows}$ using the right split $\mathcal{S}_L$ and Eq. 5.35
        $\Delta\mathcal{L}$, `Split` $\leftarrow$ TAKEBEST($\{\Delta\mathcal{L}$, `Split`$\}$, $\{\Delta\mathcal{L}_L^{\rightleftarrows}, (k', k)\}$, $\{\Delta\mathcal{L}_R^{\rightleftarrows}, (k, k')\}$)
        Update `TopL`, `SecondL`, `TopkL` and `SecondkL` using $\Delta\mathcal{L}_L^{\rightleftarrows}$ to keep track of the two best switch gains and their respective target cluster k'
        Update `TopR`, `SecondR`, `TopkR` and `SecondkR` using $\Delta\mathcal{L}_R^{\rightleftarrows}$ to keep track of the two best switch gains and their respective target cluster k'
    end for
    if Reallocation is allowed **then**
        Compute ϵ using Eq. 5.39
        Compute $\Delta\mathcal{L}^{\hookrightarrow}$ using `TopL`, `SecondL`, `TopkL`, `SecondkL`, `TopR`, `SecondR`, `TopkR`, `SecondkR` and ϵ.
        $\Delta\mathcal{L}$, `Split` $\leftarrow$ TAKEBEST($\{\Delta\mathcal{L}$, `Split`$\}$, $\{\Delta\mathcal{L}^{\hookrightarrow}$, (target left, target right)$\}$)
    end if
    return $\Delta\mathcal{L}$, `Split`
end function

5.4.2 Optimising split evaluation

Thanks to the formulation of the *star gain* $\Delta\mathcal{L}^{\star}$, the *double star gain* $\Delta\mathcal{L}^{\star\star}$ and assuming we know the variables $\sigma(\mathcal{S}_L \times \mathcal{S}_L)$ (resp. $\mathcal{S}_R$) and $\sigma(\mathcal{S}_L \times \mathcal{C}_k)$ (resp. $\mathcal{S}_R$), evaluating the creation of clusters is done in $\mathcal{O}(1)$. Inevitably, we achieve $\mathcal{O}(K)$ for the switch gains $\Delta\mathcal{L}^{\rightleftarrows}$ since evaluating these gains is easy but needs iteration over all clusters.

To alleviate the complexity of the *reallocation gain* $\Delta\mathcal{L}^{\hookrightarrow}$ due to the exploration of all pairs of clusters, we propose to remember the top two switch gains per left children and right children. Indeed, the corrective term ϵ (Eq. 5.39 does not depend on the two clusters to which the left and right children will be reallocated. Hence maximising the *reallocation gain* is the same as finding the combination of the best switch gains. Thus, remembering the top two switch gains and finding the best combination between left and right child, with different clusters membership per child, will yield the optimal *reallocation gain*. Therefore we achieved the best gain in $\mathcal{O}(K)$ and the evaluation of all types of gain is done in $\mathcal{O}(K)$. This search is summarised in Algorithm 1.

5.4.3 An iterative rule for split stocks

Starting from here, we seek an update rule that allows us to easily update the kernel stocks of the splits $\sigma(\mathcal{S} \times \mathcal{C}_k)$ and $\sigma(\mathcal{S}^2)$ that are required by Algorithm 1. We must explore all possible splits by considering thresholds on chosen variables. Therefore, a split on a variable must be done according to the ordering imposed by that variable, leaving a left child $\mathcal{S}_L^{(l)}$ and a right child $\mathcal{S}_R^{(l)}$ at the l-th threshold. The algorithm consist in starting from the split of a single sample to the left child $\mathcal{S}_L^{(1)}$ and all other samples to the right child $\mathcal{S}_R^{(1)}$ then progressively remove or add samples according to an ordering given by a sorted feature: $t = \nu(l)$ to compute $\mathcal{S}_L^{(l)}$ and $\mathcal{S}_R^{(l)}$. For example, if the p-th node has the data samples 5, 8, 9 and 15, a feature may order those as 9,8,15,5. Then, $\nu(1) = 9$, $\nu(2) = 8$, $\nu(3) = 15$ and $\nu(4) = 5$.

Note that there is a key difference regarding the indices notations. We write i the absolute index of sample from the dataset $\mathcal{D}$ while l refers to the ordered count of samples inside a specific node. The index t is the absolute index according to the ordering $\nu(l)$.

We introduce 3 helping variables. The first one is the sample-wise cluster adversarial stocks:

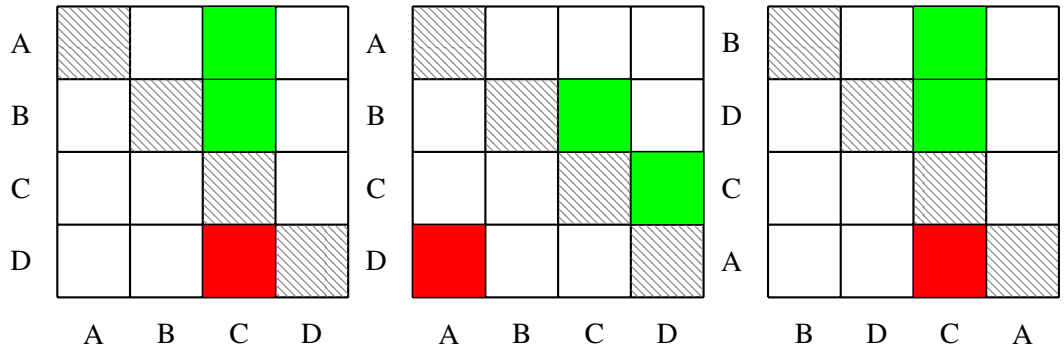
$$\boldsymbol{\omega}_{k,i} = \sigma(\mathcal{C}_k \times \{\boldsymbol{x}_t\}), \tag{5.43}$$

which does not depend on the ordering specified by a feature and will ease the computation of both $\sigma(\mathcal{S}_L \times \mathcal{C}_k)$ and $\sigma(\mathcal{S}_R \times \mathcal{C}_k)$. We can shortly write that $\boldsymbol{\omega} = \boldsymbol{Y}\boldsymbol{Z}\boldsymbol{\kappa}$. To alleviate the computations of $\sigma(\mathcal{S}_L^2)$ and $\sigma(\mathcal{S}_R^2)$, we introduce the ordering-dependent variables:

$$\alpha_t^{\nu} = \sum_{\substack{l=1\cdots|\mathcal{T}_p| \\ \nu(l)<t}} \kappa_{\nu(l),t}, \tag{5.44}$$

and:

$$\beta_t^{\nu} = \sum_{\substack{l=1\cdots|\mathcal{T}_p| \\ \nu(l)>t}} \kappa_{\nu(l),t}. \tag{5.45}$$



(a) There is no ordering of the samples (b) There exists an ordering ν (c) Reordering according to ν

Figure 5.4 – An example of the value of the variables α_3 and β_3 which respectively are the sum of green squares and red squares on the kernel matrix of the elements A, B, C and D in a leaf. In 5.4b and 5.4c, the ordering is $\nu(\{1,2,3,4\}) = \{B, D, C, A\}$.

These two variables verify a constant sum $\beta_t^\nu + \alpha_t^\nu + \kappa_{tt} = \sigma(\{x_t\} \times \mathcal{T}_p)$ for all $t \in \mathcal{T}_p$. We provide a visual intuition of the definition of these variables in Fig. 5.4. Once the variables $\boldsymbol{\omega}_{k,i}$, α_t^ν, β_t^ν are initialised, we can compute all split gains with simple additions.

The initialisation of the variables is easy. For the split self-stock, we have:

$$\sigma(\mathcal{S}_L^{(0)} \times \mathcal{S}_L^{(0)}) = 0, \tag{5.46}$$

$$\sigma(\mathcal{S}_R^{(0)} \times \mathcal{S}_L^{(0)}) = \sigma(\mathcal{T}_p^2), \tag{5.47}$$

because starting from no sample yields all content of the node $\mathcal{T}_p$ to the right split $\mathcal{S}_R^{(0)}$. The adversarial stocks follow the same logic:

$$\sigma(\mathcal{S}_L^{(0)} \times \mathcal{C}_k) = 0, \tag{5.48}$$

$$\sigma(\mathcal{S}_R^{(0)} \times \mathcal{C}_k) = \sigma(\mathcal{T}_p \times \mathcal{C}_k), \tag{5.49}$$

where the last term is simply an element of $\boldsymbol{\gamma}$ indexed by the respective leaf and cluster. The iterations then consist in removing adequate adversarial stock or self-kernel stock:

$$\sigma(\mathcal{S}_L^{(l)} \times \mathcal{S}_L^{(l)}) = \sigma(\mathcal{S}_L^{(l-1)} \times \mathcal{S}_L^{(l-1)}) + 2\alpha_{\nu(l)}^\nu + \kappa_{\nu(l),\nu(l)}, \tag{5.50}$$

$$\sigma(\mathcal{S}_R^{(l)} \times \mathcal{S}_R^{(l)}) = \sigma(\mathcal{S}_R^{(l-1)} \times \mathcal{S}_R^{(l-1)}) - 2\beta_{\nu(l)}^\nu - \kappa_{\nu(l),\nu(l)}. \tag{5.51}$$

The adversarial scores are easier to update:

$$\sigma(\mathcal{S}_L^{(l)} \times \mathcal{C}_k) = \sigma(\mathcal{S}_L^{(l-1)} \times \mathcal{C}_k) + \boldsymbol{\omega}_{k,\nu(l)}, \tag{5.52}$$

and conversely:

$$\sigma(\mathcal{S}_R^{(l)} \times \mathcal{C}_k) = \sigma(\mathcal{S}_R^{(l-1)} \times \mathcal{C}_k) - \boldsymbol{\omega}_{k,\nu(l)}. \tag{5.53}$$

Thanks to these iterative variables, the iterative computation of all kernel stocks can be achieved in $\mathcal{O}(|\mathcal{T}_p|(|\mathcal{T}_p| + K))$ for a specific feature and node samples $\mathcal{T}_p$ as summarised in Algorithm 2. The pre-computing of $\boldsymbol{\omega}$ takes $\mathcal{O}(n^2)$ and can be done in advance at the tree level.

Algorithm 2 Finding the best split according a feature-specified ordering ν at a given leaf $\mathcal{T}$

Require: $\mathcal{T}$ the set of samples in the leaf of length $|\mathcal{T}|$
Require: p the index of the leaf
Require: ν an ordering precised by a feature of length $|\mathcal{T}|$
Require: $\boldsymbol{\kappa}$ a kernel of shape $n \times n$
Require: $\boldsymbol{\Lambda}$ the $T_{\max} \times n$ leaf-sample stocks
Require: $\boldsymbol{\omega}$ the $K_{\max} \times n$ cluster-sample stocks
Require: $\boldsymbol{\gamma}$ the $K_{\max} \times K_{\max}$ cluster-cluster stocks
Require: $|\mathcal{C}_k|, \forall k$ the size of all clusters
Require: k the cluster of the considered leaf

function FINDBESTSPLIT($\mathcal{T}$, ν, $\boldsymbol{\kappa}$, Λ, $\boldsymbol{\omega}$, $\boldsymbol{\gamma}$, $\{|\mathcal{C}_k|\}$, k)
  $\sigma(\mathcal{S}_L^{(0)} \times \mathcal{S}_L^{(0)}) \leftarrow 0$ ▷ Initialise all iteration variables
  $\sigma(\mathcal{S}_R^{(0)} \times \mathcal{S}_R^{(0)}) \leftarrow \sum_{\boldsymbol{x}_i \in \mathcal{T}} \boldsymbol{\Lambda}_{pi}$
  $\sigma(\mathcal{S}_L^{(0)} \times \mathcal{C}_k) \leftarrow 0, \forall k \leq K_{\max}$ ▷ Arrays of size $K_{\max}$
  $\sigma(\mathcal{S}_R^{(0)} \times \mathcal{C}_k) \leftarrow \sum_{\boldsymbol{x}_i \in \mathcal{T}} \boldsymbol{\omega}_{ki}, \forall k \leq K_{\max}$
  $\sigma(\mathcal{T} \times \mathcal{T}) \leftarrow \sigma(\mathcal{S}_R^{(0)} \times \mathcal{S}_R^{(0)})$
  $K \leftarrow |\{k \text{ s.t. } |\mathcal{C}_k| \neq 0\}|$ ▷ Current number of clusters
  $\Delta\mathcal{L}$, `BestSplit` $\leftarrow 0, \emptyset$ ▷ Best split so far
  for $l \leftarrow 1$ to $|\mathcal{T}| - 1$ **do**
    $\alpha, \beta \leftarrow 0,0$
    for $l' \leftarrow 1$ to $|\mathcal{T}|$ **do**
      if $l' < l$ **then**
        $\alpha \leftarrow \alpha + \kappa_{\nu(l),\nu(l')}$
      end if
      if $l' > l$ **then**
        $\beta \leftarrow \beta + \kappa_{\nu(l),\nu(l')}$
      end if
    end for
    Update $\sigma(\mathcal{S}_L^{(l)} \times \mathcal{S}_L^{(l)})$, $\sigma(\mathcal{S}_R^{(l)} \times \mathcal{S}_R^{(l)})$, $\sigma(\mathcal{S}_L^{(l)} \times \mathcal{C}_k)$ and $\sigma(\mathcal{S}_R^{(l)} \times \mathcal{C}_k)$ using equations 5.50, 5.51, 5.52, 5.53.
    $\tilde{\Delta\mathcal{L}}$, `split` $\leftarrow$ COMPUTESPLITS($\sigma(\mathcal{S}_L^{(l)} \times \mathcal{S}_L^{(l)})$, l, $\sigma(\mathcal{S}_R^{(l)} \times \mathcal{S}_R^{(l)})$, $|\mathcal{T}|-1-l$, diag($\boldsymbol{\gamma}$), $|C_k|$, $\sigma(\mathcal{S}_L^{(l)} \times \mathcal{C}_k)$, $\sigma(\mathcal{S}_R^{(l)} \times \mathcal{C}_k)$, $\sigma(\mathcal{T} \times \mathcal{T})$, k, K)
    if $\tilde{\Delta\mathcal{L}} > \Delta\mathcal{L}$ **then**
      $\Delta\mathcal{L} \leftarrow \tilde{\Delta\mathcal{L}}$
      `BestSplit` $\leftarrow$ `split` $\cup (\nu(l))$ ▷ The split gives the left target, the right target, the sample on which the split is done
    end if
  end for
  return $\Delta\mathcal{L}$, `BestSplit`
end function

```python
from sklearn import datasets
from gemclus.tree import Kauri, print_kauri_tree
# Load a dataset
iris = datasets.load_iris()

# Train Kauri with a specific kernel function
model = Kauri(max_clusters=3, kernel="linear", max_depth=3).
    fit(iris["data"])

# Print the tree using the feature names of the dataset in the
    rules
print_kauri_tree(model, iris["feature_names"])
```

Listing 5.1 – An example of Kauri in GemClus

Finally, we can optimise the computation of all splits.

5.4.4 Complete picture

The complete algorithm of Kauri is written in Algorithm 3. We estimate the complexity of the split search from line 14 to line 25 to $\mathcal{O}(dn(K+n))$ assuming to use a structure with $\mathcal{O}(n)$ insertion complexity and $\mathcal{O}(1)$ access complexity.

5.4.5 Implementation in GemClus

Kauri is implemented in GemClus. We give an example of usage in the Listing 5.1.

5.5 Douglas: DNDTs optimised using GEMINI leverage apprised splits

5.5.1 The Douglas model

The Douglas model seeks the full potential of GEMINI by combining it with differentiable trees, *e.g.* deep neural trees (C. Yang, Ojha, Aranoff, Green, & Tavassolian, 2020). Thanks to this choice of architecture, we can optimise Wasserstein-GEMINI, an objective more efficient for clustering than the MMD-GEMINI, with respect to the parameters through gradient descent. Indeed, the MMD-GEMINI only carries information through the means of cluster distributions and does not encompass all information on the data space whereas the expected Wasserstein distance between two randomly chosen clusters will take into account the complete distribution. However, the cost of Douglas is the loss of depth in tree as all rules are produced at the root level.

Deep neural decision trees (DNDTs, Y. Yang et al., 2018) aim at learning individual rules per feature and then merge those rules to provide a final decision. Formally, each feature f among a subset of selected features is assigned a vector of T sorted thresholds $b^f_{1\dots T}$ called cut-points that determines the binnings of the feature. By defining a bias $\boldsymbol{c}^f = [0, -b^f_1, -b^f_1 - b^f_2, \cdots, -b^f_1 - b^f_2 -$

Algorithm 3 Training the complete Kauri tree

Require: $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$ a dataset, $\boldsymbol{x}_i \in \mathbb{R}^d$
Require: $K_{\max} >= 2$ the maximum number of allowed clusters
Require: $d_{\max} \in \{1, d\}$ the maximum number of feature to consider per split.
Require: $T_{\max} \leq n$ the maximum number of leaves

function TRAINKAURI($\mathcal{D}$, $K_{\max}$, $d_{\max}$, $T_{\max}$)
  $\boldsymbol{\kappa} \leftarrow \langle \varphi(\mathcal{D}), \varphi(\mathcal{D}) \rangle$ ▷ Kernel value of samples, $n \times n$
  Initialise $\boldsymbol{Z}$ and $\boldsymbol{Y}$ ▷ All samples in one leaf, leaf belongs to only one cluster
  Initialise the tree structure in `Tree`.
  `Leaves` $\leftarrow$ `List`(0) ▷ Only one starting leaf to explore
  $\Delta\mathcal{L} \leftarrow \infty$ ▷ Last gain value
  `BestSplit` $\leftarrow \emptyset$ ▷ The best split proposal
  while `Leaves` $\neq \emptyset \wedge |$`Tree`$| \leq T_{\max} \wedge \bar{\Delta\mathcal{L}} > 0$ **do**
    $\Delta\mathcal{L} \leftarrow 0$ ▷ Best split achieved so far
    $\boldsymbol{\Lambda} \leftarrow \boldsymbol{Z}\boldsymbol{\kappa}$ ▷ Compute $\sigma(\mathcal{T}_p \times \{\boldsymbol{x}_j\})$
    $\boldsymbol{\omega} \leftarrow \boldsymbol{Y}\boldsymbol{\Lambda}$ ▷ Compute $\sigma(\mathcal{C}_k \times \{\boldsymbol{x}_j\})$
    $\boldsymbol{\gamma} \leftarrow \boldsymbol{\omega}\boldsymbol{Z}^\top\boldsymbol{Y}^\top$ ▷ Compute $\sigma(\mathcal{C}_k \times \mathcal{C}_{k'})$
    $|\mathcal{C}_k| \leftarrow \boldsymbol{Y}\boldsymbol{Z}\boldsymbol{1}_n$ ▷ Sizes of clusters
    for $p \in$ `Leaves` **do**
      $\mathcal{T}_p \leftarrow \{\boldsymbol{x}_i | \boldsymbol{Z}_{ji} == 1\}$ ▷ Find the indices of leaf p
      $k \leftarrow \text{argmax}_{k'} \boldsymbol{Y}_{k',p}$ ▷ The current cluster of leaf p
      for $f \leftarrow 1$ to $d_{\max}$ **do**
        $\nu \leftarrow$ Argsort($\{\boldsymbol{x}_{if} | \boldsymbol{x}_i \in \mathcal{T}_p\}$)
        $\tilde{\Delta\mathcal{L}}$, `split` $\leftarrow$ FINDBESTSPLIT($\mathcal{T}$, j, ν, $\boldsymbol{\kappa}$, Λ, $\boldsymbol{\omega}$, $\boldsymbol{\gamma}$, $|\mathcal{C}_k|$, k)
        if $\tilde{\Delta\mathcal{L}} > \Delta\mathcal{L}$ **then**
          $\Delta\mathcal{L} \leftarrow \tilde{\Delta\mathcal{L}}$
          `BestSplit` $\leftarrow$ `split` $\cup (p, f)$ ▷ Add node and feature information to the best split
        end if
      end for
    end for
    if $\Delta\mathcal{L} > 0$ **then**
      Remove the best leaf from the list `Leaves` and add the children of the split in `Leaves` if they satisfy structural constraints.
      Update `Tree` using `BestSplit`
      Update $\boldsymbol{Z}$ and $\boldsymbol{Y}$.
    end if
  end while
  return `Tree`
end function

```python
from sklearn import datasets
from gemclus.tree import Douglas
# Load a dataset
X, _ = datasets.load_iris(return_X_y=True)

# Train Douglas with OvA Wasserstein-GEMINI
y_pred =Douglas(n_clusters=3, gemini="wasserstein_ova",
    max_iter=100, n_cuts=1)
```

Listing 5.2 – An example of Douglas in GemClus

$\cdots - b_T^f]$ and a vector $\boldsymbol{a}^f = [0, 1, \cdots, T]$, Y. Yang et al. (2018) write a feature-wise probability distribution with:

$$p_{\boldsymbol{a}^f, \boldsymbol{c}^f}(\beta | \boldsymbol{x}^f) = \text{SoftMax}\left(\frac{\boldsymbol{a}^f \boldsymbol{x}^f + \boldsymbol{c}^f}{\tau}\right), \tag{5.54}$$

named soft-binning where τ is a temperature hyperparameter set to 0.1. After each individual soft binning is applied, all combinations of features are computed using a Kronecker product. For example, if the d features are all separated in $T + 1$ binnings, the final decision will contain $(T + 1)^d$ entries per sample. To produce a decision from this entry, a matrix multiplication with some parameters $\boldsymbol{W}$ is applied. The global model can be described as:

$$p_\theta(y = k | \boldsymbol{x}) = \sum_{t_1=1}^{T} \sum_{t_2=1}^{T} \cdots \sum_{t_d=1}^{T} W_{k, t_1 + d t_2 + \cdots + d^{d-1} t_d} \prod_{f=1}^{d} p_{\boldsymbol{a}^f, \boldsymbol{c}^f}(\beta = t_f | \boldsymbol{x}^f). \tag{5.55}$$

This model is therefore differentiable and can be trained by gradient descent. However, DNDTs are hardly scalable in terms of features because of the Kronecker product between all soft binnings.

For interpretation purposes, we choose to exploit active cut points as proposed by Y. Yang et al. (2018). This is the number of features for which the respective cut points parameters do not lie outside of the feature boundaries in the dataset. For example, if for a single cut value (two bins) the bias b_1^f is lower or greater than all samples on its respective feature f, then this cut point is not active and does not participate in the decision because all samples are in the same bin.

5.5.2 Implementation in GemClus

Douglas is implemented in GemClus. We give an example of usage in the Listing 5.2.

5.6 Experiments

We start by proposing a summary of the advantages and limitations of both tree algorithms in Table 5.1. Overall, Kauri is recommended for small-scale datasets whereas Douglas can be used with large datasets on condition that there are few features.

We tried to cover datasets used for benchmarking by Laber et al. (2023), Frost, Moshkovitz, et Rashtchian (2020) and those from the Fundamental Clustering Data Suite (Thrun & Stier, 2021) as

Table 5.1 – Advantages and disadvantages of the Kauri and Douglas algorithms for unsupervised tree construction. The number of samples is noted n and the number of features d.

	Splits	Scalable with n	Scalable with d	Hyperparameters
Kauri	Binary	No	Yes	$K_{\max}, T_{\max}$
Douglas	k-ary	Yes with minibatches	No	Number of cut-points T

Table 5.2 – Summary of the datasets used in the experiments. *The number of features may be slightly larger than the actual number of variables as discrete variables were one-hot encoded.

Name	Samples	Features	Classes
Atom	800	3	2
Avila	20,867	10	12
Breast Cancer	683	9	2
Car evaluation*	1,728	21	4
Chainlink	1,000	3	2
US Congress	435	16	2
Engytime	4,096	2	2
Digits	1,797	64	10
Haberman Survival	306	3	2
Hepta	212	3	7
Iris	150	4	3
Lsun	404	3	3
Mice protein	552	77	8
Target	770	2	6
Tetra	400	3	4
Twodiamonds	800	2	2
Vowel	990	10	2
Wine	178	13	3
Wingnut	1,016	2	2

did Bertsimas et al. (2021). We will assess the general clustering performances and explanation power of the models before showing qualitative examples.

We used minmax scaling for all datasets in order to be comparable with the ICOT (Bertsimas et al., 2021) method which assumes features in the [0,1] range. All categorical variables were one-hot-encoded, except for the US congressional votes dataset, where we encoded specifically the answer yes as 1, the no as -1 and the unknown votes as 0 as we did in Section 4.4.5. In other datasets, we tossed away all samples that presented missing values for the sake of simplicity.

We used 4 different metrics for assessing the performances: the adjusted rand index (ARI, Hubert & Arabie, 1985) and the kernel K-means score normalised by the reference score of the sole kernel K-means algorithm for the clustering quality, then the weighted average depth (WAD, Laber et al., 2023) or weighted average explanation size (WAES, Laber et al., 2023) for the tree structure. The WAD score corresponds to the mean depth where a decision is returned for samples, *i.e.* for the set of leaves $\{\mathcal{T}_p\}$ we have:

$$\text{WAD}(\{\mathcal{T}_p\}_{p=1}^{T_{\max}}) = \frac{1}{n}\sum_{p=1}^{T_{\max}} \text{Depth}(\mathcal{T}_p) \times |\mathcal{T}_p|\,. \tag{5.56}$$

In the case of the WAES, we do not consider the depth of the nodeas weighting factor but the number of non-redundant rules. For instance, if the path leading to a leaf specifics $\boldsymbol{x}_{i1} \leq \tau_{p_1 1}$, $\boldsymbol{x}_{i2} \leq \tau_{p_2 2}$ and $\boldsymbol{x}_{i1} \leq \tau_{p_3 1}$, then only two rules are sufficient to cluster $\boldsymbol{x}_i$ because the first and third rules inspect the same feature.

Table 5.3 – ARI scores $_{\text{std}}$ (greater is better) after 30 runs on random subsamples of 80% of the input datasets. Entries marked X were not run because of excessive runtime due to large numbers of features. All models are limited to finding as many leaves as clusters.

Dataset	Kauri	K-means+DT	ICOT	IMM	ExShallow	RDM
Atom	$0.19_{0.03}$	$0.17_{0.03}$	$0.17_{0.05}$	$\mathbf{0.20_{0.04}}$	$0.19_{0.03}$	$0.17_{0.01}$
Avila	$0.02_{0.02}$	$0.04_{0.02}$	X	$\mathbf{0.05_{0.03}}$	$0.04_{0.02}$	$\mathbf{0.05_{0.03}}$
Cancer	$0.74_{0.01}$	$0.73_{0.01}$	$\mathbf{0.78_{0.05}}$	$0.73_{0.02}$	$0.73_{0.01}$	$0.65_{0.02}$
Car	$0.04_{0.06}$	$\mathbf{0.07_{0.08}}$	X	$0.06_{0.06}$	$0.06_{0.06}$	$\mathbf{0.07_{0.05}}$
Chainlink	$\mathbf{0.10_{0.01}}$	$0.09_{0.02}$	$0.08_{0.04}$	$\mathbf{0.10_{0.01}}$	$0.09_{0.01}$	$0.08_{0.01}$
Congress	$0.48_{0.02}$	$0.47_{0.04}$	$\mathbf{0.49_{0.03}}$	$0.47_{0.04}$	$0.48_{0.02}$	$0.39_{0.02}$
Digits	$0.40_{0.04}$	$0.42_{0.03}$	X	$0.36_{0.04}$	$\mathbf{0.43_{0.02}}$	$0.23_{0.04}$
Engytime	$\mathbf{0.51_{0.01}}$	$\mathbf{0.51_{0.01}}$	X	$0.50_{0.01}$	$0.49_{0.01}$	$\mathbf{0.51_{0.01}}$
Haberman	$\mathbf{0.00_{0.00}}$	$\mathbf{0.00_{0.00}}$	$\mathbf{0.00_{0.00}}$	$\mathbf{0.00_{0.00}}$	$\mathbf{0.00_{0.00}}$	$\mathbf{0.00_{0.00}}$
Hepta	$\mathbf{1.00_{0.01}}$	$\mathbf{1.00_{0.00}}$	$0.79_{0.16}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$0.93_{0.02}$
Iris	$\mathbf{0.79_{0.09}}$	$0.73_{0.04}$	$0.57_{0.02}$	$0.74_{0.06}$	$0.78_{0.05}$	$0.51_{0.03}$
Lsun	$0.89_{0.02}$	$\mathbf{0.91_{0.04}}$	$0.78_{0.20}$	$0.86_{0.04}$	$0.90_{0.05}$	$0.58_{0.10}$
Mice	$\mathbf{0.23_{0.02}}$	$0.20_{0.03}$	X	$0.17_{0.03}$	$0.21_{0.03}$	$0.12_{0.03}$
Target	$\mathbf{0.64_{0.01}}$	$0.56_{0.05}$	X	$0.63_{0.02}$	$\mathbf{0.64_{0.02}}$	$0.28_{0.02}$
Tetra	$0.94_{0.07}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$0.61_{0.03}$
Twodiamonds	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$0.98_{0.01}$
Vowel	$0.02_{0.02}$	$0.02_{0.02}$	X	$0.03_{0.03}$	$\mathbf{0.04_{0.06}}$	$\mathbf{0.04_{0.03}}$
Wine	$0.67_{0.08}$	$0.73_{0.04}$	$0.48_{0.16}$	$\mathbf{0.75_{0.03}}$	$\mathbf{0.75_{0.04}}$	$0.29_{0.09}$
Wingnut	$0.15_{0.01}$	$0.15_{0.01}$	$\mathbf{0.46_{0.38}}$	$0.15_{0.01}$	$0.15_{0.01}$	$0.13_{0.01}$

We compare the performances of Kauri against recent methods for unsupervised tree constructions, namely ExShallow (Laber et al., 2023), RDM (Makarychev & Shan, 2022), IMM (Moshkovitz et al., 2020), ExKMC (Frost et al., 2020) and ICOT (Bertsimas et al., 2021) for which we found available packages or implementations. These methods are twofold and start by fitting K-means centroids to the data. Then, they learn a tree to explain the obtained clusters. The differences in all methods lie in attempts to limit the depth of the tree for the sake of simple explanations as deep trees tend to lose expressivity in explanation. Specifically in the case of ICOT, the K-means only serves as a warm-start initialisation for a greedy tree that is then refined through a mixed integer optimisation problem. We also choose to provide a combination of kernel-K-means and a standard CART decision tree classifier as a baseline for all algorithms, noted K-means+DT. To the best of our knowledge, only ExKMC (Frost et al., 2020) was surprisingly compared to such baseline, and Bertsimas et al. (2021) compared their model to K-means.

Due to the nature of the Douglas model, we preferred to keep the evaluation of this model in a separate subsection on the same datasets.

5.6.1 Performances with as many leaves as clusters

Table 5.4 – Relative K-means score $_{\text{std}}$ (lower is better) after 30 runs on subsamples of 80% of the input datasets divided by the K-means reference score (=1.0). All models are limited to finding as many leaves as clusters.

Dataset	Kauri	K-means+DT	ICOT	IMM	ExShallow	RDM
Atom	$1.03_{0.02}$	$1.04_{0.02}$	$1.05_{0.03}$	$\mathbf{1.02_{0.02}}$	$1.04_{0.02}$	$1.03_{0.02}$
Avila	$\mathbf{1.45_{0.58}}$	$3.33_{1.16}$	X	$1.49_{0.66}$	$1.52_{0.71}$	$1.99_{0.68}$
Cancer	$1.14_{0.03}$	$1.14_{0.03}$	$\mathbf{1.09_{0.04}}$	$1.14_{0.03}$	$1.14_{0.03}$	$1.37_{0.04}$
Car	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	X	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$1.02_{0.03}$
Chainlink	$\mathbf{1.02_{0.01}}$	$\mathbf{1.02_{0.01}}$	$1.04_{0.02}$	$\mathbf{1.02_{0.01}}$	$\mathbf{1.02_{0.01}}$	$1.03_{0.01}$
Congress	$1.09_{0.01}$	$1.09_{0.02}$	$\mathbf{1.08_{0.01}}$	$1.09_{0.01}$	$1.09_{0.02}$	$1.19_{0.02}$
Digits	$1.21_{0.01}$	$1.24_{0.02}$	X	$1.25_{0.02}$	$\mathbf{1.20_{0.01}}$	$1.38_{0.04}$
Engytime	$1.14_{0.04}$	$1.15_{0.05}$	X	$\mathbf{1.13_{0.04}}$	$1.14_{0.06}$	$1.15_{0.05}$
Haberman	$1.10_{0.07}$	$\mathbf{1.08_{0.04}}$	$1.09_{0.04}$	$1.10_{0.07}$	$\mathbf{1.08_{0.04}}$	$1.10_{0.06}$
Hepta	$1.09_{0.09}$	$1.08_{0.03}$	$3.12_{1.94}$	$\mathbf{1.07_{0.04}}$	$1.08_{0.03}$	$1.36_{0.09}$
Iris	$1.18_{0.06}$	$1.19_{0.05}$	$1.92_{0.10}$	$1.17_{0.05}$	$\mathbf{1.16_{0.04}}$	$1.46_{0.10}$
Lsun	$1.09_{0.04}$	$1.09_{0.04}$	$1.14_{0.12}$	$\mathbf{1.07_{0.03}}$	$\mathbf{1.07_{0.04}}$	$1.66_{0.09}$
Mice	$\mathbf{1.08_{0.02}}$	$1.11_{0.04}$	X	$1.14_{0.05}$	$\mathbf{1.08_{0.02}}$	$1.41_{0.06}$
Target	$1.35_{0.05}$	$1.58_{0.13}$	X	$\mathbf{1.19_{0.08}}$	$\mathbf{1.19_{0.03}}$	$1.83_{0.10}$
Tetra	$1.12_{0.15}$	$\mathbf{1.04_{0.02}}$	$\mathbf{1.04_{0.01}}$	$\mathbf{1.04_{0.02}}$	$1.05_{0.02}$	$1.53_{0.06}$
Twodiamonds	$1.05_{0.02}$	$\mathbf{1.04_{0.02}}$	$\mathbf{1.04_{0.02}}$	$1.05_{0.02}$	$\mathbf{1.04_{0.02}}$	$\mathbf{1.04_{0.02}}$
Vowel	$\mathbf{1.07_{0.01}}$	$1.08_{0.01}$	X	$1.09_{0.02}$	$\mathbf{1.07_{0.01}}$	$1.11_{0.01}$
Wine	$1.15_{0.05}$	$1.15_{0.08}$	$1.37_{0.17}$	$\mathbf{1.11_{0.04}}$	$\mathbf{1.11_{0.04}}$	$1.54_{0.10}$
Wingnut	$\mathbf{1.09_{0.02}}$	$\mathbf{1.09_{0.01}}$	$1.14_{0.06}$	$\mathbf{1.09_{0.02}}$	$\mathbf{1.09_{0.02}}$	$\mathbf{1.09_{0.01}}$

As some related works focus on trees with one leaf per cluster, we limit the Kauri tree and the K-means+DT to as many leaves as clusters. However, we could not limit the number of leaves for ICOT, so we restrained the depth to 5 levels. Consequently, the scores of ICOT may outperform other methods due to a freer architecture.

Since some algorithms are deterministic in nature, we introduce stochasticity in results by selecting 80% of the training data over 30 runs. We report the ARI for all algorithms in Table 5.3, the normalised K-means score in Table 5.4 and the WAD in Table 5.5.

We observe in Table 5.3 that Kauri often performs on par with related works. In general, the performances of all methods are quite similar regarding the ARI. Notably, the ARI of Kauri is often close to the K-means+DT baseline, except for the wine dataset. We believe that the differences of scores are in general negligible as they are covered by the standard deviations throughout the multiple subsets of data. However, for very different scores like the ARI on the Target dataset, we believe that this difference can be explained by the order of the choice of splits in the trees owing to the presence of the K-means objective among methods or just the usage of labels. We did not manage to run ICOT on datasets with a large number of features or clusters. This joins

Table 5.5 – WAD scores $_{\text{std}}$ (lower is better) after 30 runs on random subsamples of 80% of the input datasets. All models are limited to finding as many leaves as clusters.

Dataset	Kauri	K-means+DT	IMM	ExShallow	RDM
Atom	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$
Avila	$5.37_{0.13}$	$\mathbf{5.12_{0.17}}$	$8.16_{0.72}$	$5.74_{0.55}$	$7.82_{0.69}$
Cancer	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$
Car	$\mathbf{3.00_{0.00}}$	$3.06_{0.06}$	$3.04_{0.05}$	$3.04_{0.06}$	$\mathbf{3.00_{0.12}}$
Chainlink	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$
Congress	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$
Digits	$\mathbf{4.47_{0.08}}$	$4.60_{0.21}$	$6.55_{0.36}$	$4.88_{0.21}$	$4.48_{0.17}$
Engytime	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$
Haberman	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$
Hepta	$4.80_{0.06}$	$\mathbf{4.75_{0.05}}$	$4.88_{0.07}$	$4.77_{0.06}$	$4.91_{0.06}$
Iris	$\mathbf{2.67_{0.03}}$	$\mathbf{2.67_{0.02}}$	$\mathbf{2.67_{0.01}}$	$\mathbf{2.67_{0.02}}$	$2.68_{0.02}$
Lsun	$\mathbf{2.48_{0.01}}$	$\mathbf{2.48_{0.02}}$	$2.71_{0.11}$	$2.58_{0.13}$	$2.59_{0.09}$
Mice	$\mathbf{4.03_{0.04}}$	$4.14_{0.10}$	$5.61_{0.59}$	$4.16_{0.11}$	$4.38_{0.31}$
Target	$4.20_{0.02}$	$\mathbf{3.74_{0.21}}$	$4.30_{0.07}$	$4.30_{0.02}$	$4.03_{0.02}$
Tetra	$3.23_{0.08}$	$3.25_{0.02}$	$3.25_{0.02}$	$3.26_{0.02}$	$\mathbf{3.14_{0.03}}$
Twodiamonds	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$
Vowel	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$
Wine	$\mathbf{2.62_{0.05}}$	$2.65_{0.04}$	$2.70_{0.02}$	$2.70_{0.02}$	$2.63_{0.10}$
Wingnut	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$

the observations of Bertsimas et al. (2021) who reported a runtime of more than an hour for the Engytime dataset. We observe similarly in Table 5.4 that despite the strong performances of ExShallow on most of the datasets, both Kauri and the K-means+DT baseline do not fall short behind in scores, except on the Target dataset. Finally, we observe good performances in Table 5.5 regarding the depth of the nodes at which the cluster decisions are returned. We purposefully removed in Table 5.5 the Atom, Cancer and Chainlink datasets because they are binary, which implies that the WAD is always equal to 2 when the model is constrained to 2 leaves. Moreover, we did not manage to obtain the tree structure underlying in ICOT to compute the WAD.

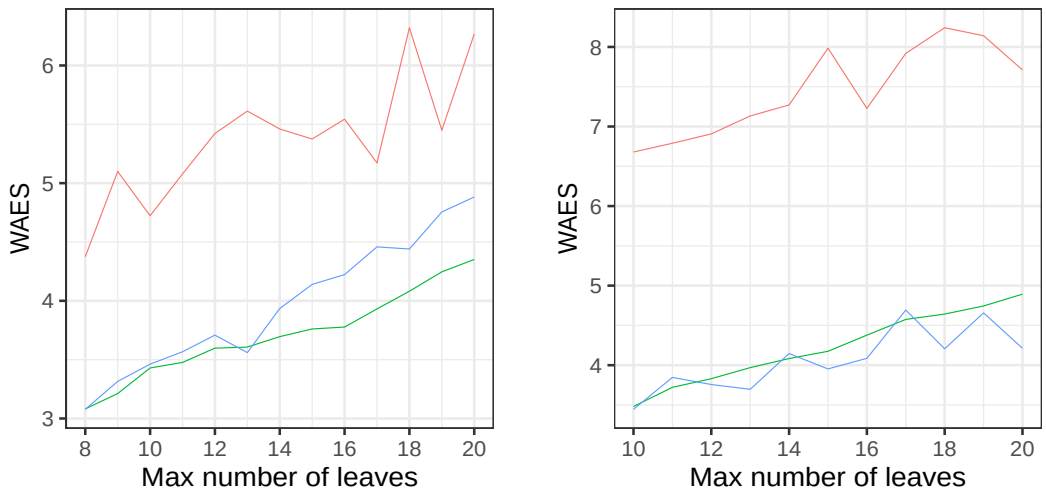
5.6.2 Performances with more leaves than clusters

We run here the exact same benchmark as proposed in section 5.6.1, except we seek to compare Kauri with the ExKMC method. To that end, all trees are now limited to 4 times more leaves than clusters following the result of Frost et al. (2020). Contrary to section 5.6.1, the excessive number of leaves will necessarily imply that multiple leaves might explain a single cluster. We chose then to measure the WAES because we want to emphasize the complexity of the explanation of a cluster rather than the depth of trees. We report the ARI in Table 5.6 and the WAES in Table 5.7.

We observe in Table 5.6 that Kauri often performs on par with ExKMC and the baseline regarding the clustering performances. However, we find that our WAES scores are steadily lower with the baseline compared to ExKMC on most datasets in Table 5.7. In this context, we believe that

Table 5.6 – ARI scores $_{\text{std}}$ (greater is better) after 30 runs on random subsamples of 80% of the input datasets. All models are limited to finding 4 times more leaves than clusters.

Method	Kauri	K-means+DT	ExKMC
Atom	$\mathbf{0.18_{0.03}}$	$0.17_{0.02}$	$0.17_{0.02}$
Avila	$0.03_{0.03}$	$\mathbf{0.04_{0.02}}$	$\mathbf{0.04_{0.02}}$
Cancer	$0.86_{0.01}$	$0.84_{0.02}$	$\mathbf{0.87_{0.01}}$
Car	$\mathbf{0.07_{0.06}}$	$0.05_{0.05}$	$0.06_{0.06}$
Chainlink	$0.10_{0.01}$	$0.09_{0.01}$	$\mathbf{0.11_{0.01}}$
Congress	$0.50_{0.05}$	$\mathbf{0.57_{0.04}}$	$0.53_{0.03}$
Digits	$0.55_{0.03}$	$\mathbf{0.58_{0.02}}$	$\mathbf{0.58_{0.02}}$
Engytime	$0.73_{0.02}$	$\mathbf{0.80_{0.02}}$	$0.72_{0.04}$
Haberman	$\mathbf{-0.00_{0.00}}$	$\mathbf{-0.00_{0.00}}$	$\mathbf{-0.00_{0.00}}$
Hepta	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$
Iris	$\mathbf{0.72_{0.03}}$	$\mathbf{0.72_{0.03}}$	$\mathbf{0.72_{0.04}}$
Lsun	$\mathbf{0.88_{0.03}}$	$\mathbf{0.88_{0.03}}$	$0.87_{0.03}$
Mice	$\mathbf{0.22_{0.01}}$	$0.20_{0.02}$	$0.20_{0.02}$
Target	$0.63_{0.02}$	$0.63_{0.01}$	$\mathbf{0.64_{0.02}}$
Tetra	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$
Twodiamonds	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$
Vowel	$0.09_{0.04}$	$0.12_{0.04}$	$\mathbf{0.15_{0.04}}$
Wine	$0.85_{0.04}$	$\mathbf{0.87_{0.04}}$	$0.85_{0.04}$
Wingnut	$0.15_{0.01}$	$\mathbf{0.41_{0.02}}$	$0.30_{0.05}$

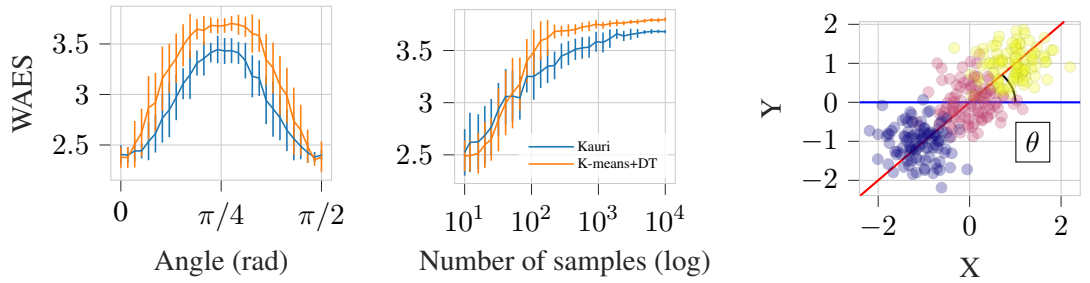


(a) Mice (b) Digits

Figure 5.5 – Explanation scoring with WAES (lower is better) as the maximal number of leaves increases on the mice protein and digits datasets for Kauri (green), K-means+Tree (blue) and ExKMC (red).

Table 5.7 – WAES scores $_{\text{std}}$ (lower is better) after 30 runs on random subsamples of 80% of the input datasets. All models are limited to finding 4 times more leaves than clusters.

Dataset	Kauri	K-means+DT	ExKMC
Atom	$\mathbf{2.37_{0.41}}$	$2.89_{0.44}$	$3.18_{0.72}$
Avila	$\mathbf{5.76_{0.40}}$	$6.25_{0.45}$	$7.87_{0.78}$
Cancer	$\mathbf{3.45_{0.16}}$	$3.88_{0.13}$	$4.02_{0.12}$
Car	$\mathbf{3.00_{0.00}}$	$3.02_{0.05}$	$3.04_{0.05}$
Chainlink	$\mathbf{2.59_{0.32}}$	$2.78_{0.31}$	$3.26_{0.39}$
Congress	$\mathbf{2.63_{0.37}}$	$3.40_{0.28}$	$3.53_{0.32}$
Digits	$\mathbf{6.59_{0.16}}$	$\mathbf{6.59_{0.15}}$	$8.59_{0.35}$
Engytime	$\mathbf{3.03_{0.16}}$	$3.14_{0.05}$	$3.36_{0.26}$
Haberman	$\mathbf{2.00_{0.00}}$	$2.06_{0.13}$	$2.15_{0.35}$
Hepta	$4.39_{0.05}$	$\mathbf{4.37_{0.05}}$	$4.46_{0.05}$
Iris	$\mathbf{3.32_{0.35}}$	$3.48_{0.20}$	$3.65_{0.35}$
Lsun	$\mathbf{2.82_{0.27}}$	$2.86_{0.27}$	$3.39_{0.46}$
Mice	$6.54_{0.29}$	$\mathbf{6.39_{0.31}}$	$8.00_{0.45}$
Target	$\mathbf{4.17_{0.04}}$	$4.22_{0.05}$	$4.18_{0.07}$
Tetra	$3.42_{0.16}$	$\mathbf{3.25_{0.02}}$	$\mathbf{3.25_{0.02}}$
Twodiamonds	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$	$\mathbf{2.00_{0.00}}$
Vowel	$4.06_{0.33}$	$\mathbf{3.85_{0.40}}$	$4.51_{0.41}$
Wine	$\mathbf{3.68_{0.39}}$	$3.86_{0.30}$	$4.36_{0.52}$
Wingnut	$\mathbf{2.00_{0.00}}$	$3.17_{0.02}$	$3.30_{0.20}$

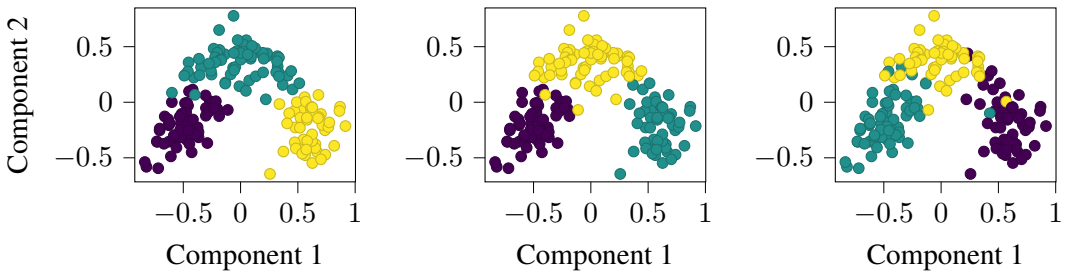


(a) Variations for growing angle, number of samples fixed to 300.

(b) Variations for growing number of samples, angle fixed to $\pi/4$.

(c) Example of dataset for 300 samples and an angle of $\pi/4$

Figure 5.6 – Variations of WAES scores for aligned isotropic 2d Gaussian distributions separated by Kauri or K-means+Tree as the angle of the alignment (red line in 5.6c) with the x-axis (blue line in 5.6c) grows or the number of samples increases over 30 runs. The distance between the means is $\sqrt{2}$ and the scale matrices are $0.2\boldsymbol{I}_2$.



(a) True labels (b) Kauri clusters (c) K-means+DT clusters

Figure 5.7 – PCA of the wine dataset with samples coloured according to clusters found by Kauri or K-means+DT with a χ^2 kernel.

the difference between the baseline and Kauri is mainly due to the contrast between the respective optimisation of a Gini impurity on clustering labels and the greedy optimisation of kernel K-means.

To further investigate some differences, we observe in Fig. 5.5 the differences in WAES scores between K-means+Tree, Kauri and ExKMC (Frost et al., 2020). We observe that overall, for a fixed amount of leaves to use, we obtained explanations with lower WAES scores than ExKMC while maintaining an ARI that is close to K-means+Tree.

To highlight some differences in behaviour between Kauri and K-means + DT, we show in Figure 5.6 how the angles of the decision boundary and the number of samples in the dataset can change the performances in seemingly identical distributions. Indeed, K-means easily builds linear boundaries that are not axis-aligned, hence as the boundaries become less and less aligned with the axes, the decision trees struggle to maintain a low number of leaves to mimic these "diagonal" boundaries. This effect gets worse if the number of samples to separate is high on this decision boundary. However, as soon as the decision boundaries are axis-aligned, the decision tree becomes again a fierce competitor. Both trees have unlimited leaves and stop only when no gain is longer possible.

5.6.3 Varying the kernel

Our comparisons have been done so far with related works using the linear kernel in kernel K-means. However, using a different kernel leads to the absence of a definition of centroids in the Euclidean space where the data lie. Consequently, previously compared methods are not compatible with such a setup because they require an explicit centroid, and we are only left with the K-means+DT baseline as a competitor. We explore 4 different kernels with default parameters from `scikit-learn`: χ^2, additive χ^2, Laplacian and RBF kernels with Table 5.8 for the ARI and Table 5.9 for the normalised K-means score.

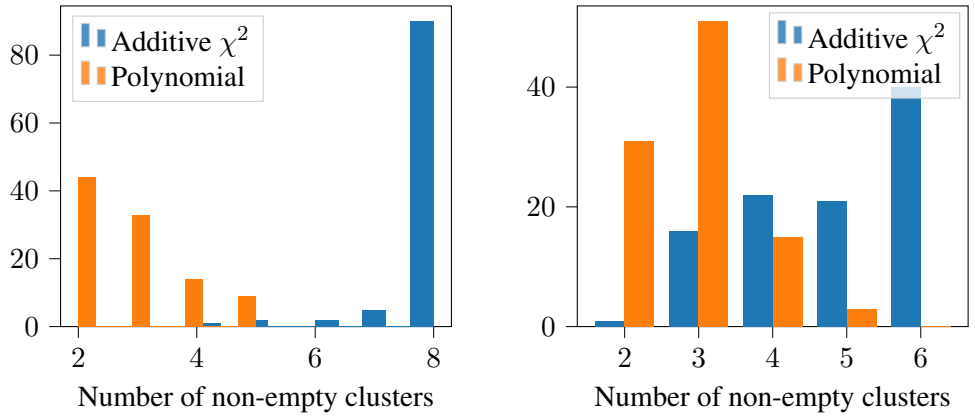
We generally observe equal or stronger ARI scores for the Kauri algorithm in Table 5.8. The same observation goes for the K-means score in Table 5.9, especially for the Laplacian kernel in both tables. We note that some kernel K-means scores are below 1 after normalisation. This is due to the phenomenon of empty clusters that arises in the kernel K-means algorithm. Consequently, the basis on which the decision tree is learnt does not provide enough clusters, thus lowering the ARI and increasing the kernel K-means score. Similarly, the reference kernel K-means score used

Table 5.8 – ARI scores $_{\text{std}}$ (greater is better) after 30 runs on random subsamples of 80% of the input datasets for varying kernels. All models are limited to finding 4 times more leaves than clusters. *: Datasets that suffered from the kernel K-means convergence to empty clusters.

Kernel	Additive χ^2		χ^2		Laplacian		RBF	
Dataset	Kauri	K-means+DT	Kauri	K-means+DT	Kauri	K-means+DT	Kauri	K-means+DT
Atom	$0.10_{0.01}$	$0.09_{0.02}$	$0.09_{0.01}$	$0.11_{0.03}$	$\mathbf{0.44_{0.37}}$	$0.15_{0.04}$	$0.18_{0.03}$	$0.14_{0.02}$
Avila*	$0.02_{0.02}$	$0.00_{0.03}$	$0.02_{0.02}$	$-0.00_{0.01}$	$\mathbf{0.05_{0.02}}$	$0.04_{0.04}$	$0.03_{0.03}$	$0.00_{0.01}$
Cancer	$0.86_{0.01}$	$0.84_{0.02}$	$0.82_{0.02}$	$0.86_{0.02}$	$\mathbf{0.87_{0.01}}$	$0.79_{0.03}$	$\mathbf{0.87_{0.02}}$	$0.78_{0.03}$
Car*	$0.06_{0.06}$	$0.06_{0.06}$	$0.06_{0.06}$	$0.02_{0.04}$	$0.05_{0.06}$	$0.06_{0.06}$	$\mathbf{0.07_{0.06}}$	$0.06_{0.06}$
Chainlink	$0.32_{0.02}$	$0.28_{0.08}$	$0.33_{0.01}$	$0.26_{0.13}$	$\mathbf{0.40_{0.05}}$	$0.36_{0.11}$	$0.11_{0.01}$	$0.09_{0.02}$
Congress	$0.50_{0.04}$	$\mathbf{0.56_{0.03}}$	$0.37_{0.04}$	$0.36_{0.23}$	$0.50_{0.04}$	$\mathbf{0.56_{0.03}}$	$0.49_{0.04}$	$0.55_{0.04}$
Digits	$0.52_{0.03}$	$0.55_{0.03}$	$0.11_{0.03}$	$0.22_{0.04}$	$0.53_{0.04}$	$\mathbf{0.57_{0.03}}$	$0.56_{0.02}$	$0.54_{0.04}$
Engytime	$0.66_{0.03}$	$0.77_{0.03}$	$0.67_{0.02}$	$0.78_{0.02}$	$0.57_{0.07}$	$0.70_{0.02}$	$0.73_{0.04}$	$\mathbf{0.82_{0.01}}$
Haberman	$-0.01_{0.00}$	$0.00_{0.05}$	$-0.01_{0.01}$	$\mathbf{0.02_{0.07}}$	$-0.00_{0.00}$	$-0.00_{0.00}$	$-0.00_{0.00}$	$-0.00_{0.00}$
Hepta	$\mathbf{1.00_{0.00}}$	$0.48_{0.21}$	$\mathbf{1.00_{0.00}}$	$0.62_{0.19}$	$\mathbf{1.00_{0.00}}$	$0.82_{0.14}$	$\mathbf{1.00_{0.00}}$	$0.57_{0.25}$
Iris	$0.67_{0.04}$	$0.62_{0.09}$	$0.67_{0.04}$	$0.61_{0.11}$	$\mathbf{0.78_{0.05}}$	$0.71_{0.14}$	$0.72_{0.03}$	$0.71_{0.05}$
Lsun	$\mathbf{0.98_{0.00}}$	$0.89_{0.21}$	$\mathbf{0.98_{0.01}}$	$0.81_{0.26}$	$\mathbf{0.98_{0.01}}$	$0.96_{0.01}$	$0.88_{0.02}$	$0.93_{0.02}$
Mice*	$\mathbf{0.23_{0.02}}$	$0.19_{0.04}$	$0.19_{0.02}$	$0.21_{0.04}$	$0.20_{0.01}$	$0.20_{0.02}$	$0.21_{0.01}$	$0.17_{0.05}$
Target*	$\mathbf{0.63_{0.01}}$	$0.38_{0.21}$	$\mathbf{0.63_{0.02}}$	$0.49_{0.20}$	$\mathbf{0.63_{0.05}}$	$\mathbf{0.63_{0.07}}$	$\mathbf{0.63_{0.02}}$	$0.39_{0.23}$
Tetra	$0.99_{0.01}$	$0.91_{0.03}$	$0.99_{0.01}$	$0.93_{0.03}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$
Twodiamonds	$0.98_{0.00}$	$0.95_{0.02}$	$0.98_{0.00}$	$0.97_{0.02}$	$\mathbf{1.00_{0.00}}$	$0.77_{0.43}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$
Vowel	$0.05_{0.08}$	$0.11_{0.05}$	$0.12_{0.07}$	$\mathbf{0.14_{0.08}}$	$0.11_{0.03}$	$0.11_{0.04}$	$0.10_{0.03}$	$0.11_{0.04}$
Wine	$0.87_{0.03}$	$0.74_{0.03}$	$\mathbf{0.90_{0.03}}$	$0.73_{0.10}$	$0.89_{0.03}$	$0.83_{0.03}$	$0.85_{0.03}$	$0.80_{0.04}$
Wingnut	$0.13_{0.01}$	$0.29_{0.04}$	$0.14_{0.02}$	$0.29_{0.03}$	$0.17_{0.13}$	$\mathbf{0.44_{0.17}}$	$0.15_{0.01}$	$0.42_{0.03}$

Table 5.9 – Relative Kernel K-means scores $_{\text{std}}$ (lower is better) of Kauri and kernel K-means + Decision Tree after 30 runs on random subsamples of 80% of the input datasets for varying kernels. All models are limited to finding 4 times more leaves than clusters. *: Datasets that suffered from the kernel K-means convergence to empty clusters.

Kernel	Additive χ^2		χ^2		Laplacian		RBF	
Dataset	Kauri	K-means+DT	Kauri	K-means+DT	Kauri	K-means+DT	Kauri	K-means+DT
Atom	$1.05_{0.03}$	$1.09_{0.08}$	$1.05_{0.02}$	$1.10_{0.07}$	$\mathbf{1.00_{0.02}}$	$1.03_{0.02}$	$1.03_{0.02}$	$1.04_{0.02}$
Avila*	$\mathbf{0.47_{0.12}}$	$0.94_{0.24}$	$0.93_{0.25}$	$1.45_{0.23}$	$1.14_{0.28}$	$1.36_{0.30}$	$0.73_{0.28}$	$1.38_{0.43}$
Cancer	$1.04_{0.02}$	$1.05_{0.02}$	$1.02_{0.01}$	$1.03_{0.01}$	$\mathbf{1.01_{0.02}}$	$1.04_{0.02}$	$1.04_{0.02}$	$1.07_{0.03}$
Car*	$\mathbf{1.00_{0.00}}$	$1.01_{0.01}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$\mathbf{1.00_{0.00}}$	$1.01_{0.01}$	$\mathbf{1.00_{0.00}}$	$1.01_{0.01}$
Chainlink	$1.03_{0.01}$	$1.04_{0.03}$	$1.02_{0.01}$	$1.03_{0.04}$	$\mathbf{1.01_{0.00}}$	$\mathbf{1.01_{0.02}}$	$1.02_{0.01}$	$1.02_{0.02}$
Congress	$1.05_{0.02}$	$1.03_{0.01}$	$\mathbf{0.99_{0.00}}$	$1.01_{0.01}$	$1.03_{0.01}$	$1.02_{0.01}$	$1.05_{0.01}$	$1.03_{0.01}$
Digits	$1.07_{0.01}$	$1.08_{0.02}$	$\mathbf{0.99_{0.00}}$	$1.00_{0.00}$	$1.04_{0.01}$	$1.05_{0.01}$	$1.07_{0.01}$	$1.10_{0.03}$
Engytime	$1.09_{0.04}$	$1.06_{0.05}$	$1.09_{0.04}$	$1.07_{0.04}$	$\mathbf{1.03_{0.02}}$	$\mathbf{1.03_{0.02}}$	$1.09_{0.04}$	$1.06_{0.04}$
Haberman	$\mathbf{1.05_{0.04}}$	$1.08_{0.08}$	$\mathbf{1.05_{0.03}}$	$1.10_{0.09}$	$\mathbf{1.05_{0.03}}$	$1.07_{0.05}$	$1.06_{0.06}$	$1.09_{0.07}$
Hepta	$\mathbf{1.01_{0.03}}$	$4.33_{2.01}$	$1.10_{0.05}$	$3.34_{1.65}$	$1.03_{0.02}$	$1.31_{0.28}$	$1.05_{0.03}$	$5.88_{3.22}$
Iris	$1.11_{0.05}$	$1.18_{0.16}$	$1.10_{0.04}$	$1.20_{0.16}$	$\mathbf{1.05_{0.02}}$	$1.09_{0.08}$	$1.06_{0.05}$	$1.06_{0.09}$
Lsun	$1.08_{0.03}$	$1.13_{0.11}$	$\mathbf{1.04_{0.02}}$	$1.11_{0.12}$	$\mathbf{1.04_{0.01}}$	$\mathbf{1.04_{0.01}}$	$1.05_{0.03}$	$1.07_{0.04}$
Mice*	$1.01_{0.02}$	$1.11_{0.13}$	$\mathbf{0.98_{0.01}}$	$1.02_{0.01}$	$1.02_{0.01}$	$1.04_{0.01}$	$1.02_{0.02}$	$1.12_{0.13}$
Target*	$1.07_{0.08}$	$2.08_{0.76}$	$1.12_{0.05}$	$1.71_{0.69}$	$\mathbf{1.02_{0.02}}$	$1.09_{0.12}$	$1.10_{0.04}$	$2.32_{0.96}$
Tetra	$1.05_{0.02}$	$1.06_{0.03}$	$\mathbf{1.02_{0.02}}$	$1.04_{0.03}$	$\mathbf{1.02_{0.01}}$	$\mathbf{1.02_{0.01}}$	$1.03_{0.02}$	$\mathbf{1.02_{0.01}}$
Twodiamonds	$1.03_{0.02}$	$1.03_{0.02}$	$1.05_{0.02}$	$1.05_{0.02}$	$\mathbf{1.01_{0.01}}$	$1.06_{0.09}$	$1.04_{0.01}$	$1.03_{0.02}$
Vowel	$1.04_{0.02}$	$1.03_{0.02}$	$1.03_{0.01}$	$1.03_{0.01}$	$\mathbf{1.01_{0.00}}$	$1.02_{0.01}$	$1.02_{0.01}$	$1.03_{0.02}$
Wine	$1.03_{0.03}$	$1.05_{0.04}$	$1.05_{0.02}$	$1.08_{0.05}$	$\mathbf{1.02_{0.02}}$	$\mathbf{1.02_{0.02}}$	$1.06_{0.03}$	$1.08_{0.03}$
Wingnut	$1.07_{0.02}$	$1.03_{0.01}$	$1.07_{0.02}$	$1.03_{0.02}$	$1.03_{0.01}$	$\mathbf{1.02_{0.01}}$	$1.09_{0.02}$	$1.05_{0.02}$



(a) Mice protein (8 classes)

(b) Target (6 classes)

Figure 5.8 – Number of non-empty clusters for 100 runs of kernel K-means with an additive χ^2 or polynomial kernel. The algorithm had to find the same number of clusters as classes per dataset.

for normalisation suffered from the same problem. Empirically, we observed this behaviour for 2 implementations of the kernel K-means algorithm. *

As a simple example, we ran 100 kernel K-means models with an additive χ^2 kernel or a polynomial kernel for the Mice protein dataset and the Target dataset. From Figure 5.8, we observe that for a relatively small number of clusters, the kernel K-means may converge to several empty clusters, with the worst effect from the polynomial kernel in this example. To complete Figure 5.8, the number of clusters found with the best kernel K-means score was 8 for the Mice protein dataset and 6 for the Target dataset. In contrast, datasets with only 2 clusters to find often converged to the good number of clusters, making scores comparable.

This shows that with the end-to-end construction of the Kauri tree, we do not suffer from dependence to the basis K-means algorithm, and manage to get the correct user-desired number of clusters.

In the case of the presented datasets in Table 5.8 where kernel K-means managed to converge to the desired number of clusters, we think that the improved performances by Kauri come from its greedy nature. The greedy optimisation leads to a local minimum where no branch can be further found, even with fewer leaves than the maximum specified. Consequently, the obtained tree remains in a simpler state than a decision tree that would overfit the labels from kernel K-means. We provide a simple visualisation through PCA and clustering of the wine dataset in Figure 5.7.

5.6.4 Pure numpy Douglas performances

We present here in the tables 5.10 and 5.11 respectively the ARI and normalised K-means score of our two implementations of the Douglas algorithm. We first observe that in most datasets, the performances of both algorithms are similar. We note that Douglas finds clusters for which the K-means score can exceed the score of a K-means algorithm even for the MMD objective, *e.g.* 10 times more on the Hepta dataset or almost 4 times more on the Target dataset. We believe that this is due

*. https://gist.github.com/mblondel/6230787 and https://tslearn.readthedocs.io/en/latest/gen_modules/clustering/tslearn.clustering.KernelK-means.html

Table 5.10 – ARI scores $_{\text{std}}$ (greater is better) after 30 runs on random subsamples of 80% of the input datasets. We present here the results on two different Douglas implementations with OvO GEMINIs.

Method	Numpy version		Torch version	
GEMINI	MMD	Wasserstein	MMD	Wasserstein
Atom	$0.08_{0.06}$	$\mathbf{0.21_{0.08}}$	$0.06_{0.07}$	$0.05_{0.06}$
Avila	$\mathbf{0.02_{0.03}}$	$0.01_{0.02}$	$-0.01_{0.01}$	$-0.00_{0.02}$
Cancer	$0.81_{0.05}$	$0.80_{0.06}$	$\mathbf{0.83_{0.02}}$	$0.82_{0.02}$
Chainlink	$0.10_{0.06}$	$\mathbf{0.12_{0.06}}$	$0.10_{0.05}$	$0.08_{0.04}$
Congress	$0.50_{0.11}$	$0.44_{0.12}$	$\mathbf{0.56_{0.04}}$	$\mathbf{0.56_{0.04}}$
Engytime	$0.33_{0.17}$	$\mathbf{0.44_{0.21}}$	$0.43_{0.13}$	$0.40_{0.13}$
Haberman	$\mathbf{0.02_{0.06}}$	$0.01_{0.05}$	$-0.00_{0.01}$	$-0.00_{0.01}$
Hepta	$\mathbf{0.25_{0.12}}$	$\mathbf{0.25_{0.10}}$	$0.12_{0.09}$	$0.16_{0.11}$
Iris	$\mathbf{0.51_{0.09}}$	$0.46_{0.12}$	$0.30_{0.24}$	$0.37_{0.23}$
Lsun	$0.51_{0.24}$	$\mathbf{0.55_{0.21}}$	$0.32_{0.23}$	$0.45_{0.17}$
Target	$0.06_{0.06}$	$0.08_{0.11}$	$0.11_{0.09}$	$\mathbf{0.15_{0.08}}$
Tetra	$\mathbf{0.48_{0.17}}$	$0.44_{0.16}$	$0.23_{0.10}$	$0.27_{0.15}$
Twodiamonds	$0.67_{0.48}$	$0.77_{0.42}$	$0.70_{0.45}$	$\mathbf{0.82_{0.37}}$
Vowel	$0.02_{0.03}$	$0.01_{0.02}$	$\mathbf{0.04_{0.04}}$	$\mathbf{0.04_{0.04}}$
Wine	$\mathbf{0.43_{0.15}}$	$0.42_{0.15}$	$0.36_{0.11}$	$0.33_{0.10}$
Wingnut	$0.39_{0.36}$	$\mathbf{0.56_{0.41}}$	$0.28_{0.28}$	$0.28_{0.28}$

Table 5.11 – K-means scores $_{\text{std}}$ (lower is better) after 30 runs on random subsamples of 80% of the input datasets. We present here the results on two different Douglas implementations with OvO GEMINIs.

Method	Numpy version		Torch version	
GEMINI	MMD	Wasserstein	MMD	Wasserstein
Atom	$1.16_{0.06}$	$1.14_{0.07}$	$\mathbf{1.11_{0.05}}$	$\mathbf{1.11_{0.05}}$
Avila	$3.54_{1.14}$	$3.62_{1.24}$	$3.71_{1.07}$	$\mathbf{3.35_{1.06}}$
Cancer	$1.12_{0.07}$	$1.15_{0.08}$	$\mathbf{1.04_{0.02}}$	$\mathbf{1.04_{0.03}}$
Chainlink	$\mathbf{1.06_{0.05}}$	$1.07_{0.05}$	$1.07_{0.07}$	$\mathbf{1.06_{0.05}}$
Congress	$1.11_{0.04}$	$1.17_{0.07}$	$\mathbf{1.05_{0.02}}$	$\mathbf{1.05_{0.01}}$
Engytime	$1.22_{0.09}$	$1.18_{0.09}$	$\mathbf{1.15_{0.06}}$	$1.19_{0.09}$
Haberman	$1.38_{0.26}$	$\mathbf{1.27_{0.23}}$	$1.29_{0.19}$	$1.30_{0.21}$
Hepta	$10.99_{2.70}$	$\mathbf{10.83_{2.25}}$	$13.67_{1.70}$	$13.25_{2.17}$
Iris	$\mathbf{2.03_{0.76}}$	$2.43_{1.04}$	$3.64_{1.70}$	$3.21_{1.76}$
Lsun	$1.89_{0.42}$	$\mathbf{1.80_{0.42}}$	$2.27_{0.50}$	$2.12_{0.20}$
Target	$\mathbf{3.81_{0.70}}$	$3.86_{0.67}$	$3.97_{0.48}$	$4.02_{0.47}$
Tetra	$\mathbf{2.57_{0.63}}$	$2.66_{0.49}$	$3.17_{0.33}$	$3.05_{0.48}$
Twodiamonds	$1.26_{0.31}$	$1.19_{0.26}$	$1.23_{0.28}$	$\mathbf{1.15_{0.23}}$
Vowel	$1.09_{0.02}$	$1.09_{0.02}$	$\mathbf{1.05_{0.02}}$	$1.06_{0.02}$
Wine	$\mathbf{1.37_{0.17}}$	$1.39_{0.14}$	$1.46_{0.14}$	$1.45_{0.11}$
Wingnut	$1.10_{0.04}$	$1.13_{0.05}$	$1.09_{0.04}$	$\mathbf{1.08_{0.03}}$

to the nature of the algorithm which can provide non-linear boundaries through the combinations of multiple soft-binnings. In general, except for the Congress dataset, the performances of the Douglas model are worst in ARI than the binary decision trees from Table 5.3.

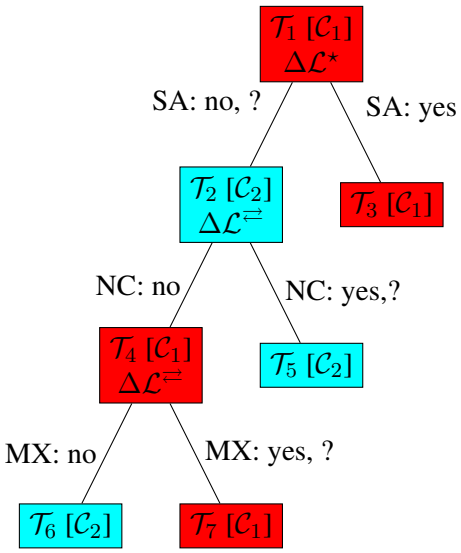


Figure 5.9 – The unsupervised Kauri tree for 2 clusters on the Congressional votes dataset. SA stands for the El Salvador Aid vote, NC for the Nicaraguan Contras vote and MX for the MX-missile vote. The question mark means that the voter did not vote or was missing. Nodes contain their name, the associated cluster to which they assign samples and the type of split that occured during learning.

5.6.5 A qualitative example of the obtained decision tree

In this example, we focus on the congressional votes dataset that details 16 key votes from the 435 members of the US Congress in 1985. The targets of the dataset are the Republican or Democrat affiliations of the voters. We preprocessed the dataset by binarising the vote outcome with -1 for “no" and 1 for “yes". Missing values due to the absence of votes were converted to 0 which is midway between yes and no and hence does not influence the linear kernel by favouring one type of answer. The Kauri tree that was fitted on this dataset is described in Figure 5.9. The obtained clusters translate very well the Republican and Democrat opposition through arming and international assistance, with one cluster containing up to 73% of Republicans and the second one adding up to 96% of Democrats. The ARI is 0.47 for this tree which corresponds to an unsupervised accuracy of 84%.

Upon running 30 times the Douglas tree on this dataset, we measured the number of active cut points. The most selected active cut points were on the exact same features as the ones selected by Kauri in Figure 5.9: the aid to Nicaraguan Contras (selected 93% of time), the El Salvador aid (83%) and the MX missile votes (63%). The models had an average ARI of 0.53.

5.7 Conclusion

We introduced a framework for unsupervised end-to-end learning of trees. By combining tree structures with GEMINI, we derived two novel examples: Kauri and Douglas. The former maximises a kernel-K-means-like objective to build iteratively unsupervised splits through the affectation of tree leaves to existing or new clusters while the latter exploits the combined potential of differential trees and the Wasserstein distance. Kauri can be privileged for small-scale datasets

whereas Douglas is better suited for long datasets on condition of few features. Overall, both methods achieve good performances in clustering with Kauri being on par with related works for unsupervised trees using shallower trees while tackling the empty cluster behaviour of kernel K-means owing to its greedy nature. The strong advantage of these methods is building an interpretable by-nature clustering instead of seeking to explain another clustering output from a different algorithm. Finally, we think that the combination of K-means and a decision tree remains a strong baseline that should be provided in works on unsupervised tree works.

These models complete the GEMINI framework that we can now apply on the PROGRESSA dataset.

PART

Applying clustering for aortic stenosis

CHAPTER 6

A GEMINI pipeline for the identification of phenogroups of aortic stenosis

7.1 Overview of the thesis contributions 135
7.1.1 Discriminative clustering 135
7.1.2 Phenogroups of aortic stenosis 135
7.2 Ongoing work 135
7.2.1 Adding supervision to consensus clustering 135
7.2.2 Towards heterogenous source clustering 136
7.3 Perspectives and future work 137
7.3.1 Integration of radiomics in PROGRESSA clustering 137
7.3.2 Exploiting pretrained model for clustering 137
7.3.3 Learning metrics 138

Important disclaimer: A part of this chapter's contents and figures credit go to Marie-Ange Fleury, PhD Student in collaboration with Nancy Côté, PhD, and Philippe Pibarot, DVM PhD from the Quebec Heart and Lung Institute, or were jointly created.

6.1 Introduction

Aortic stenosis (AS) is a chronic progressive disease and is the most prevalent valvular heart disease in high income countries (Coffey et al., 2021 ; Iung et al., 2019 ; Lindman et al., 2016). In North America, 3 million people are estimated to be affected by AS, and its prevalence is expected to increase substantially with the aging of the population and its ensuing health and economic burden is expected to do the same (Coffey et al., 2021 ; Roth et al., 2020). To this day, no effective pharmacological treatment to prevent the development and/or progression of AS exists. Surgical aortic valve replacement (SAVR) and transcatheter aortic valve implantation (TAVI) remain the only therapeutic options for patients with severe AS (Beyersdorf et al., 2021 ; Otto et al., 2021). Echocardiography is the primary imaging modality used for diagnosing and assessing

hemodynamic severity and progression rate of AS, which ultimately determine optimal timing for intervention (Beyersdorf et al., 2021 ; Otto et al., 2021). To further adjust the pharmacotherapeutic profile of patients ahead of such interventions, we hypothesise that different phenogroups exist amongst AS patients. These phenotypes could potentially be non-mutually exclusive and determining the specific physiopathological pathways of AS are more present in specific patients. Knowing this could help improve patient-specific pharmacological treatment for AS. Machine and deep learning have been previously used for various cardiology tasks, including risk prediction and decision-making optimisation (Ahmad et al., 2018 ; Feeny et al., 2019 ; Motwani et al., 2017 ; Tokodi et al., 2020).

Recently, several studies reported the applicability and accuracy of machine and deep learning-based algorithms to detect AS in various settings (Chang et al., 2021 ; Cohen-Shelly et al., 2021 ; Hernandez-Suarez et al., 2019 ; Kwon et al., 2020 ; Wang et al., 2020). Furthermore, the usefulness of a novel ML pipeline that integrates a few echocardiographic parameters to improve risk stratification of AS was previously demonstrated (Sengupta et al., 2021). Approaches for identifying phenogroups, *i.e.* the clustering task, have already been applied in aortic stenosis (Bohbot et al., 2022 ; Kwak et al., 2020 ; Lachmann et al., 2021 ; Sengupta et al., 2021). However, these related works often focus on the identification of clusters for which the defining features are the severity and/or survival rate of patients rather than clinical or echocardiographic variables. This is interesting for determining populations with strong risk factors, but is different from identifying causes of AS that could be targeted by future therapies.

Based on clinical experience, we started from the hypothesis of invariant phenogroups over time. We hypothesised that each patient belong to one or several specific phenogroups for which only the severity will increase over time. Therefore, we developed a complete pipeline for clustering without parametric assumptions on the data distribution and selecting the most stable models through multiple visits of patients. By applying this pipeline to different sources of data for the same patients, we created different phenogroups, which are not mutually exclusive throughout all patients. From previous studies (Capoulade, Clavel, et al., 2012 ; Fatima et al., 2019 ; Gardezi et al., 2018), our medical collaborators hypothesised that the main phenogroups of AS will be lipidic, inflammatory, thrombotic (Sellers et al., 2019), fibrotic (Simard et al., 2017) and calcific (Lindman et al., 2016).

Preliminary work: Before our search for phenogroups of aortic stenosis on the PROGRESSA dataset, we performed supervised analyses on this dataset (Mrs. Sanabria et al., under review). In this work, we focused on the forecast of the AS progression rate for patients on a 2-year and 5-year window. To that end, recurrent neural networks (RNNs) were trained to predict the presence of any clinical outcome, *i.e.* aortic valve intervention, all-cause mortality or AS hemodynamic progression, in a window of n years to come. The final model achieved an average AUC of 86% for the annual follow-up visit of each patient, thus being superior to non-sequential machine learning approaches, which reached an AUC of 75%. Overall, this provided a good example of how deep learning approaches could predict disease progression and clinical outcomes in patients with mild to moderate AS.

6.2 Known phenogroups

In the last decade, other works focused on identifying phenogroups for aortic stenosis using clustering methods. These methods often tried to seek mutually exclusive phenotypes qualifying the severity of aortic stenosis (Sengupta et al., 2021) or qualifying the survival rates of different populations (Kwak et al., 2020). Overall, these methods often accessed a population of patients of size comparable to PROGRESSA with few hundreds, yet performed feature selection and clustering in different flavours. Indeed, other aortic stenosis phenotyping works rather started with variable selection using PCA as a variable importance criterion and redundant variables removal according to Pearson correlation (Kwak et al., 2020 ; Lachmann et al., 2021). This implies that the selection process is independent of the clustering purpose. The number of initial variables before selection is lower than in our study: from 5 (Bohbot et al., 2022) to 60 (Bohbot et al., 2022). The clustering algorithms then ranged from ward agglomerative clustering (Lachmann et al., 2021) to model-based clustering using `mclust` (Scrucca, Fop, Murphy, & Raftery, 2016) in R (Kwak et al., 2020) passing through topological data analysis (Bohbot et al., 2022). The found clusters often highlighted phenotypes with severe AS and low survival rate (Kwak et al., 2020 ; Sengupta et al., 2021), present extensive disease characteristics such as high New-York Heart Association (NYHA) score and impaired cardiac functions (Lachmann et al., 2021), potentially severe AS with one or several comorbidities especially among the older population (Bohbot et al., 2022 ; Kwak et al., 2020), and finally groups with younger population still through healthy AS.

6.3 Methods

We constructed using GEMINI a full pipeline for finding potentially stable-over-time clusters along variable selection. A graphical summary of the pipeline is provided in Figure 6.1.

6.3.1 PROGRESSA modalities and datasets

The PROGRESSA study is a cohort of 351 patients with at least mild AS, *i.e.* peak aortic jet velocity $\geq$ 2 m/s. These patients underwent yearly follow-up visits, thus accounting for a different number of visits per patient depending on their clinical outcome. Patients were excluded if they had symptomatic AS, moderate or greater aortic regurgitation, mitral valve disease (stenosis or regurgitation), left ventricular ejection fraction (LVEF) $\leq$ 50%, and if they were pregnant or lactating.

The PROGRESSA study comprises three different sources of data: a *clinicopathological* database, a *proteomics* database, and a *radiomics* database.

The clinicopathological database consists of clinical and metabolic data. These data include age, sex, body surface area (BSA), body mass index (BMI) and functional status, *i.e.* the functional classification of the New York Heart Association (NYHA) at the time of the index echocardiography. This database also includes clinical comorbodities such as hypertension, diabetes mellitus, history of smoking and other clinical risk factors. Hematologic profiles are also dressed for each patient using fasting blood samples. The variables in the clinicopathological database are continuous and categorical.

The proteomics database contains the expression levels of 100 selected proteins, most of which are related to the hematologic profile, *e.g.* alipoproteins, clusterin, or coagulation factors. As for the radiomics modality, this thesis Chapter is the first mention ever of the proteomics modality in

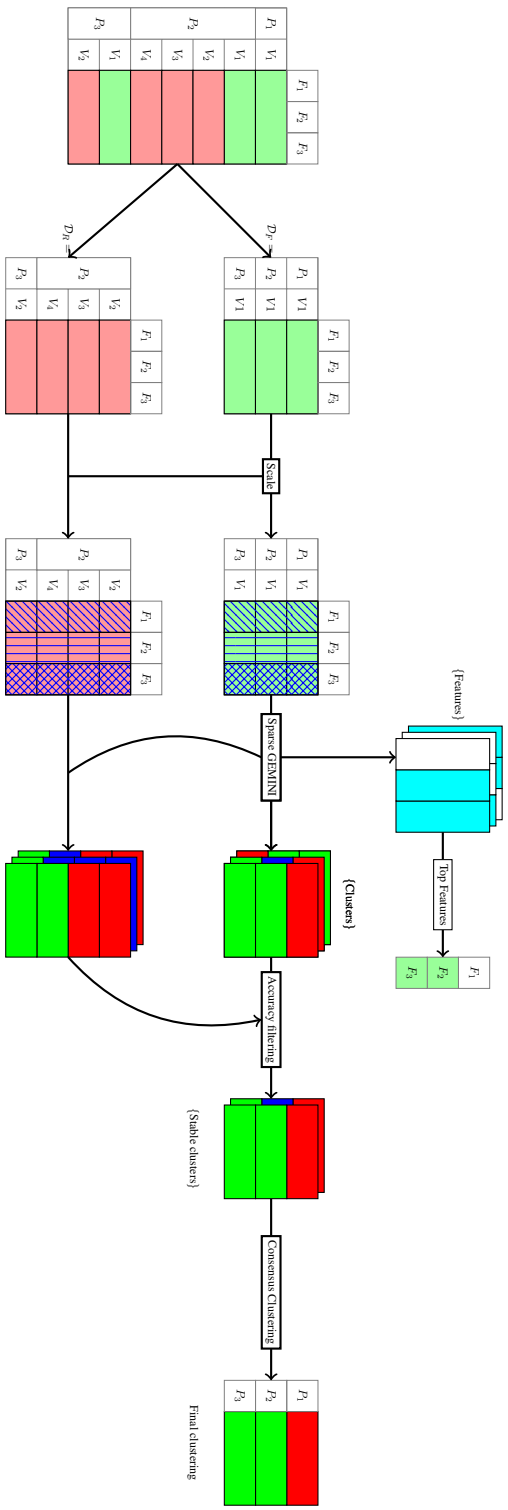


Figure 6.1 – Complete pipeline for finding stable-over-time clusters in the PROGRESSA dataset with Sparse GEMINI and nonparametric consensus clustering.

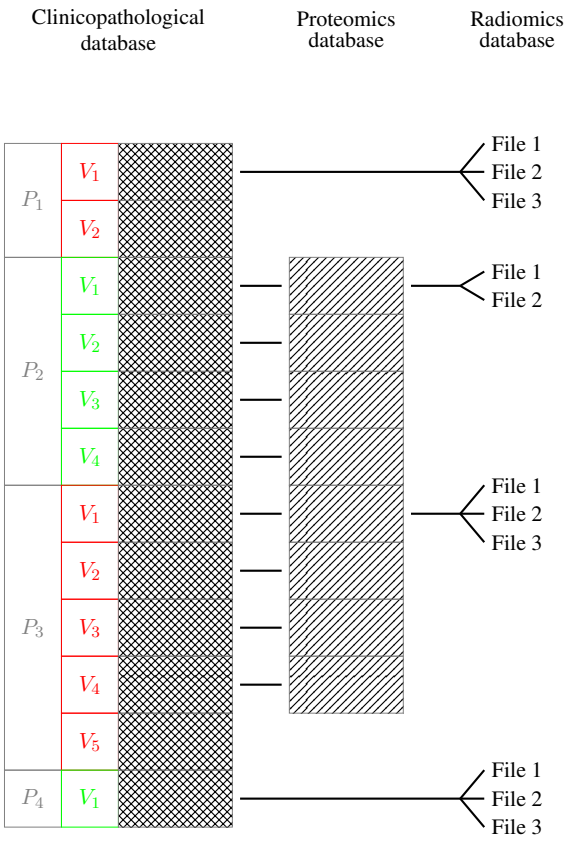


Figure 6.2 – In the PROGRESSA dataset, 3 major modalities coexist. However, each of this modality spans over a different set of visits per patient. Moreover, the proteomics database is only available for a subset of patients. P stands for a patient, V for one of his/her visit and gray areas represent the presence of data. In the radiomics database, the files are videos or pictures.

PROGRESSA. This portion of the dataset was acquired thanks to collaborators outside of the AS project, and we consequently have no background on the choice of technology and methodology for obtaining these few proteins.

Finally, the radiomics modality consists of comprehensive Doppler-echocardiography conducted by the same team of sonographers and cardiologists. The images were analysed by experienced readers. From this analysis, echocardiographic variables were extracted and incorporated into the clinicopathological database, including stroke volume, left ventricular outflow tract, the velocity time integral, the mean pressure gradient and the aortic valve area.

Each patient from the dataset is associated with multiple visits: an initial visit and one or more follow-ups. However, not all visits display the same data. The clinicopathological database contains information for all patients throughout all visits, up to some missing values. Proteomics only concerns approximately half of the patients and only the first four visits. Finally, the radiomics modality of the PROGRESSA study only concerns the first visits of each patient. From few to many videos can be associated to each patient. We provide a graphical summary of the multimodal merges in Fig. 6.2.

Table 6.1 – Dimensions of the considered datasets extracted from the PROGRESSA study. The number of features exceeds the number of variables because of the one-hot encoding of the categorical variables.

Database	Number of samples	Number of variables	Number of features
Clinicopathological	351	87	95
Proteomics	141	87	87
Fused modalities	141	179	191

As our goal is to find stable clusters of phenogroups per patient, we divided all datasets into two subsets: the subset of the first patients' visits $\mathcal{D}_F$ and the subset of subsequent visits $\mathcal{D}_R$. The set of subsequent visits was used to assess the stability of the clusters found using the first visits only: we favour models that were able to put a patient in the same cluster regardless of its visit number following the constraint of time invariance. As mentioned above, the proteomics dataset limits the number of visits. As all patients come a variable number of times, those undergoing proteomics study have an exact count of 4 proteomics visits which correspond to the first, second, third and fourth visit of the clinicopathological database. Consequently, the set of patients with proteomics data is smaller than the global set of all patients.

We could not use the radiomics modality because we do not have access to videos and pictures for other visits than the first one and therefore could not assess if the clusters were stable over time using subsequent visits. With two modalities left, we considered 3 different subsets that we will call indifferently "the dataset" in the remainder of this section: *clinicopathological*, *proteomics*, and the early fusion of both modalities that we call *fused*.

When integrating the proteomics database, whether alone or with the clinicopathological database, we chose, in agreement with our medical collaborators, to only look at the second, third and fourth visits, therefore discarding the first one.

6.3.2 Preprocessing

We started by dropping the variables that have more than 5% missing values and replacing the missing values for the remaining ones either by the mean or the most frequent term depending on the type of variable. This strategy present the advantage of being agnostic to the acquisition technology for proteomics data which is often plagued by missing values Harris, Fondrie, Oh, et Noble (2023). Continuous variables were transformed with robust scaling. This type of scaling removes the median and divides the data by the interquartile range. The specific interest of robust scaling is its preservation of outliers: this avoids compression of the data in a specific region of space as the z-score normalisation would do. This is particulary interesting for GEMINI methods as this will lead outliers to remain far from other samples. For categorical variables, we employed two different transformations depending on the number of categories. Non-binary variables were encoded using one-hot encoding. As we intended to use GEMINI for clustering, which is distance-based, we wanted to ensure that all categorical variables have the same distance when varying from one category to another. Therefore, the binary variables were transformed to either $-\sqrt{2}/2$ or $\sqrt{2}/2$. The goal of such an encoding is to ensure that the distance between two different categorical variables remains $\sqrt{2}$ while avoiding excessive features for binary variables. A binary variable thus produces a one dimensional feature instead of a two-dimensional feature with one-hot-encoding.

The scalers were fitted on the set of the first visits. The subsequent visits then underwent the same scaling, just as we would do for a test set in a supervised context. We present a summary of the datasets dimensions in Table 6.1. It is important to note here that the case of the fused dataset does not have the same initial number of input variables since the reduction of the number of patients allowed some variables to have fewer missing values than the 5% threshold. Consequently, these five additional variables were only available in the fused modalities dataset and emerged from the clinicopathological database.

6.3.3 Applying multiple Sparse GEMINI models

For each dataset previously described, we applied the Sparse GEMINI algorithm from Chapter 4 with static training regime. We seeked to find between 5 and 10 clusters using either a linear regression or a multi-layered perceptron as inference model combined with the one-vs-all or one-vs-one GEMINI. The upper limit of 10 clusters was arbitrarily agreed for the sake of interpretation by cardiology experts. In view of the very similar performances in terms of clustering with slightly better variable selection in Section 4.4.3, we chose to use only the MMD-GEMINI with linear kernel and drop the Wasserstein-GEMINI, which would have been too costly. For the group-lasso regularisation, we let each continuous variable on its own and created a group per one-hot-encoded variable. Each combination of model and objective was run 30 times, resulting thus in 120 different models per number of cluster. Each model provided a clustering of the patients' first visit and a subset of features that were selected to obtain these clusters.

6.3.4 Variables ranking

Although we previously concluded that Sparse GEMINI could be good at eliminating noisy independent features for a reasonable amount of input features and samples in Chapter 4, some variance remains at sake in the number of selected variables. Indeed, when the number of input variables is high, the same subsets of variables may not always be selected by Sparse GEMINI. We will highlight the selection rates in tables 6.4, 6.5 and 6.6. Consequently, we were interested in using the 120 trained GEMINI models to further refine our variable selection. By simply counting how many times each variable was kept by a final model, we obtained an ordering of the variables, from the most frequently involved to the least used. For later plots, we kept the variables that were selected more than 95% of the times, *i.e.* by at least 114 models.

6.3.5 Accuracy filtering for time stability

To find clusters that could match stable-over-time phenogroups, we needed to filter the models that did not achieve such invariance through the visits. Since we consider that the phenogroup of a patient does not change over time, we used the clustering of the first visit as ground-truth label per patient. Then, we simply clustered with the same model all subsequent visits and measure their accuracy. However, as each patient comes with multiple visits, the unweighted accuracy may not reveal stable clusters per patient. For instance, a patient with 6 correctly clustered visits and another patient with 2 misclustered visits yield an accuracy of 75%, which does not reveal stable phenogroups at the individual level: we prefer an accuracy of 50% because we do not successfully cluster one of the patients. That is why we introduced weighting factors per visit to account for the imbalances: 1 over the number of visits. We kept the models that have an accuracy within 90% of the best accuracy achieved per dataset / number of clusters.

6.3.6 Consensus clustering

We finished the pipeline with consensus clustering (Strehl & Ghosh, 2002) to provide a single clustering per dataset and number of clusters. We now cover consensus clustering more in-depth than in Section 3.5.8.

6.3.6.1 Building the consensus matrix

Consensus clustering consists in aggregating different clusters produced by a finite number of clustering models and producing a final agreeing clustering. The intuition is that if two samples are always in the same cluster, they are likely to be together in the final clustering, otherwise they should not. To that end, we build a *consensus* matrix that describes how frequently two samples are in the same clusters (Monti, Tamayo, Mesirov, & Golub, 2003).

In consensus clustering, we assume that we have a set of T clustering models trained on the dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$. We write the t-th model as the distribution of the cluster membership y among K clusters given the data $\boldsymbol{x}$ using the parameters θ^t: $p_{\theta^t}(y|\boldsymbol{x})$.

We then define the connectivity matrix $\boldsymbol{C}^{(t)} \in \{0,1\}^{n\times n}$ of the t-th model that counts when two samples are put in the same clusters by the model:

$$\boldsymbol{C}_{ij}^{(t)} = \mathbb{1}\left[\arg\max_k p_{\theta^t}(y=k|\boldsymbol{x}_i) = \arg\max_k p_{\theta^t}(y|\boldsymbol{x}_j)\right]. \tag{6.1}$$

Finally, the consensus matrix $\boldsymbol{M} \in [0,1]^{n\times n}$ is the connectivity matrix divided by T to obtain the frequency of similar clustering per pair of samples:

$$\boldsymbol{M}_{ij} = \frac{\sum_{t=1}^T \boldsymbol{C}_{ij}^{(t)}}{T}. \tag{6.2}$$

6.3.6.2 Using the consensus matrix

The consensus matrix $\boldsymbol{M}$ comes with 2 different usages. It can be interpreted as a distance between samples using $1-\boldsymbol{M}$, or can be used to derive an informative score on the clustering difficulty of the dataset: the proportion of ambiguous clusters (PAC, Şenbabaoğlu et al., 2014). For the PAC, the intuition is that a set of models that find clear-cut clusters would always be certain about pairs of samples being together or not. Consequently, we would expect the least certain pairs to have a consensus value close to 0.5, *i.e.* uncertainty. The PAC is defined as the proportion of such uncertain pairs. For 2 arbitrary thresholds $u_1 < u_2$, often defined to 0.1 and 0.9, that bound the definition of an uncertain pair, the PAC score is computed as:

$$\text{PAC}_{u_1,u_2}(\boldsymbol{M}) = \frac{\sum_{i=1}^n \sum_{j=1}^n \mathbb{1}[u_1 \leq \boldsymbol{M}_{ij} \leq u_2]}{n^2}. \tag{6.3}$$

The PAC score is bounded in $[0,1]$, and the lower the better. A PAC score equals to 0 reveals that the set of models always found the same clusters.

6.3.6.3 Consensus model

The distance matrix $1-\boldsymbol{M}$ can be incorporated into any distance-based clustering algorithm to produce the final clustering. Common solutions for consensus clustering, for instance, employ

```python
# We create a nonparametric model with custom distance
model = gemclus.nonparametric.CategoricalWasserstein(
    n_clusters=3, ovo=True, metric="precomputed")

# We fit the model on the data using 1-consensus as a distance
consensus_pred = model.fit_predict(X, 1-consensus)
```

Listing 6.1 – An example of consensus clustering using GEMINI. We assume to have a consensus matrix describing the pairwise co-clustering frequency of samples.

ascending hierarchical clustering (Kiselev et al., 2017). Although such approach is well-motivated, we believe that this is not the necessarily perfect algorithm for consensus. In fact, in hierarchical clustering, clusters are merged according to a linkage function. For instance, the single linkage merges clusters with the closest samples. With such a linkage, two clusters would be merged as soon as they possess one part of a pair for which the distance is 0. Consequently, if most of the samples in one cluster should not match those in another cluster, they could still be merged as soon as there is a positive pair favouring some proximity. Other linkage methods may even lead to undesired solutions. For example, complete linkage would merge clusters as soon as they share a pair of samples that is the farthest, *i.e.* that should not be together. This result is the contrary of what we seek.

We propose instead to exploit GEMINI to the fullest and used it again for consensus clustering. To that end, we simply used the nonparametric model, initially defined in Section 3.5.1:

$$y|\boldsymbol{x} = \boldsymbol{x}_i \sim \text{Categorical}(\boldsymbol{\theta}_{1i}, \boldsymbol{\theta}_{2i}, \dots, \boldsymbol{\theta}_{Ki}), \tag{6.4}$$

with parameters $\boldsymbol{\theta} \in \mathbb{R}^{K \times n}$. This is interesting because it breaks the relationships between the value of the samples and the predicted clusters. A probability of belonging to a cluster is directly assigned to each sample, however this probability $\boldsymbol{\theta}_{ik}$ is not computed using $\boldsymbol{x}_i$. Consequently, only the distance between samples matters, as in hierarchical clustering. We used the OvO Wasserstein objective to train this model. This allowed us to train all parameters with a holistic view on the distance matrix thanks to the Wasserstein distances. Indeed, the Wasserstein distance takes into account the entire distributions, and not just their mean, to compare them. Consequently, the model was likely better at breaking the uncertain pairs. We show a sample of code achieving this consensus clustering with GemClus in Listing 6.

Thus the complete pipeline for finding clusters in the PROGRESSA dataset modalities involved only flavours of GEMINI.

6.4 From clusters to phenogroups

We present here all results that were produced by the pipeline and the finally identified phenogroups. The results notably helped filtering out models that were not deemed interesting by our cardiology expert colleagues. The pipeline results cover the 3 modalities previously discussed: the clinicopathological database, the proteomics database, and their fusion. The identified phenogroups were only elaborated on the results of the clinicopathological database.

Table 6.2 – Average accuracy [95% confidence interval] of the individual models for the 3 datasets. Each combination of model Regression/MLP with a mode OvA/OvO represents 30 trained models for 10 clusters. Accuracy is evaluated on the subsequent visits using the first visit as ground-truth. Confidence intervals are computed assuming a normal distribution.

Dataset	Logistic regression		MLP	
	OvA	OvO	OvA	OvO
Clinicopathological	60% [59% - 60%]	55% [55% - 56%]	55% [54% - 56%]	53% [53% - 54%]
Proteomics	55% [54% - 56%]	30% [28% - 31%]	45% [43% - 48%]	29% [27% - 31%]
Fused modalities	60% [58% - 62%]	60% [59% - 61%]	53% [52% - 54%]	50% [48% - 52%]

Table 6.3 – Proportion of ambiguous clusters (lower is better) for the subsets of accurate-over-time models depending on the number of clusters and datasets.

Dataset	Number of clusters					
	5	6	7	8	9	10
Clinicopathological	57.4%	45.5%	46.2%	51.4%	40.1%	**37.9**%
Proteomics	26.1%	32.2%	26.5%	20.0%	**19.1**%	20.7%
Fused modalities	63.4%	50.0%	42.6%	38.9%	34.7%	**17.9**%

6.4.1 Results and metrics

6.4.1.1 Accuracy of models for subsequent visits

To begin with, we provide in Table 6.2 the patient-wise accuracy of the top-selected models. Notice that we restricted these tables to the models that were trained to find 10 clusters only for clarity. In general, models trained for fewer clusters presented similar or greater accuracy, up to 5-10% more. We observed that the presence of the clinicopathological database helps finding more stable-over-time clusters than the proteomics modality only. Intriguingly, the OvO mode for both types of architectures nearly halved the accuracy for the proteomics dataset. While these clusters might be of interest, they do not match our criterion of stability over time and so we discarded them. The logistic regression models generally found more accurate clusters over time, which we can attribute to the simplicity of the decision boundary, contrary to MLP models. Overall, the obtained clusters are not holding throughout time, however we are satisfied that their definition remains stable more than 50% of the time in most datasets and models.

6.4.1.2 Ambiguity of the filtered models for consensus

We then mixed the subsets of filtered models together to produce a consensus matrix per dataset and number of clusters. We provide in Table 6.3 the PAC score of all these subsets. It turned out that keeping 9 to 10 clusters was a good strategy to get better clear-cut clusters as input for consensus clustering, specifically in the case of the fused dataset where the increase from 5 to 10 clusters leveraged a nearly 50% drop in ambiguity.

With very few models in the proteomics dataset reaching the accuracy threshold, *i.e.* only some for the OvA Logistic Regression in Table 6.2, it was then chosen with our medical collaborators to not continue with this dataset.

6.4.2 Identifying phenogroups

6.4.2.1 Merging clusters

As we initially provided our medical collaborators with 10 clusters with little ambiguity for the clinicopathological dataset and the fused dataset, it was chosen to merge those stable clusters down to 5 to alleviate the burden of interpretation. However, in contrast to directly setting the pipeline parameters to 5 clusters, this allowed us a more controlled view of the merging process. In fact, finding 10 clusters allows the algorithm to find more fine-grained distinctions between the various natures of patients compared to 5 clusters. In this context, more clusters means than we can potentially distinguish patients that are part of 2 phenogroups at the same time from other patients that belong to only one of these phenogroups. With the lowest PAC score obtained for 10 clusters, we are confident that we have obtained stable clusters over time and multiple models.

We merged the clusters according to their similarity in terms of interpretation by our medical collaborators. For example, we merged clusters where patients had a high cardio-metabolic profile, despite slight variations in other variables deemed less important in the interpretation of all clusters.

6.4.2.2 Visualising the pipeline outputs

We provide the final heatmaps in Figure 6.3 for the clinicopathological dataset and Figure 6.4 for the fused dataset. These heatmaps are built using the subset of variables selected 95% of the time across all 120 models. This threshold is arbitrary for the visualisation quality, as most variables were selected many times, *e.g.* in Table 6.4.

To start with, we observe in Figure 6.3 a cluster where patients already suffer from moderate aortic stenosis, sometimes severe: cluster $\mathcal{C}_1$. This condition is translated with high gradient and blood pressure among the echocardiographic variables: the mean gradient of the aorta is the highest for some patients, as well as the Vpeak and the velocity time integral (vti). All other patients who did not have serious aortic stenosis at first are in the other clusters, with few exceptions. The second cluster $\mathcal{C}_2$ does not have any high characteristics: no high levels of inflammation, no high levels of cardiometabolic stress, or high insulin levels. Like the third cluster $\mathcal{C}_3$, it is mainly composed of men. However, the third cluster $\mathcal{C}_3$ displays some high echocardiographic variables despite a low aortic stenosis severity to start with. Furthermore, it corresponds to the bicuspid profile of the patients. A bicuspid valve is an aortic valve composed of two leaflets, whereas a tricuspid valve is composed of three leaflets. The mostly female cluster $\mathcal{C}_4$ contained a majority of patients with tricuspid valves. Although a subset of this cluster exhibits a high level of inflammation with the C-reactive protein or high glycemia, the echocardiographic properties of this cluster do not match the high blood pressure and velocities from clusters $\mathcal{C}_1$ and $\mathcal{C}_3$. Finally, the last cluster $\mathcal{C}_5$ comprises patients with some comobordities: high weight and high insulin levels. Some patients exhibit inflammation or glycemia, and their overall blood outflow is between the average and high levels.

Looking at Figure 6.4, some observed clusters remain. We recover again a cluster of patients with very high metabolic characteristics, *e.g.* high body mass index, waist, weight, and insulin levels: $\mathcal{C}_4$. However, we now have another cluster showing above average, though less intense,

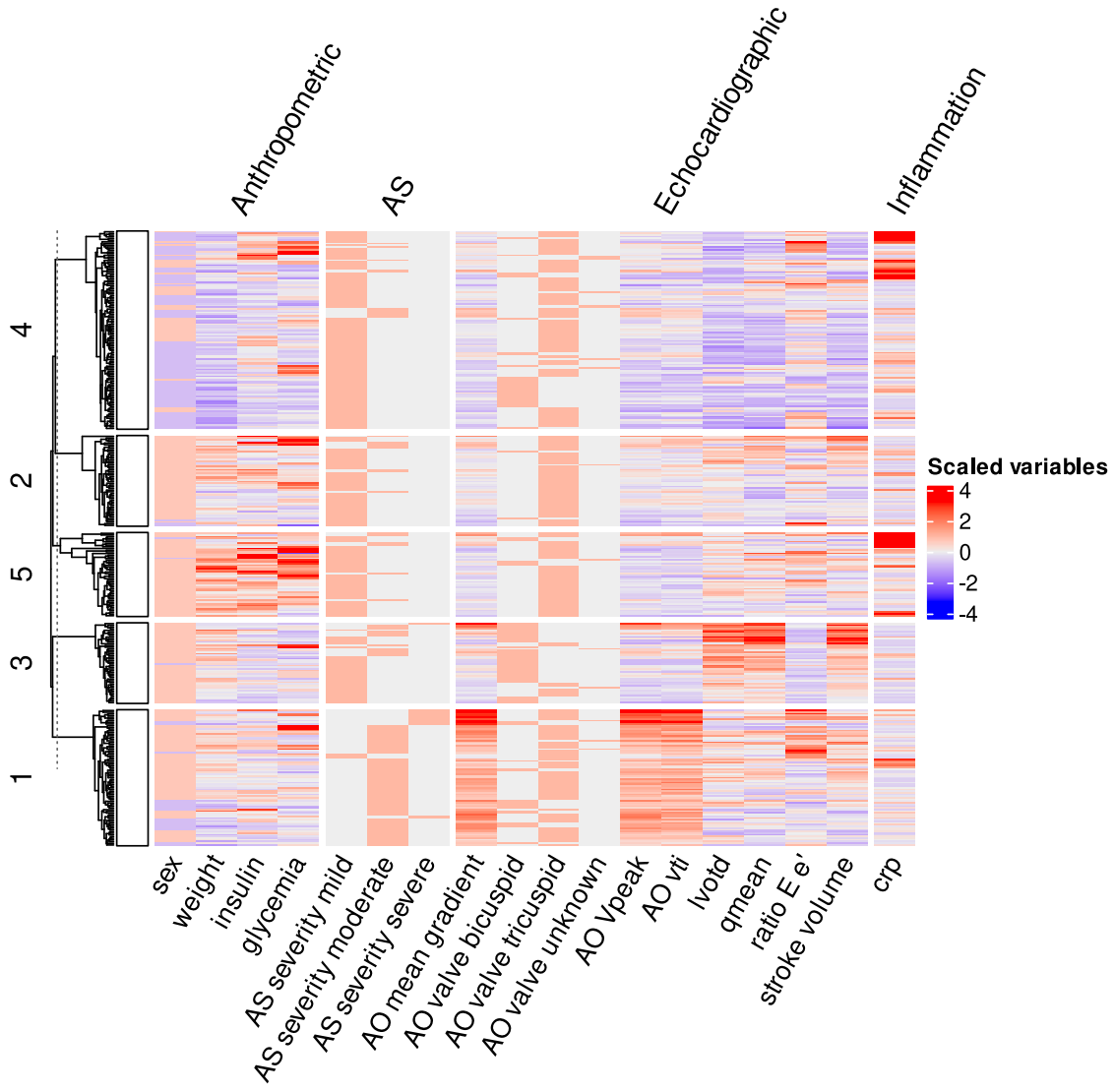


Figure 6.3 – Heatmap of the scaled top 10% selected input variables for the clinicopathological dataset. Samples on the rows are sorted according to the final hand-merged 5 clusters.

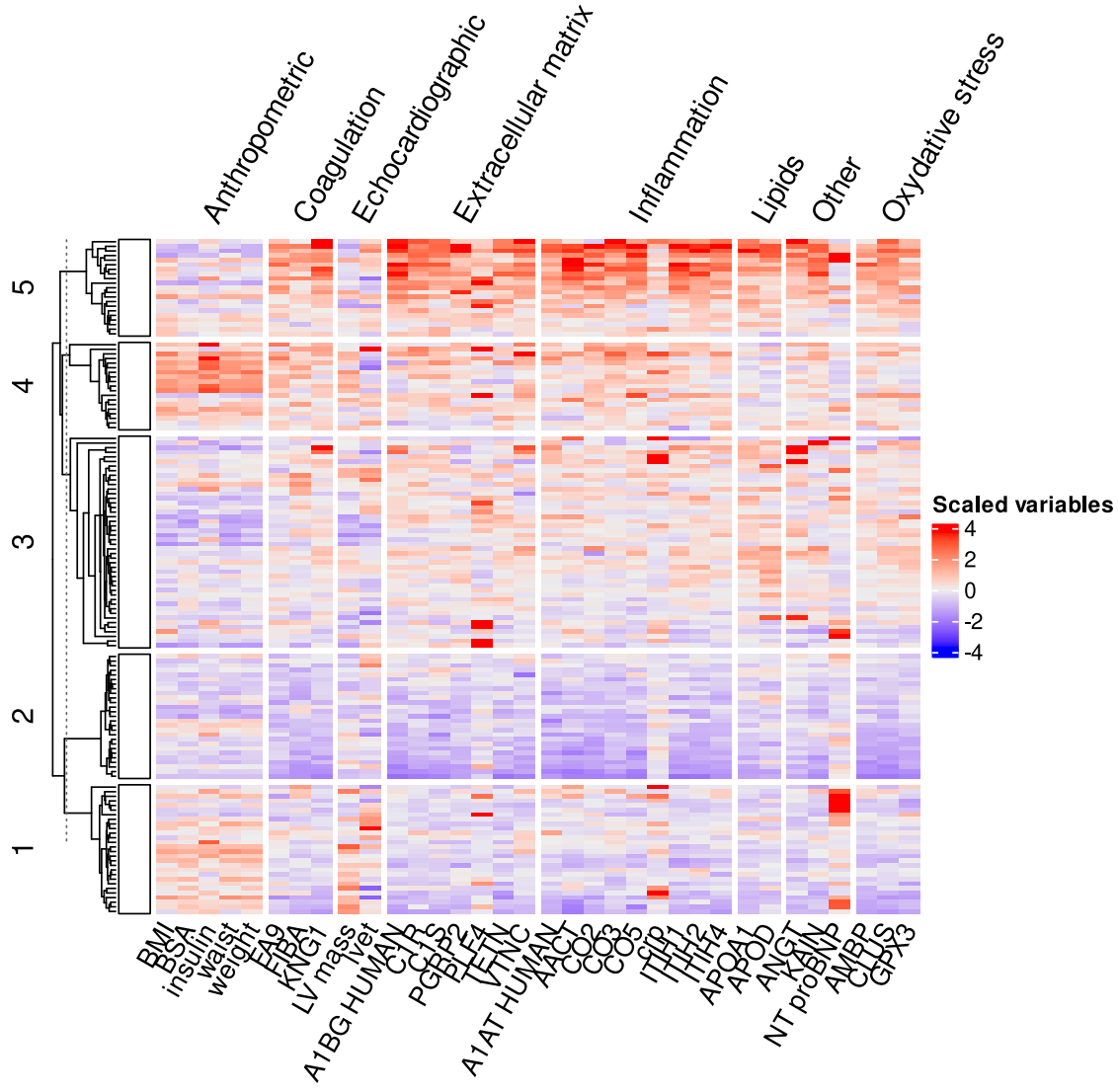


Figure 6.4 – Heatmap of the scaled top 10% selected input variables for the fused dataset. Samples on the rows are sorted according to the final hand-merged 5 clusters.

metabolic characteristics: $\mathcal{C}_1$. However, this cluster comes with rather low variables regarding all other categories except the echocardiographic variables. We observe again another cluster with low variables throughout all categories again, $\mathcal{C}_2$. Finally, the last cluster $\mathcal{C}_3$ and $\mathcal{C}_5$ present average to high values for all categories except metabolic and echocardiographic variables. They highlight groups of patients with notable inflammation, stress and coagulation necessities.

We chose to focus on the clustering of the clinicopathological dataset only, as the proposed heatmap from Figure 6.3 seemed more insightful to them than the fused case from Figure 6.4.

6.4.2.3 Characteristics comparisons

Beyond the subset of variables that were frequently selected by our set of clustering algorithms throughout the pipeline, clinical variables usually assessed by practitioners for aortic stenosis were evaluated per cluster and compared. We offer these comparisons of characteristics through Table 6.4 for the main clinical characteristics of the patients, Table 6.5 for the metabolic characteristics and Table 6.6 for the echocardiographic variables.

Different tests were used to compare the set of characteristics between all clusters. Continuous variables were tested for normality using the Shapiro-Wilk or the Kolmogorov-Smirnov tests and presented as mean ± standard deviation or as median and interquartile range if not normally distributed. If the variables were deemed normally distributed, we used a Student's t-test to evaluate differences between groups, otherwise we used the Mann-Whitney test. Categorical variables were expressed as the number of patients (per cent) and compared using the χ^2 or the Fisher's exact test. We report pairs of clusters for which the p-value was deemed significant following the usual threshold of 5%. We added to each variable a column indicating the number of times it was selected by an algorithm in the pipeline for the 120 models.

From these tables, we enumerate five profiles.

High-risk patients ($\mathcal{C}_1$) This cluster contains patients who present a high cardiovascular risk. They generally start or are close to an already moderate aortic stenosis severity (Table 6.6). Their echocardiographic profile exceeds the other clusters and presents hence a faster risk of aortic stenosis progression. These patients exhibit a strong presence of calcium compared to other clusters: they suffer more from calcific aortic stenosis (Lindman et al., 2016).

Young patients ($\mathcal{C}_3$) The cluster exhibits the youngest patients. These patients have a particularly good metabolic profile (Table 6.4) with small BMI, little hypertension and diabetes. This cluster is also characterised by the presence of bicuspid valve for few patients.

Female cluster ($\mathcal{C}_4$) This cluster is mostly characterised by the dominant presence of women (Table 6.4). Similarly to the previous cluster, these patients are in good physical condition. They present nonetheless a high level of cholesterol and inflammation through the c-reactive protein (Table 6.5).

The high cardio-metabolism cluster ($\mathcal{C}_5$) In contrast to the previous cluster, this cluster is mostly composed of male patients. These patients exhibit several comorbidities. Those include severe diabetes, visceral obesity, hypertension and an overall high-cardiometabolism (tables 6.4 and 6.5).

Table 6.4 – Comparison of the cluster attributes with respect to the patient characteristics based on the clinicopathological data clustering. When a P-value is deemed signficative, a superscript indicates the pair of clusters for which it holds. 1: $\mathcal{C}_1$-$\mathcal{C}_2$; 2: $\mathcal{C}_1$-$\mathcal{C}_3$; 3: $\mathcal{C}_1$-$\mathcal{C}_4$; 4: $\mathcal{C}_1$-$\mathcal{C}_5$; 5: $\mathcal{C}_2$-$\mathcal{C}_3$; 6: $\mathcal{C}_2$-$\mathcal{C}_4$; 7: $\mathcal{C}_2$-$\mathcal{C}_5$; 8: $\mathcal{C}_3$-$\mathcal{C}_4$; 9: $\mathcal{C}_3$-$\mathcal{C}_5$; 10: $\mathcal{C}_4$-$\mathcal{C}_5$.

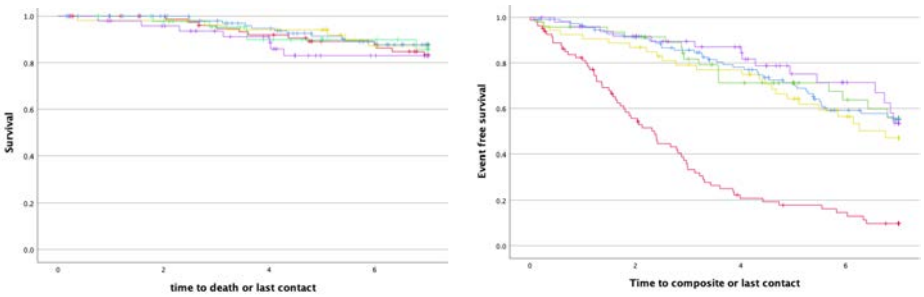
Variable	Cluster $\mathcal{C}_1$ $n = 81$	Cluster $\mathcal{C}_2$ $n = 54$	Cluster $\mathcal{C}_3$ $n = 47$	Cluster $\mathcal{C}_4$ $n = 117$	Cluster $\mathcal{C}_5$ $n = 50$	P-value	Selection Frequency
Age, years	69 [61 ; 75]	72 [63 ; 76]	47 [41 ; 60]	70 [58 ; 75]	69 [63 ; 72]	**<0.001** 2,5,8,9	85.0%
Height, cm	168 [162 ; 174]	171 [166 ; 175]	175 [171 ; 179]	159 [155 ; 165]	172 [167 ; 175]	**<0.001** 2-6,8,10	88.3%
Weight, kg	80.5 [73.3 ; 88.2]	86.3 [76.7 ; 94.9]	82.2 [76.6 ; 94.2]	67.6 [62.1 ; 74.0]	96.4 [83.4 ; 107.2]	**<0.001** 1,3,4,6-10	98.3%
BSA	1.92 [1.80 ; 2.01]	1.96 [1.87 ; 2.09]	1.97 [1.89 ; 2.15]	1.70 [1.61 ; 1.79]	2.10 [1.97 ; 2.20]	**<0.001** 1-4,6-8,10	84.2%
BMI	28.7 [25.7 ; 30.5]	30.1 [27.1 ; 32.1]	27.9 [25.1 ; 30.5]	26.4 [23.7 ; 29.2]	32.5 [29.6 ; 37.0]	**<0.001** 3-5,6,7,9,10	83.3%
Male sex, n (%)	63 (77)	51 (94)	47 (100)	36 (31)	48 (97)	**<0.001** 3,6,8,10	100%
Hypertension, n (%)	59 (72)	51 (94)	13 (28)	69 (59)	48 (97)	**<0.001** 2,5,6,8-10	93.3%
Diabetes, n (%)	15 (19)	19 (35)	3 (6)	21 (18)	28 (56)	**<0.001** 4,5,9,10	76.7%
CAD, n (%)	23 (28)	29 (54)	2 (4)	25 (21)	23 (46)	**<0.001** 1,2,5,6,9,10	75.5%
Previous MI, n (%)	15 (19)	19 (35)	2 (4)	9 (8)	11 (22)	**<0.001** 5,6	73.3%
History of AF, n (%)	9 (11)	8 (15)	2 (4)	18 (15)	10 (20)	0.192	74.2%
COPD, n (%)	7 (9)	4 (7)	2 (4)	8 (7)	8 (16)	0.258	74.2%
History of stroke or TIA, n (%)	7 (9)	5 (9)	2 (4)	8 (7)	0 (0)	0.258	73.3%
Bicuspid aortic valve, n (%)	16 (20)	2 (4)	36 (77)	28 (24)	5 (10)	**<0.001** 2,5,6,8,9	100%
NYHA >=3, n (%)	1 (1)	1 (2)	0 (0)	3 (3)	1 (2)	0.071	87.5%

Table 6.5 – Comparison of the cluster attributes with respect to the patient metabolic characteristics based on the clinicopathological data clustering. When a P-value is deemed significative, a superscript indicates the pair of clusters for which it holds. 1: $\mathcal{C}_1$-$\mathcal{C}_2$; 2: $\mathcal{C}_1$-$\mathcal{C}_3$; 3: $\mathcal{C}_1$-$\mathcal{C}_4$; 4: $\mathcal{C}_1$-$\mathcal{C}_5$; 5: $\mathcal{C}_2$-$\mathcal{C}_3$; 6: $\mathcal{C}_2$-$\mathcal{C}_4$; 7: $\mathcal{C}_2$-$\mathcal{C}_5$; 8: $\mathcal{C}_3$-$\mathcal{C}_4$; 9: $\mathcal{C}_3$-$\mathcal{C}_5$; 10: $\mathcal{C}_4$-$\mathcal{C}_5$.

Variable	Cluster $\mathcal{C}_1$ $n = 81$	Cluster $\mathcal{C}_2$ $n = 54$	Cluster $\mathcal{C}_3$ $n = 47$	Cluster $\mathcal{C}_4$ $n = 117$	Cluster $\mathcal{C}_5$ $n = 50$	P-value	Selection Frequency
Glycemia	5.3 [5.0 ; 6.0]	5.7 [5.2 ; 6.3]	5.2 [4.8 ; 5.8]	5.2 [4.8 ; 5.8]	6.4 [5.5 ; 7.6]	**<0.001** [1,4-7,9,10]	98.3%
Insulin	72 [46 ; 105]	76 [54 ; 116]	51 [38 ; 79]	58 [41 ; 102]	143 [92 ; 177]	**<0.001** [2,4-7,9,10]	100%
LDL	2.18 [1.90 ; 2.63]	1.91 [1.47 ; 2.49]	2.61 [2.18 ; 3.32]	2.42 [1.86 ; 3.06]	1.75 [1.34 ; 2.07]	**<0.001** [1-10]	90.8%
HDL	1.39 [1.19 ; 1.66]	1.32 [1.13 ; 1.50]	1.38 [1.19 ; 1.78]	1.54 [1.35 ; 1.81]	1.22 [1.02 ; 1.37]	**<0.001** [3,4,6,7,9,10]	75.8%
Total cholesterol	4.38 [3.76 ; 4.77]	3.94 [3.41 ; 4.48]	4.66 [4.12 ; 5.39]	4.70 [3.97 ; 5.55]	3.58 [3.26 ; 4.01]	**<0.001** [1-7,9,10]	93.3%
TG	1.27 [0.98 ; 1.66]	1.49 [0.80 ; 1.79]	1.01 [0.80 ; 1.33]	1.19 [0.77 ; 1.71]	1.45 [1.17 ; 1.98]	**0.005** [9,10]	44.2%
Apo A	1.51 [1.33 ; 1.67]	1.43 [1.29 ; 1.56]	1.56 [1.34 ; 1.69]	1.65 [1.44 ; 1.80]	1.35 [1.19 ; 1.47]	**<0.001** [3,4,6,9,10]	50.8%
Apo B	0.81 [0.73 ; 0.94]	0.77 [0.63 ; 0.95]	0.91 [0.75 ; 1.05]	0.86 [0.72 ; 1.07]	0.74 [0.67 ; 0.85]	**<0.001** [5,6,9,10]	63.3%
Creatinin	84 [74 ; 95]	87 [76 ; 104]	84 [76 ; 90]	72 [56 ; 85]	83 [76 ; 96]	**<0.001** [3,6,8,10]	85.0%
CRP	1.65 [0.80 ; 2.60]	0.98 [0.57 ; 2.54]	0.77 [0.42 ; 1.64]	2.27 [0.83 ; 4.71]	2.47 [1.14 ; 8.07]	**<0.001** [2,6-9]	100%

Table 6.6 – Comparison of the cluster attributes with respect to the patient echocardiographic/CT characteristic data based on the clinicopathological data clustering. When a P-value is deemed significant, a superscript indicates the pair of clusters for which it holds. The calcium scores, its deltas and other variables with NA indicated for the selection frequency were not part of the database version used for clustering. 1: $\mathcal{C}_1$-$\mathcal{C}_2$; 2: $\mathcal{C}_1$-$\mathcal{C}_3$; 3: $\mathcal{C}_1$-$\mathcal{C}_4$; 4: $\mathcal{C}_1$-$\mathcal{C}_5$; 5: $\mathcal{C}_2$-$\mathcal{C}_3$; 6: $\mathcal{C}_2$-$\mathcal{C}_4$; 7: $\mathcal{C}_2$-$\mathcal{C}_5$; 8: $\mathcal{C}_3$-$\mathcal{C}_4$; 9: $\mathcal{C}_3$-$\mathcal{C}_5$; 10: $\mathcal{C}_4$-$\mathcal{C}_5$.

Variable	Cluster $\mathcal{C}_1$ $n = 81$	Cluster $\mathcal{C}_2$ $n = 54$	Cluster $\mathcal{C}_3$ $n = 47$	Cluster $\mathcal{C}_4$ $n = 117$	Cluster $\mathcal{C}_5$ $n = 50$	P-value	Selection Frequency
Stroke volume	81.7 [73.0 ; 90.0]	80.1 [71.6 ; 90.9]	89.0 [82.2 ; 106.1]	67.6 [61.7 ; 74.9]	76.8 [72.8 ; 85.8]	**<0.001** [2,3,5,6,8-10]	95.0%
Stroke volume index	43.5 [38.1 ; 47.1]	40.2 [37.7 ; 44.3]	45.1 [41.6 ; 52.2]	39.7 [36.4 ; 44.6]	37.9 [34.4 ; 41.8]	**<0.001** [1-5,7-10]	68.3%
Vpeak	344 [314 ; 376]	262 [247 ; 289]	255 [234 ; 287]	251 [231 ; 276]	249 [235 ; 264]	**<0.001** [1-4]	99.2%
MG	28.2 [23.3 ; 33.1]	15.1 [13.3 ; 18.2]	15.2 [11.8 ; 20.0]	14.2 [11.6 ; 16.4]	13.6 [12.0 ; 16.9]	**<0.001** [1-4]	100% ?
AVA	0.97 [0.85 ; 1.08]	1.31 [1.11 ; 1.47]	1.48 [1.34 ; 1.70]	1.17 [1.04 ; 1.34]	1.43 [1.22 ; 1.60]	**<0.001** [1-5,8,10]	94.2%
AVA index	0.50 [0.46 ; 0.58]	0.65 [0.58 ; 0.74]	0.75 [0.67 ; 0.86]	0.67 [0.61 ; 0.78]	0.68 [0.56 ; 0.77]	**<0.001** [1-5,9]	90.0%
LV mass	200 [168 ; 249]	213 [185 ; 239]	224 [191 ; 250]	159 [131 ; 182]	219 [195 ; 257]	**<0.001** [3,6,8,10]	92.5%
LVEF	65 ± 5	63 ± 6	63 ± 5	67 ± 5	63 ± 6	**<0.001** [6,8,10]	49.2%
AS severity ≥ moderate, n (%)	11 (14)	0 (0)	1 (2)	0 (0)	0 (0)	**<0.001** [1-4]	99.2%
Calcium score (UA)	1257 [806 ; 1837]	739 [537 ; 1053]	708 [281 ; 1480]	374 [163 ; 588]	358 [230 ; 731]	**<0.001** [3,4,6,8]	
Calcium volume	1005 [638 ; 1431]	620 [457 ; 875]	549 [246 ; 1137]	328 [147 ; 487]	354 [202 ; 602]	**<0.001** [3,4,6,8]	NA
Delta Vpeak	22 [9 ; 39]	11 [3 ; 19]	11 [3 ; 22]	7 [3 ; 17]	8 [0 ; 14]	**<0.001** [1,3,4]	NA
Delta SV	-5.0 [-12.5 ; 2.0]	-1.0 [-9.0 ; 5.0]	4.0 [-6.0 ; 14.8]	-1.5 [-7.0 ; 4.1]	-1.0 [-7.0 ; 9.0]	**0.006** [2]	NA
Delta MG	4.4 [2.0 ; 7.7]	1.4 [0.4 ; 3.2]	1.3 [0.1 ; 2.9]	1.1 [0.4 ; 2.7]	1.0 [0.3 ; 2.3]	**<0.001** [1-4]	NA
Delta AVA	-0.08 [-0.13 ; -0.04]	-0.05 [-0.08 ; -0.01]	-0.05 [-0.09 ; -0.01]	-0.04 [-0.08 ; -0.01]	-0.05 [-0.09 ; 0.01]	**0.002** [1-3]	NA
Delta calcium score	213 [111 ; 307]	93 [35 ; 140]	102 [24 ; 311]	64 [11 ; 121]	61 [24 ; 156]	**0.001** [1,3,4]	NA
Delta calcium volume	156 [89 ; 244]	70 [30 ; 107]	80 [22 ; 230]	53 [12 ; 95]	52 [21 ; 124]	**<0.001** [1,3,4]	NA



(a) Mortality (p-value = 0.829) (b) Composite endpoint (p-value < 0.001)

Figure 6.5 – Kaplan-Meier curves of the clusters for two clinical endpoints using the clusters of the clinicopathological database from PROGRESSA. Red: $\mathcal{C}_1$ (high-risk patients), yellow: $\mathcal{C}_2$ (remaining patients), green: $\mathcal{C}_3$ (young patients), blue: $\mathcal{C}_4$ (female patients), purple: $\mathcal{C}_5$ (high cardiometabolism patients).

The remaining patients ($\mathcal{C}_2$) This cluster is not clearly defined by any specific characteristics overall. It simply contains the remaining patients that did not match any other clusters.

Similar analysis as tables 6.4, 6.5 and 6.6 were conducted on the proteomics and fused datasets. However, their insights either covered again what was shown with the clinicopathological database in a least significant manner, or were not considered informative at all.

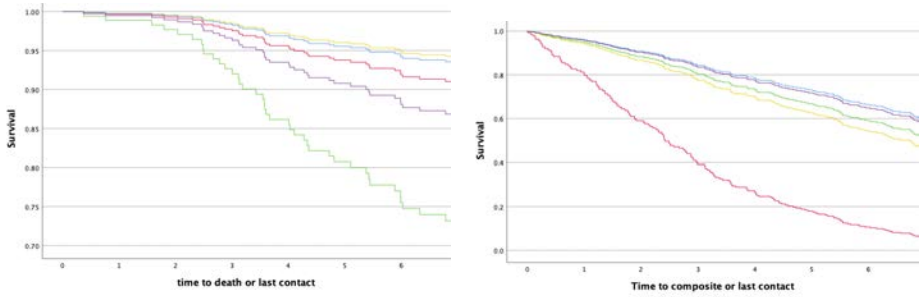
6.4.2.4 Endpoints of the clusters

We finally evaluated the endpoints of the clusters using Kaplan-Meier curves in Figure 6.5 and Cox regressions in Figure 6.6 obtained with STAT version 17.0 (StatCorp, College Station, Texas). The endpoints are markers of the disease progression. Different endpoints were considered: either the death, or composite endpoints that encompass both death and aortic valve replacement. For Kaplan-Meier curves, a log-rank test was performed. For the Cox regression, Wald tests are performed (Bradburn, Clark, Love, & Altman, 2003). We did not run a study of the endpoints of the fused case because we discarded it at this step of the analysis.

Despite a non-significant result with high p-value in Figure 6.5a, other figures highlight clusters with higher risks of endpoints with time passing. Notably, composite endpoints, which comprises surgical aortic valve replacement, are accentuated for the cluster $\mathcal{C}_1$ both in the Kaplan-Meier and Cox curves of figures 6.5b and 6.6b. We must note here that this result is adjusted for both sex and age in Figure 6.6. This implies that the increased risk of composite endpoint for $\mathcal{C}_1$ holds independently of age and sex. Compared to the mortality rate, this sudden difference in cluster $\mathcal{C}_1$ can be justified by a tendency to favour aortic valve replacement among patients with high cardio-metabolism. If we consider only death as endpoint, then the cluster $\mathcal{C}_3$ of young patients present an increased risk in Figure 6.6a, which could be imputed to the dominant bicuspid nature of their aortic valve (M. Shen et al., 2020).

6.5 Conclusion

We constructed in this chapter a pipeline using multiple GEMINI models to simultaneously cluster and select a subset of informative variables. The pipeline is concluded with a consensus



(a) Mortality (p-value < 0.001) (b) Composite endpoint (p-value < 0.001)

Figure 6.6 – Cox models of the clusters for two clinical endpoints adjusted for the age and the sex of patients using the clusters of the clinicopathological database from PROGRESSA. Red: $\mathcal{C}_1$ (high-risk patients), yellow: $\mathcal{C}_2$ (remaining patients), green: $\mathcal{C}_3$ (young patients), blue: $\mathcal{C}_4$ (female patients), purple: $\mathcal{C}_5$ (high cardiometabolism patients).

clustering algorithm using GEMINI. We applied this pipeline to 3 different subsets of the PROGRESSA dataset: the clinicopathological database, the proteomics database and their fusion. Our results converged to 5 clusters using only the clinicopathological database as others were deemed less interesting. Among the clusters, 4 distinct profiles were found: young patients with a bicuspid valve which exhibits an increased mortality progression regardless of age and sex; high-risk patients starting with mild or moderate aortic stenosis with increased risk of composite endpoint, *i.e.* death or aortic valve replacement, female patients in good condition with inflammation and no particular risk progression; and finally high cardio-metabolic patients with obesity, diabetes and hypertension without a significant risk progression. Although these phenogroups differ from the clinical hypotheses, they illustrate distinct categories of population for which the progression risk can be significant.

CHAPTER 7

Final words, ongoing and future works

7.1 Overview of the thesis contributions

7.1.1 Discriminative clustering

In this thesis, we brought up a complete framework for discriminative clustering using a distance-based and information-theoretic objective: the generalised mutual information (GEMINI). We demonstrated how this objective can be compatible with various clustering distributions, data distributions, and distances. We also extended this objective to incorporate joint feature selection to obtain subsets of features relevant to the clusters being discovered. We showed as well how GEMINI objective could be altered to deliver tractable gains in unsupervised tree constructions, allowing thus an interpretable model in clustering equivalent to kernel K-means in performances. All these major contributions are wrapped up in package named *GemClus* which facilitates the reproduction of most of our experiments.

7.1.2 Phenogroups of aortic stenosis

Building up on GEMINI, we created a complete pipeline for finding phenogroups of aortic stenosis among patients from the PROGRESSA study. We identified four profiles of patients: young patients with bicuspid valve who exhibit an increased progression of mortality regardless of age and sex; high-risk patients starting with an already mild or moderate aortic stenosis with an increased risk of composite endpoint, *i.e.* death or valvular replacement, women in good condition with inflammation and no particular risk progression; and finally, high-cardiometabolic patients with obesity, diabetes, and hypertension without a significant risk progression.

7.2 Ongoing work

7.2.1 Adding supervision to consensus clustering

We presented how GEMINI can be used for consensus clustering by viewing the consensus matrix as a distance and using the non-parametric model, which does not require positions of the samples. However, we can imagine better types of consensus clustering. Notably, we can add must-link and cannot-link constraints in the clustering algorithms (Wagstaff & Cardie, 2000). These constraints simply consist in telling, according to some expert, whether some samples should necessarily be together or apart. This type of supervision is different from giving labels as it only focuses on the interaction between the samples.

To implement must-link and cannot-link constraints, we propose to simply use gradient descent with the addition of regularisation in the objective function. In fact, the output of the model is a stochastic vector $\boldsymbol{y}$ of dimension K. A must-link constraint can be seen as minimising the distance between two predictions $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, while a cannot-link constraint can be seen as maximising the distance between those vectors. Let $\mathcal{T}_{\text{ML}} = \{(a_i, b_i)\}$ be the set of pairs that obey the must-link constraint and $\mathcal{T}_{\text{CL}}$ be the set of pairs that obey the cannot-link constraint. The new optimisation problem becomes:

$$\theta^\star = \arg\max_\theta \mathcal{I}_D\left(\boldsymbol{x}; y\right) + \lambda \sum_{(i,j)\in\mathcal{T}_{\text{CL}}} \|\psi_\theta(\boldsymbol{x}_i) - \psi_\theta(\boldsymbol{x}_j)\|_2^2 - \omega \sum_{(i,j)\in\mathcal{T}_{\text{ML}}} \|\psi_\theta(\boldsymbol{x}_i) - \psi_\theta(\boldsymbol{x}_j)\|_2^2, \quad (7.1)$$

where λ and ω are the regularisation weights and ψ_θ the underlying architecture of the distribution $p_\theta(y|\boldsymbol{x})$. This constraint is currently proposed experimentally in GemClus under the name `add_ml_cl_constraint`. Of course, this principle of constraints can be extended to all discriminative models, and not just the non-parametric model in the case of consensus clustering.

7.2.2 Towards heterogenous source clustering

We initially motivated discriminative clustering with the use of neural network producing latent representations of similar dimensions at the beginning of Chapter 2.

Assume that we have S sources of data. For all of these sources, the data spaces are different. We denote these spaces $\mathcal{X}^s, 1 \leq s \leq S$. The complete data space is $\mathcal{X} = \prod_{s=1}^S \mathcal{X}^s$. Let $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$ be a dataset with $\boldsymbol{x}_i \in \mathcal{X}$. We denote $\boldsymbol{x}_i^{(s)}$ the part of $\boldsymbol{x}_i$ that comes from the source space $\mathcal{X}^s$.

We are interested in building a complete model that is able to cluster the dataset $\mathcal{D}$:

$$p_\theta(y|\boldsymbol{x}) = \text{Categorical}(\text{Softmax} \circ \psi_\theta(\boldsymbol{x})), \quad (7.2)$$

with parameters θ and $y \in [K]$ the discrete cluster membership assigned to $\boldsymbol{x} \in \mathcal{X}$. Following our previous chapters, we want to optimise the parameters θ to maximise a GEMINI. However, GEMINI requires a metric in the data space $\mathcal{X}$ to work properly, yet the context of heterogeneous sources makes it challenging (Kan et al., 2020). Overall, it is easier to consider one relevant metric per data source, yet this does not completely solve the global picture because we would then need to think of linear coefficients (or more complex) to link all sub-metrics into one, which remains challenging. For instance, in the context of multi-omics, a mixture of kernels (Mariette & Villa-Vialaneix, 2017) could be used for the MMD-GEMINI.

We propose instead to first cluster each source individually using GEMINI, then derive from each model a common latent representation which is comparable across all sources and that we can concatenate. Let $p_{\omega^s}(y^s|\boldsymbol{x}^{(s)})$ be the clustering distribution of the source $\mathcal{X}^s$. This distribution is categorical and based on the composition of two consecutive functions, one for projecting into a latent space $g_{\omega_1^s} : \mathcal{X}^s \mapsto \mathcal{Z}_s = \mathbb{R}^{d_s}$ and another for projecting the latent space into the clustering space: $h_{\omega_2^s} : \mathcal{Z}^s \mapsto \mathbb{R}^K$. All individual models therefore take the form:

$$p_{\omega^s}(y^{(s)}|\boldsymbol{x}^{(s)}) = \text{Categorical}\left(\text{Softmax} \circ h_{\omega_2^s} \circ g_{\omega_1^s}(\boldsymbol{x}^{(s)})\right). \quad (7.3)$$

All individual functions $\{g_{\omega_1^s}\}_{s=1}^S$ end in a latent space $\mathbb{R}^{d_s}$ that we can concatenate. We can design the global latent representation space $\mathcal{Z} = \prod_{s=1}^S \mathcal{Z}^s = \mathbb{R}^D$ with $D = \sum_{s=1}^S d_s$. The multi-source clustering function becomes:

$$\psi_\theta(\boldsymbol{x}) = f_\beta \circ \text{Concat}\left[\{g_{\omega_1^s}(\boldsymbol{x}^{(s)})\}_{s=1}^S\right], \tag{7.4}$$

where f_β is a final clustering function with input space $\mathcal{Z}$. Parameters θ contain the sub-parameters $\{\beta, \omega_1^1, \cdots, \omega_1^S\}$. Note that the global model completely omits the clustering heads of each individual source $h_{\omega_2^s}$.

With the means of sub-clustering for GEMINI optimisation, it follows that learning can only be twofold:

1. We start by training all sub-clustering models $\{p_{\omega^s}\}_{s=1}^S$ independently.
2. We finish by optimising only β in p_θ to get a final clustering. All other weights are frozen.

Indeed, due to the nature of the gradient of GEMINI, optimising on all of θ at once would imply that sub-clustering parameters ω_1^s receive gradient information coming from other sources s'. Hence, optimising θ at once violates the idea that each projection of a source into the same latent space is independent of other sources.

The question of the number of layers in each sub-neural network $g_{\omega_1^s}$ is critical. Indeed, as we want to cluster samples using all sources, we need to make sure that the representation learnt per source preserves the structure of the data yet remains sufficiently different to justify the usage of neural network. If the learnt representation remains identical to the original input source $\mathcal{X}_s$, then the neural network $g_{\omega_1^s}$ does not need many layers. We previously used GEMINI in conjunction with the Euclidean distance and so it is interesting to know whether this guiding metric affects the representation quality.

7.3 Perspectives and future work

Our search for phenogroups of aortic stenosis used mainly early fusion of datasets to tackle the multi-modality problem. However, there remains a different source: the radiomics modality that contains images and videos. There are also more extensions of GEMINI algorithm that we can bring up to improve on this thesis.

7.3.1 Integration of radiomics in PROGRESSA clustering

Using the principle previously described for heterogeneous clustering, the next step is to include videos from the PROGRESSA dataset into the construction of the clusters. We could expect for example that the addition of videos bring a distinctive feature among the uncharacterised cluster of patients. We could also expect to maintain the same set of clusters using novel distinctive features. However, for all patients, the set of videos corresponds only to the first visit ever. This means that we need to summarise a set of videos to a single feature vector that can be combined with the features derived from all other modalities. This can for instance be achieved using PointNet-like or Deep Sets architectures (Qi, Su, Mo, & Guibas, 2017 ; Zaheer et al., 2017) that are designed to process clouds of points. This architecture design is currently under reflection.

7.3.2 Exploiting pretrained model for clustering

Looking back to the early experiments with GEMINI, we showed how we could use features derived from a SIMCLR model (*i.e.* an unsupervised model) to define a custom kernel or distance

that was sufficiently insightful on the data distribution to better guide the GEMINI clustering. As a future step, it would be interesting to adapt the notion of transfer learning to GEMINI where the features used for a kernel or distance could be extracted from a pretrained supervised model, which is adequate for the data. This would probably help to build clusters that share some properties with the initial classes used for the pretrained model.

7.3.3 Learning metrics

Finally, we believe that GEMINI could be inverted to address the ill-posed problem of *metric learning* in a supervised context. Indeed, during this entire thesis we discussed the maximisation of the output of a prediction model for a fixed affinity matrix. In contrast, we could use labels from a supervised dataset to provide a fixed distribution in GEMINI and let the affinity matrix be optimised through gradient descent. In doing so, we would leverage a model that learns features that are insightful enough to provide an accurate metric between samples. However, additional regularisations would be required to avoid trivial solutions.

Bibliography

Abeshouse, A., Ahn, J., Akbani, R., Ally, A., Amin, S., Andry, C. D., … Arora, A. (2015). The Molecular Taxonomy of Primary Prostate Cancer. *Cell*, *163*(4), 1011–1025. (Publisher: Elsevier)

Ahmad, T., Lund, L. H., Rao, P., Ghosh, R., Warier, P., Vaccaro, B., … Desai, N. R. (2018). Machine Learning Methods Improve Prognostication, Identify Clinically Distinct Phenotypes, and Detect Heterogeneity in Response to Therapy in a Large Cohort of Heart Failure Patients. *Journal of the American Heart Association*, *7*(8), e008081. (Publisher: American Heart Association)

Alelyani, S., Tang, J., & Liu, H. (2018). Feature Selection for Clustering: A Review. *Data Clustering*, 29–60. (Publisher: Chapman and Hall/CRC)

Alemi, A. A., Fischer, I., Dillon, J. V., & Murphy, K. (2016). Deep Variational Information Bottleneck. In *International Conference on Learning Representations.*

Andell, P., Li, X., Martinsson, A., Andersson, C., Stagmo, M., Zöller, B., … Smith, J. G. (2017). Epidemiology of Valvular Heart Disease in a Swedish Nationwide Hospital-Based Register Study. *Heart*, *103*(21), 1696–1703. (Publisher: BMJ Publishing Group Ltd and British Cardiovascular Society)

Andrews, J. L., & McNicholas, P. D. (2014). Variable Selection for Clustering and Classification. *Journal of Classification*, *31*(2), 136–153. (Publisher: Springer)

Arjovsky, M., & Bottou, L. (2017). Towards Principled Methods for Training Generative Adversarial Networks. In *International Conference on Learning Representations.*

Arjovsky, M., Chintala, S., & Bottou, L. (2017, août). Wasserstein Generative Adversarial Networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 214–223). PMLR.

Arora, S., Ge, R., Liang, Y., Ma, T., & Zhang, Y. (2017, août). Generalization and Equilibrium in Generative Adversarial Nets (GANs). In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 224–232). PMLR.

Bach, D. S., Siao, D., Girard, S. E., Duvernoy, C., McCallister Jr, B. D., & Gualano, S. K. (2009). Evaluation of Patients with Severe Symptomatic Aortic Stenosis who do not Undergo Aortic Valve Replacement: The Potential Role of Subjectively Overestimated Operative Risk. *Circulation: Cardiovascular Quality and Outcomes*, *2*(6), 533–539. (Publisher: Am Heart Assoc)

Bach, F., Jenatton, R., Mairal, J., & Obozinski, G. (2012). Optimization with Sparsity-Inducing Penalties. *Foundations and Trends® in Machine Learning*, *4*(1), 1–106. (Publisher: Now Publishers, Inc.)

Bachman, P., Hjelm, R. D., & Buchwalter, W. (2019). Learning Representations by Maximizing Mutual Information Across Views. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 32). Curran Associates, Inc.

Back, M., & Larsson, S. C. (2020). Risk Factors for Aortic Stenosis. *e-Journal of Cardiology Practice*, *18*(11).

Banfield, J. D., & Raftery, A. E. (1993). Model-Based Gaussian and Non-Gaussian Clustering. *Biometrics*, 803–821. (Publisher: JSTOR)

Barber, D., & Agakov, F. V. (2003). The IM Algorithm: A Variational Approach to Information Maximization. In *NIPS*.

Basak, J., & Krishnapuram, R. (2005). Interpretable Hierarchical Clustering by Constructing an Unsupervised Decision Tree. *IEEE transactions on knowledge and data engineering*, *17*(1), 121–132. (Publisher: IEEE)

Baudry, J.-P., Raftery, A. E., Celeux, G., Lo, K., & Gottardo, R. (2010). Combining Mixture Components for Clustering. *Journal of computational and graphical statistics*, *19*(2), 332–353. (Publisher: Taylor & Francis)

Bedi, P., & Sharma, C. (2016). Community Detection in Social Networks. *WIREs Data Mining and Knowledge Discovery*, *6*(3), 115–135. doi: https://doi.org/10.1002/widm.1178

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., & Hjelm, D. (2018, juillet). Mutual Information Neural Estimation. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning* (Vol. 80, pp. 531–540). PMLR.

Bellman, R. (1957). Dynamic Programming. *Press Princeton, New Jersey*.

Ben Ali, W., Pesaranghader, A., Avram, R., Overtchouk, P., Perrin, N., Laffite, S., … Hussin, J. G. (2021). Implementing Machine Learning in Interventional Cardiology: The Benefits Are Worth the Trouble. *Frontiers in Cardiovascular Medicine*, *8*, 711401. (Publisher: Frontiers)

Bertsimas, D., Orfanoudaki, A., & Wiberg, H. (2021, janvier). Interpretable Clustering: an Optimization Approach. *Machine Learning*, *110*(1), 89–138. doi: 10.1007/s10994-020-05896-2

Bertsimas, D., & Tsitsiklis, J. N. (1997). *Introduction to Linear Optimization* (Vol. 6). Athena scientific Belmont, MA.

Beyersdorf, F., Vahanian, A., Milojevic, M., Praz, F., Baldus, S., Bauersachs, J., … De Paulis, R. (2021). 2021 ESC/EACTS Guidelines for the Management of Valvular Heart Disease: Developed by the Task Force for the Management of Valvular Heart Disease of the European Society of Cardiology (ESC) and the European Association for Cardio-Thoracic Surgery (EACTS). *European Journal of Cardio-Thoracic Surgery*, *60*(4), 727–800. (Publisher: Oxford University Press)

Biernacki, C., Celeux, G., & Govaert, G. (2000). Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(7), 719–725. doi: 10.1109/34.865189

Biernacki, C., Marbac, M., & Vandewalle, V. (2021). Gaussian-Based Visualization of Gaussian and Non-Gaussian-Based Clustering. *Journal of Classification*, *38*, 129–157. (Publisher: Springer)

Billingsley, P. (1999). *Convergence of Probability Measures*. John Wiley & Sons.

Bishop, C. M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)* (1re éd.). Springer.

Blockeel, H., Raedt, L. D., & Ramon, J. (1998). Top-Down Induction of Clustering Trees. In *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 55–63).

Blum, A. L., & Langley, P. (1997). Selection of Relevant Features and Examples in Machine Learning. *Artificial intelligence*, *97*(1-2), 245–271. (Publisher: Elsevier)

Bock, H. (1994). Information and Entropy in Cluster Analysis. In *Proceedings of the First US/Japan Conference on the Frontiers of Statistical Modeling: An Informational Approach: Volume 2 Multivariate Statistical Modeling* (pp. 115–147). Springer.

Bode, L., Hanna, A., Yang, J., & Shah, D. V. (2015). Candidate Networks, Citizen Clusters, and Political Expression: Strategic Hashtag Use in the 2010 Midterms. *The ANNALS of the American Academy of Political and Social Science*, *659*(1), 149–165. doi: 10.1177/0002716214563923

Bohbot, Y., Raitière, O., Guignant, P., Ariza, M., Diouf, M., Rusinaru, D., … Tribouilloy, C. (2022, novembre). Unsupervised Clustering of Patients with Severe Aortic Stenosis: A Myocardial Continuum. *Archives of Cardiovascular Diseases*, *115*(11), 578–587. doi: 10.1016/j.acvd.2022.06.007

Bonneel, N., Van De Panne, M., Paris, S., & Heidrich, W. (2011). Displacement interpolation using Lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia conference* (pp. 1–12).

Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization Methods for Large-Scale Machine Learning. *SIAM review*, *60*(2), 223–311. (Publisher: SIAM)

Bouveyron, C., & Brunet, C. (2012). Simultaneous Model-Based Clustering and Visualization in the Fisher Discriminative Subspace. *Statistics and Computing*, *22*(1), 301–324.

Bouveyron, C., & Brunet-Saumard, C. (2014a, juin). Discriminative Variable Selection for Clustering with the Sparse Fisher-Em Algorithm. *Computational Statistics*, *29*(3), 489–513. doi: 10.1007/s00180-013-0433-6

Bouveyron, C., & Brunet-Saumard, C. (2014b). Model-Based Clustering of High-Dimensional Data: A Review. *Computational Statistics & Data Analysis*, *71*, 52–78. doi: https://doi.org/10.1016/j.csda.2012.12.008

Bouveyron, C., Celeux, G., Murphy, T. B., & Raftery, A. E. (2019). *Model-Based Clustering and Classification for Data Science: With Applications in R*. Cambridge University Press.

Bouveyron, C., Latouche, P., & Zreik, R. (2018). The Stochastic Topic Block Model for the Clustering of Vertices in Networks with Textual Edges. *Statistics and Computing*, *28*(1), 11–31. (Publisher: Springer)

Bradburn, M. J., Clark, T. G., Love, S. B., & Altman, D. G. (2003, août). Survival Analysis Part Ii: Multivariate Data Analysis – an Introduction to Concepts and Methods. *British Journal of Cancer*, *89*(3), 431–436. doi: 10.1038/sj.bjc.6601119

Bramon, R., Boada, I., Bardera, A., Rodriguez, J., Feixas, M., Puig, J., & Sbert, M. (2011). Multimodal Data Fusion Based on Mutual Information. *IEEE Transactions on Visualization and Computer Graphics*, *18*(9), 1574–1587. (Publisher: IEEE)

Breiman, L. (1984). Classification and Regression Trees. *The Wadsworth & Brooks/Cole*. (Publisher: Advanced Books & Software)

Bridle, J., Heading, A., & MacKay, D. (1992). Unsupervised Classifiers, Mutual Information and 'Phantom' Targets. In J. Moody, S. Hanson, & R. P. Lippmann (Eds.), *Advances in Neural Information Processing Systems* (Vol. 4). Morgan-Kaufmann.

Brown, G. (2004). *Diversity in Neural Network Ensembles* (PhD Thesis). University of Birmingham.

Burghardt, E., Sewell, D., & Cavanaugh, J. (2022). Agglomerative and Divisive Hierarchical Bayesian Clustering. *Computational Statistics & Data Analysis*, *176*, 107566. (Publisher: Elsevier)

Caglar, U. (2014). *Divergence And Entropy Inequalities For Log Concave Functions* (PhD Thesis). Case Western Reserve University.

Capoulade, R., Clavel, M.-A., Dumesnil, J. G., Chan, K. L., Teo, K. K., Tam, J. W., ... Pibarot, P. (2012). Impact of Metabolic Syndrome on Progression of Aortic Stenosis: Influence of Age and Statin Therapy. *Journal of the American College of Cardiology*, *60*(3), 216–223. (Publisher: American College of Cardiology Foundation Washington, DC)

Capoulade, R., Després, J.-P., Mathieu, P., Clavel, M.-A., Dahou, A., Arsenault, M., ... Pibarot, P. (2012). *Excess Visceral Adiposity Is Related to Left Ventricular Hypertrophy and Dysfunction in Patients with Aortic Stenosis-Results from the Progressa Study.* Am Heart Assoc.

Capoulade, R., Mahmut, A., Tastet, L., Arsenault, M., Bédard, E., Dumesnil, J. G., ... Bossé, Y. (2015). Impact of Plasma Lp-PLA2 Activity on the Progression of Aortic Stenosis: the PROGRESSA Study. *JACC: Cardiovascular Imaging*, *8*(1), 26–33. (Publisher: American College of Cardiology Foundation Washington, DC)

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., & Joulin, A. (2020). Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (Vol. 33, pp. 9912–9924). Curran Associates, Inc.

Carreira-Perpinan, M. A., & Tavallali, P. (2018). Alternating Optimization of Decision Trees, with Application to Learning Sparse Oblique Trees. *Advances in Neural Information Processing Systems*, *31*.

Casquilho, J. A., & Österreicher, F. (2018). On the Gini–Simpson Index and Its Generalisation—a Historic Note. *South African Statistical Journal*, *52*(2), 129–137. (Publisher: South African Statistical Association (SASA))

Celeux, G., Martin-Magniette, M.-L., Maugis-Rabusseau, C., & Raftery, A. E. (2014). Comparing Model Selection and Regularization Approaches to Variable Selection in Model-Based Clustering. *Journal de la Société Française de Statistique*, *155*(2), 57–71. (Publisher: Société Française de Statistique)

Chang, S., Kim, H., Suh, Y. J., Choi, D. M., Kim, H., Kim, D. K., ... Choi, B. W. (2021). Development of a Deep Learning-Based Algorithm for the Automatic Detection and Quantification of Aortic Valve Calcium. *European Journal of Radiology*, *137*, 109582. (Publisher: Elsevier)

Chen, H., Wang, W., Feng, X., & He, R. (2018). Discriminative and Coherent Subspace Clustering. *Neurocomputing*, *284*, 177–186. (Publisher: Elsevier)

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning* (pp. 1597–1607). PMLR.

Chlap, P., Min, H., Vandenberg, N., Dowling, J., Holloway, L., & Haworth, A. (2021). A Review of Medical Image Data Augmentation Techniques for Deep Learning Applications. *Journal of Medical Imaging and Radiation Oncology*, *65*(5), 545–563. (Publisher: Wiley Online Library)

Coffey, S., Roberts-Thomson, R., Brown, A., Carapetis, J., Chen, M., Enriquez-Sarano, M., ... Prendergast, B. D. (2021). Global Epidemiology of Valvular Heart Disease. *Nature Reviews Cardiology*, *18*(12), 853–864. (Publisher: Nature Publishing Group UK London)

Cohen-Shelly, M., Attia, Z. I., Friedman, P. A., Ito, S., Essayagh, B. A., Ko, W.-Y., ... Carter, R. E. (2021). Electrocardiogram Screening for Aortic Valve Stenosis Using Artificial Intelligence. *European Heart Journal*, *42*(30), 2885–2896. (Publisher: Oxford University Press)

Corduneanu, A., & Jaakkola, T. (2002). On Information Regularization. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence* (pp. 151–158).

Cover, T. M. (1999). *Elements of Information Theory*. John Wiley & Sons.

Csiszár, I. (1967). Information-Type Measures of Difference of Probability Distributions and Indirect Observation. *Studia Scientiarum Mathematicarum Hungarica*, *2*, 229–318.

Damluji, A. A., Fabbro, M., Epstein, R. H., Rayer, S., Wang, Y., Moscucci, M., ... Resar, J. R. (2020). Transcatheter Aortic Valve Replacement in Low-Population Density Areas: Assessing Healthcare Access for Older Adults with Severe Aortic Stenosis. *Circulation: Cardiovascular Quality and Outcomes*, *13*(8), e006245. (Publisher: Am Heart Assoc)

Dang, Z., Deng, C., Yang, X., Wei, K., & Huang, H. (2021). Nearest Neighbor Matching for Deep Clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 13693–13702).

Danielsen, R., Aspelund, T., Harris, T. B., & Gudnason, V. (2014). The Prevalence of Aortic Stenosis in the Elderly in Iceland and pPedictions for the Coming Decades: The AGES–Reykjavík Study. *International Journal of Cardiology*, *176*(3), 916–922. (Publisher: Elsevier)

Davies, D. L., & Bouldin, D. W. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-1*(2), 224–227. doi: 10.1109/TPAMI.1979.4766909

Defays, D. (1977). An Efficient Algorithm for a Complete Link Method. *The Computer Journal*, *20*(4), 364–366. (Publisher: Oxford University Press)

Dhillon, I. S., Guan, Y., & Kulis, B. (2004). Kernel K-Means: Spectral Clustering and Normalized Cuts. In *Proceedings of the Tenth Acm Sigkdd International Conference on Knowledge Discovery and Data Mining* (pp. 551–556).

Dhillon, I. S., Mallela, S., & Kumar, R. (2003). A Divisive Information Theoretic Feature Clustering Algorithm for Text Classification. *JMLR*, *3*, 1265–1287. (Publisher: JMLR. org)

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., & Shanahan, M. (2016). Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. *arXiv preprint arXiv:1611.02648*.

Do, K., Tran, T., & Venkatesh, S. (2021). Clustering by Maximizing Mutual Information Across Views. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 9928–9938).

Dy, J. G. (2007). Unsupervised Feature Selection. In *Computational Methods of Feature Selection* (pp. 35–56). Chapman and Hall/CRC.

Edwards, A. W., & Cavalli-Sforza, L. L. (1965). A Method for Cluster Analysis. *Biometrics*, 362–375. (Publisher: JSTOR)

Elkan, C. (2003). Using the Triangle Inequality to Accelerate K-Means. In *Proceedings of the 20th International Conference on Machine Learning (icml-03)* (pp. 147–153).

Estivill-Castro, V. (2002). Why so Many Clustering Algorithms: A Position Paper. *Acm Sigkdd Explorations Newsletter*, *4*(1), 65–75. (Publisher: ACM New York, NY, USA)

Fang, L., Jennings, A., Wen, W., Li, K.-Q., & Li, T. (1991). Unsupervised Learning for Neural Trees. In *IEEE International Joint Conference on Neural Networks* (pp. 2709–2715). IEEE.

Fatima, B., Mohananey, D., Khan, F. W., Jobanputra, Y., Tummala, R., Banerjee, K., ... Blackstone, E. (2019). Durability Data for Bioprosthetic Surgical Aortic Valve: A Systematic Review. *JAMA cardiology*, *4*(1), 71–80. (Publisher: American Medical Association)

Feeny, A. K., Rickard, J., Patel, D., Toro, S., Trulock, K. M., Park, C. J., ... Sinha, S. (2019). Machine Learning Prediction of Response to Cardiac Resynchronization Therapy: Improvement Versus Current Guidelines. *Circulation: Arrhythmia and Electrophysiology*, *12*(7), e007316. (Publisher: Am Heart Assoc)

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., ... Vayer, T. (2021). POT: Python Optimal Transport. *Journal of Machine Learning Research*, *22*(78), 1–8.

Fop, M., & Murphy, T. B. (2018). Variable Selection Methods for Model-Based Clustering. *Statistics Surveys*, *12*, 18–65. (Publisher: Amer. Statist. Assoc., the Bernoulli Soc., the Inst. Math. Statist., and the ...)

Fraiman, R., Ghattas, B., & Svarc, M. (2013). Interpretable Clustering using Unsupervised Binary Trees. *Advances in Data Analysis and Classification*, *7*, 125–145. (Publisher: Springer)

França, G., Rizzo, M. L., & Vogelstein, J. T. (2020). Kernel K-Groups via Hartigan's Method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(12), 4411–4425. (Publisher: IEEE)

Frost, N., Moshkovitz, M., & Rashtchian, C. (2020). ExKMC: Expanding Explainable k-Means Clustering. *arXiv preprint arXiv:2006.02399*.

Fruchterman, T. M., & Reingold, E. M. (1991). Graph Drawing by Force-Directed Placement. *Software: Practice and Experience*, *21*(11), 1129–1164. (Publisher: Wiley Online Library)

Gabidolla, M., & Carreira-Perpinan, M. A. (2022). Optimal Interpretable Clustering Using Oblique Decision Trees. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 400–410).

Gamlath, B., Jia, X., Polak, A., & Svensson, O. (2021). Nearly-Tight and Oblivious Algorithms for Explainable Clustering. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, & J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems* (Vol. 34, pp. 28929–28939). Curran Associates, Inc.

Garcea, F., Serra, A., Lamberti, F., & Morra, L. (2023). Data Augmentation for Medical Imaging: A Systematic Literature Review. *Computers in Biology and Medicine*, *152*, 106391. doi: https://doi.org/10.1016/j.compbiomed.2022.106391

Gardezi, S. K., Myerson, S. G., Chambers, J., Coffey, S., d'Arcy, J., Hobbs, F. R., ... Prendergast, A. (2018). Cardiac Auscultation Poorly Predicts the Presence of Valvular Heart Disease in Asymptomatic Primary Care Patients. *Heart*, *104*(22), 1832–1835. (Publisher: BMJ Publishing Group Ltd and British Cardiovascular Society)

Gini, C. W. (1912). Variability and Mutability, Contribution to the Study of Statistical Distributions and Relations. *Studi Economico-Giuridici della R. Universita de Cagliari*.

Gneiting, T., & Raftery, A. E. (2007). Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American statistical Association*, *102*(477), 359–378. (Publisher: Taylor & Francis)

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 27). Curran Associates, Inc.

Goodman, L. A. (1974). Exploratory Latent Structure Analysis Using Both Identifiable and Unidentifiable Models. *Biometrika*, *61*(2), 215–231. (Publisher: Oxford University Press)

Gouk, H., Frank, E., Pfahringer, B., & Cree, M. J. (2021, février). Regularisation of Neural Networks by Enforcing Lipschitz Continuity. *Machine Learning*, *110*(2), 393–416. doi: 10.1007/s10994-020-05929-w

Gray, R. M., & Shields, P. C. (1977). The Maximum Mutual Information Between Two Random Processes. *Information and Control*, *33*(4), 273–280. (Publisher: Elsevier)

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. (2012). A Kernel Two-Sample Test. *The Journal of Machine Learning Research*, *13*(1), 723–773.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On Calibration of Modern Neural Networks. In *International Conference on Machine Learning* (pp. 1321–1330). PMLR.

Généreux, P., Sharma, R. P., Cubeddu, R. J., Aaron, L., Abdelfattah, O. M., Koulogiannis, K. P., … Makkar, R. R. (2023). The Mortality Burden of Untreated Aortic Stenosis. *Journal of the American College of Cardiology*, *82*(22), 2101–2109. (Publisher: American College of Cardiology Foundation Washington DC)

Harchaoui, W. (2020). *Learning Representations using Neural Networks and Optimal Transport* (PhD Thesis). Université Paris Cité.

Harris, L., Fondrie, W. E., Oh, S., & Noble, W. S. (2023). Evaluating proteomics imputation methods with improved criteria. *Journal of Proteome Research*, *22*(11), 3427–3438. (Publisher: ACS Publications)

Hartley, A., Hammond-Haley, M., Marshall, D. C., Salciccioli, J. D., Malik, I. S., Khamis, R. Y., & Shalhoub, J. (2021). Trends in Mortality from Aortic Stenosis in Europe: 2000–2017. *Frontiers in Cardiovascular Medicine*, *8*, 748137. (Publisher: Frontiers Media SA)

Hastie, T., Tibshirani, R., & Wainwright, M. (2015). Statistical Learning with Sparsity. *Monographs on statistics and applied probability*, *143*, 143.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

He, X., Cai, D., & Niyogi, P. (2005). Laplacian Score for Feature Selection. *Advances in Neural Information Processing Systems*, *18*.

Held, M., & Buhmann, J. (1997). Unsupervised on-line Learning of Decision Trees for Hierarchical Data Analysis. *Advances in Neural Information Processing Systems*, *10*.

Hennig, C. (2015). What are the True Clusters? *Pattern Recognition Letters*, *64*, 53–62. doi: https://doi.org/10.1016/j.patrec.2015.04.009

Hernandez-Suarez, D. F., Kim, Y., Villablanca, P., Gupta, T., Wiley, J., Nieves-Rodriguez, B. G., … Sanina, C. (2019). Machine Learning Prediction Models for in-Hospital Mortality After Transcatheter Aortic Valve Replacement. *Cardiovascular Interventions*, *12*(14), 1328–1338. (Publisher: American College of Cardiology Foundation Washington DC)

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., ... Lerchner, A. (2017). beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations.*

Himelboim, I., Smith, M. A., Rainie, L., Shneiderman, B., & Espina, C. (2017). Classifying Twitter Topic-Networks Using Social Network Analysis. *Social Media + Society*, *3*(1), 2056305117691545. doi: 10.1177/2056305117691545

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. *arXiv preprint arXiv:1207.0580.*

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., & Bengio, Y. (2019). Learning Deep Representations by Mutual Information Estimation and Maximization. In *International Conference on Learning Representations.*

Hoff, P. D., Raftery, A. E., & Handcock, M. S. (2002). Latent Space Approaches to Social Network Analysis. *Journal of the American Statistical Association*, *97*(460), 1090–1098. (Publisher: Taylor & Francis)

Hofmann, T., Schölkopf, B., & Smola, A. J. (2008). Kernel Methods in Machine Learning. *The Annals of Statistics*, *36*(3), 1171–1220.

Hu, W., Miyato, T., Tokui, S., Matsumoto, E., & Sugiyama, M. (2017, août). Learning Discrete Representations via Information Maximizing Self-Augmented Training. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 1558–1567). PMLR.

Huang, J., Gong, S., & Zhu, X. (2020). Deep Semantic Clustering by Partition Confidence Maximisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8849–8858).

Hubert, L. (1973). Monotone Invariant Clustering Procedures. *Psychometrika*, *38*(1), 47–62. (Publisher: Springer)

Hubert, L., & Arabie, P. (1985). Comparing Partitions. *Journal of Classification*, *2*(1), 193–218. (Publisher: Springer)

Iung, B., Delgado, V., Rosenhek, R., Price, S., Prendergast, B., Wendler, O., ... Bogachev-Prokophiev, A. (2019). Contemporary Presentation and Management of Valvular Heart Disease: The Eurobservational Research Programme Valvular Heart Disease II Survey. *Circulation*, *140*(14), 1156–1169. (Publisher: American Heart Association)

Jabi, M., Pedersoli, M., Mitiche, A., & Ayed, I. B. (2019). Deep Clustering: On the Link Between Discriminative Models and K-means. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. doi: 10.1109/TPAMI.2019.2962683

Jang, E., Gu, S., & Poole, B. (2017). Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations.*

Ji, X., Henriques, J. F., & Vedaldi, A. (2019). Invariant Information Clustering for Unsupervised Image Classification and Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 9865–9874).

Jiang, Z., Zheng, Y., Tan, H., Tang, B., & Zhou, H. (2017). Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. In *Proceedings of the 26th International*

Joint Conference on Artificial Intelligence (pp. 1965–1972). AAAI Press. (Place: Melbourne, Australia)

John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant Features and the Subset Selection Problem. In *Machine learning proceedings 1994* (pp. 121–129). Elsevier.

Kan, S., Zhang, L., He, Z., Cen, Y., Chen, S., & Zhou, J. (2020, mars). Metric Learning-Based Kernel Transformer with Triplets and Label Constraints for Feature Fusion. *Pattern Recognition*, *99*, 107086. doi: 10.1016/j.patcog.2019.107086

Kansal, T., Bahuguna, S., Singh, V., & Choudhury, T. (2018). Customer Segmentation Using K-Means Clustering. In *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (ctems)* (pp. 135–139). IEEE.

Karakos, D., Khudanpur, S., Eisner, J., & Priebe, C. E. (2005). Unsupervised Classification via Decision Trees: An Information-Theoretic Perspective. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* (Vol. 5, pp. v–1081). IEEE.

Karlis, D., & Tsiamyrtzis, P. (2008). Exact Bayesian Modeling for Bivariate Poisson Data and Extensions. *Statistics and Computing*, *18*, 27–40. (Publisher: Springer)

Kashwan, K. R., & Velu, C. (2013). Customer Segmentation using Clustering and Data Mining Techniques. *International Journal of Computer Theory and Engineering*, *5*(6), 856. (Publisher: IACSIT Press)

Kaufman, L., & Rousseeuw, P. J. (1990). Finding Groups in Data. An Introduction to Cluster Analysis. *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics*.

Khan, K. R., Khan, O. A., Chen, C., Liu, Y., Kandanelly, R. R., Jamiel, P. J., ... Elmariah, S. (2023). Impact of Moderate Aortic Stenosis in Patients With Heart Failure With Reduced Ejection Fraction. *Journal of the American College of Cardiology*, *81*(13), 1235–1244. doi: https://doi.org/10.1016/j.jacc.2023.01.032

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N., & Welling, M. (2016). Variational Graph Auto-Encoders. *arXiv preprint arXiv:1611.07308*.

Kiselev, V. Y., Kirschner, K., Schaub, M. T., Andrews, T., Yiu, A., Chandra, T., ... Green, A. R. (2017). Sc3: Consensus Clustering of Single-Cell Rna-Seq Data. *Nature methods*, *14*(5), 483–486. (Publisher: Nature Publishing Group US New York)

Kleinberg, J. (2003). An Impossibility Theorem for Clustering. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in Neural Information Processing Systems* (Vol. 15). MIT Press.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Kong, Y., Deng, Y., & Dai, Q. (2015). Discriminative Clustering and Feature Selection for Brain MRI Segmentation. *IEEE Signal Processing Letters*, *22*(5), 573–577. doi: 10.1109/LSP.2014.2364612

Krause, A., Perona, P., & Gomes, R. (2010). Discriminative Clustering by Regularized Information Maximization. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), *Advances in Neural Information Processing Systems* (Vol. 23). Curran Associates, Inc.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*.

Kwak, S., Lee, Y., Ko, T., Yang, S., Hwang, I.-C., Park, J.-B., … Lee, S.-P. (2020, mai). Unsupervised Cluster Analysis of Patients With Aortic Stenosis Reveals Distinct Population With Different Phenotypes and Outcomes. *Circulation: Cardiovascular Imaging*, *13*(5), e009707. (Publisher: American Heart Association) doi: 10.1161/CIRCIMAGING.119.009707

Kwon, J.-M., Lee, S. Y., Jeon, K.-H., Lee, Y., Kim, K.-H., Park, J., … Lee, M.-M. (2020). Deep Learning–Based Algorithm for Detecting Aortic Stenosis using Electrocardiography. *Journal of the American Heart Association*, *9*(7), e014717. (Publisher: American Heart Association)

Laber, E., Murtinho, L., & Oliveira, F. (2023). Shallow Decision Trees for Explainable K-means Clustering. *Pattern Recognition*, *137*, 109239. (Publisher: Elsevier)

Lachmann, M., Rippen, E., Schuster, T., Xhepa, E., von, S. M., Pellegrini, C., … Joner, M. (2021, octobre). Subphenotyping of Patients With Aortic Stenosis by Unsupervised Agglomerative Clustering of Echocardiographic and Hemodynamic Data. *JACC: Cardiovascular Interventions*, *14*(19), 2127–2140. (Publisher: American College of Cardiology Foundation) doi: 10.1016/j.jcin.2021.08.034

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An Empirical Evaluation of Deep Architectures on Problems with many Factors of Variation. In *Proceedings of the 24th International Conference on Machine Learning* (pp. 473–480).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. (Publisher: Ieee)

Lee, D. H., Choi, S., Kim, H. J., & Chung, S.-Y. (2022). Unsupervised Visual Representation Learning via Mutual Information Regularized Assignment. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in Neural Information Processing Systems* (Vol. 35, pp. 29610–29623). Curran Associates, Inc.

Lee, K. S., Tran, N.-T., & Cheung, N.-M. (2021, janvier). InfoMax-GAN: Improved Adversarial Image Generation via Information Maximization and Contrastive Learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (pp. 3942–3952).

Leek, J. T., Scharpf, R. B., Bravo, H. C., Simcha, D., Langmead, B., Johnson, W. E., … Irizarry, R. A. (2010). Tackling the Widespread and Critical Impact of Batch Effects in High-Throughput Data. *Nature Reviews Genetics*, *11*(10), 733–739. (Publisher: Nature Publishing Group)

Lemhadri, I., Ruan, F., Abraham, L., & Tibshirani, R. (2021). LassoNet: A Neural Network with Feature Sparsity. *Journal of Machine Learning Research*, *22*(127), 1–29.

Li, X., Chen, Z., Poon, L. K. M., & Zhang, N. L. (2019). Learning Latent Superstructures in Variational Autoencoders for Deep Multidimensional Clustering. In *International Conference on Learning Representations.*

Li, Y., Hu, P., Liu, Z., Peng, D., Zhou, J. T., & Peng, X. (2021). Contrastive Clustering. In *2021 AAAI Conference on Artificial Intelligence (AAAI).*

Liang, D., Corneli, M., Bouveyron, C., & Latouche, P. (2022). *Clustering by Deep Latent Position Model with Graph Convolutional Network.*

Liese, F., & Vajda, I. (2006). On Divergences and Informations in Statistics and Information Theory. *IEEE Transactions on Information Theory*, *52*(10), 4394–4412. doi: 10.1109/TIT.2006.881731

Likas, A., Vlassis, N., & Verbeek, J. J. (2003). The Global K-Means Clustering Algorithm. *Pattern Recognition*, *36*(2), 451–461. doi: https://doi.org/10.1016/S0031-3203(02)00060-2

Lindman, B. R., Clavel, M.-A., Mathieu, P., Iung, B., Lancellotti, P., Otto, C. M., & Pibarot, P. (2016, mars). Calcific Aortic Stenosis. *Nature reviews Disease primers*, *2*, 16006.

Linsker, R. (1988). Self-Organization in a Perceptual Network. *Computer*, *21*(3), 105–117. doi: 10.1109/2.36

Liu, B., Xia, Y., & Yu, P. S. (2000). Clustering Through Decision Tree Construction. In *Proceedings of the Ninth International Conference on Information and Knowledge Management* (pp. 20–29).

Lloyd, S. (1982). Least Squares Quantization in PCM. *Ieee Transactions on Information Theory*, *28*(2), 129–137. (Publisher: IEEE)

Lo, K., Brinkman, R. R., & Gottardo, R. (2008). Automated Gating of Flow Cytometry Data via Robust Model-Based Clustering. *Cytometry Part a: The Journal of the International Society for Analytical Cytology*, *73*(4), 321–332. (Publisher: Wiley Online Library)

Long, Z.-Z., Xu, G., Du, J., Zhu, H., Yan, T., & Yu, Y.-F. (2021). Flexible Subspace Clustering: A Joint Feature Selection and K-Means Clustering Framework. *Big Data Research*, *23*, 100170. doi: https://doi.org/10.1016/j.bdr.2020.100170

Luo, J., Schumacher, M., Scherer, A., Sanoudou, D., Megherbi, D., Davison, T., ... Hong, H. (2010). A Comparison of Batch Effect Removal Methods for Enhancement of Prediction Performance Using MAQC-II Microarray Gene Expression Data. *The Pharmacogenomics Journal*, *10*(4), 278–291. (Publisher: Nature Publishing Group)

Luštrek, M., Gams, M., & Martinčić-Ipšić, S. (2016). What Makes Classification Trees Comprehensible? *Expert Systems with Applications*, *62*, 333–346. (Publisher: Elsevier)

Ma, C., Tschiatschek, S., Turner, R., Hernández-Lobato, J. M., & Zhang, C. (2020). VAEM: a Deep Generative Model for Heterogeneous Mixed Type Data. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (Vol. 33, pp. 11237–11247). Curran Associates, Inc.

Ma, H. (2022). Achieving Deep Clustering Through the Use of Variational Autoencoders and Similarity-Based Loss. *Mathematical Biosciences and Engineering*, *19*(10), 10344–10360.

Makarychev, K., & Shan, L. (2022). Explainable K-Means: Don't be Greedy, Plant Bigger Trees! In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing* (pp. 1629–1642). New York, NY, USA : Association for Computing Machinery. (event-place: Rome, Italy) doi: 10.1145/3519935.3520056

Malouf, J., Le Tourneau, T., Pellikka, P., Sundt, T. M., Scott, C., Schaff, H. V., & Enriquez-Sarano, M. (2012). Aortic Valve Stenosis in Community Medical Practice: Determinants of Outcome and Implications for Aortic Valve Replacement. *The Journal of Thoracic and Cardiovascular Surgery*, *144*(6), 1421–1427. (Publisher: Elsevier)

Marbac, M., Biernacki, C., & Vandewalle, V. (2017). Model-Based Clustering of Gaussian Copulas for Mixed Data. *Communications in Statistics-Theory and Methods*, *46*(23), 11635–11656. (Publisher: Taylor & Francis)

Marbac, M., & Sedki, M. (2017). Variable Selection for Model-Based Clustering using the Integrated Complete-Data Likelihood. *Statistics and Computing*, *27*(4), 1049–1063. (Publisher: Springer)

Marbac, M., Sedki, M., & Patin, T. (2020). Variable Selection for Mixed Data Clustering: Application in Human Population Genomics. *Journal of Classification*, *37*(1), 124–142. (Publisher: Springer)

Mariette, J., & Villa-Vialaneix, N. (2017, octobre). Unsupervised Multiple Kernel Learning for Heterogeneous Data Integration. *Bioinformatics*, *34*(6), 1009–1015. doi: 10.1093/bioinformatics/btx682

Mattei, P.-A., Bouveyron, C., & Latouche, P. (2016, mai). Globally Sparse Probabilistic PCA. In A. Gretton & C. C. Robert (Eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (Vol. 51, pp. 976–984). Cadiz, Spain : PMLR.

Maugis, C., Celeux, G., & Martin-Magniette, M.-L. (2009). Variable Selection for Clustering with Gaussian Mixture Models. *Biometrics*, *65*(3), 701–709. (Publisher: Wiley Online Library)

McLachlan, G. J., Bean, R. W., & Peel, D. (2002, mars). A Mixture Model-Based Approach to the Clustering of Microarray Expression Data. *Bioinformatics*, *18*(3), 413–422. doi: 10.1093/bioinformatics/18.3.413

McLachlan, G. J., & Krishnan, T. (2007). *The EM Algorithm and Extensions*. John Wiley & Sons.

Mello, D. P. M. d., Assunção, R. M., & Murai, F. (2022, juin). Top-Down Deep Clustering with Multi-Generator GANs. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 36, pp. 7770–7778). doi: 10.1609/aaai.v36i7.20745

Meng, C., Helm, D., Frejno, M., & Kuster, B. (2016). moCluster: Identifying Joint Patterns across Multiple Omics Data Sets. *Journal of Proteome Research*, *15*(3), 755–765. (Publisher: ACS Publications)

Messas, E., Ijsselmuiden, A., Trifunović-Zamaklar, D., Cholley, B., Puymirat, E., Halim, J., ... Goudot, G. (2023). Treatment of Severe Symptomatic Aortic Valve Stenosis Using Non-Invasive Ultrasound Therapy: A Cohort Study. *The Lancet*, *402*(10419), 2317–2325. doi: https://doi.org/10.1016/S0140-6736(23)01518-0

Min, E., Guo, X., Liu, Q., Zhang, G., Cui, J., & Long, J. (2018). A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. *IEEE Access*, *PP*, 1–1. doi: 10.1109/ACCESS.2018.2855437

Minka, T. (2005). *Discriminative Models, not Discriminative Training* (Rapport technique). Technical Report MSR-TR-2005-144, Microsoft Research.

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations.*

Miyato, T., Maeda, S.-i., Koyama, M., & Ishii, S. (2018). Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(8), 1979–1993.

Mollineda, R. A., & Vidal, E. (2000). *A Relative Approach to Hierarchical Clustering*. Citeseer. (Pages: 19–28 Publication Title: Pattern Recognition and Applications, Frontiers in Artificial Intelligence and Applications Volume: 56)

Molnar, C. (2020). *Interpretable Machine Learning*. Lulu.com.

Monti, S., Tamayo, P., Mesirov, J., & Golub, T. (2003, juillet). Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. *Machine Learning*, *52*(1), 91–118. doi: 10.1023/A:1023949509487

Moore, M., Chen, J., Mallow, P. J., & Rizzo, J. A. (2016). The Direct Health-Care Burden of Valvular Heart Disease: Evidence from US National Survey Data. *ClinicoEconomics and Outcomes Research*, 613–627. (Publisher: Taylor & Francis)

Morris, J. J., Schaff, H. V., Mullany, C. J., Rastogi, A., McGregor, C. G., Daly, R. C., ... Orszulak, T. A. (1993). Determinants of Survival and Recovery of Left Ventricular Function after Aortic Valve Replacement. *The Annals of Thoracic Surgery*, *56*(1), 22–30. (Publisher: Elsevier)

Moshkovitz, M., Dasgupta, S., Rashtchian, C., & Frost, N. (2020, juillet). Explainable k-Means and k-Medians Clustering. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning* (Vol. 119, pp. 7055–7065). PMLR.

Motwani, M., Dey, D., Berman, D. S., Germano, G., Achenbach, S., Al-Mallah, M. H., ... Callister, T. Q. (2017). Machine Learning for Prediction of All-Cause Mortality in Patients with Suspected Coronary Artery Disease: A 5-Year Multicentre Prospective Registry Analysis. *European Heart Journal*, *38*(7), 500–507. (Publisher: Oxford University Press)

Mrabah, N., Bouguessa, M., & Ksantini, R. (2020). Adversarial Deep Embedded Clustering: On a Better Trade-Off Between Feature Randomness and Feature Drift. *IEEE Transactions on Knowledge and Data Engineering*, *34*(4), 1603–1617. (Publisher: IEEE)

Mukherjee, S., Asnani, H., Lin, E., & Kannan, S. (2019). ClusterGAN: Latent Space Clustering in Generative Adversarial Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 4610–4617). (Issue: 01)

Ng, A., Jordan, M., & Weiss, Y. (2001). On Spectral Clustering: Analysis and an Algorithm. *Advances in Neural Information Processing Systems*, *14*.

Osnabrugge, R. L., Mylotte, D., Head, S. J., Van Mieghem, N. M., Nkomo, V. T., LeReun, C. M., ... Kappetein, A. P. (2013). Aortic Stenosis in the Elderly: Disease Prevalence and Number of Candidates for Transcatheter Aortic Valve Replacement: A Meta-Analysis and Modeling Study. *Journal of the American College of Cardiology*, *62*(11), 1002–1012. (Publisher: American College of Cardiology Foundation Washington, DC)

Otto, C. M., Nishimura, R. A., Bonow, R. O., Carabello, B. A., Erwin III, J. P., Gentile, F., ... McLeod, C. (2021). 2020 Acc/Aha Guideline for the Management of Patients with Valvular Heart Disease: Executive Summary: A Report of the American College of Cardiology/American Heart Association Joint Committee on Clinical Practice Guidelines. *Journal of the American College of Cardiology*, *77*(4), 450–500. (Publisher: American College of Cardiology Foundation Washington DC)

Owen, A. B. (2013). *Monte Carlo Theory, Methods and Examples*. Stanford.

Park, S., Han, S., Kim, S., Kim, D., Park, S., Hong, S., & Cha, M. (2021). Improving Unsupervised Image Clustering With Robust Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12278–12287).

Parsons, L., Haque, E., & Liu, H. (2004). Subspace Clustering for High Dimensional Data: A Review. *Acm sigkdd explorations newsletter*, *6*(1), 90–105. (Publisher: ACM New York, NY, USA)

Peel, D., & McLachlan, G. J. (2000). Robust Mixture Modelling using the t Distribution. *Statistics and computing*, *10*, 339–348. (Publisher: Springer)

Peng, C., Kang, Z., Yang, M., & Cheng, Q. (2016). Feature Selection Embedded Subspace Clustering. *IEEE Signal Processing Letters*, *23*(7), 1018–1022. (Publisher: IEEE)

Persy, V., & D'Haese, P. (2009). Vascular Calcification and Bone Disease: The Calcification Paradox. *Trends in molecular medicine*, *15*(9), 405–416. (Publisher: Elsevier)

Peyré, G., & Cuturi, M. (2019). Computational Optimal Transport: With Applications to Data Science. *Foundations and Trends® in Machine Learning*, *11*(5-6), 355–607. (Publisher: Now Publishers, Inc.)

Pezel, T., Sanguineti, F., Garot, P., Unterseeh, T., Champagne, S., Toupin, S., ... Ah-Sing, T. (2022). Machine-Learning Score Using Stress CMR for Death Prediction in Patients with Suspected or Known CAD. *Cardiovascular Imaging*, *15*(11), 1900–1913. (Publisher: American College of Cardiology Foundation Washington DC)

Pibarot, P., & Dumesnil, J. G. (2012). Improving Assessment of Aortic Stenosis. *Journal of the American College of Cardiology*, *60*(3), 169–180. (Publisher: American College of Cardiology Foundation Washington, DC)

Pilgrim, T., & Windecker, S. (2018). Expansion of Transcatheter Aortic Valve Implantation: new Indications and Socio-Economic Considerations. *European Heart Journal*, *39*(28), 2643–2645. (Publisher: Oxford University Press)

Poole, B., Ozair, S., Van Den Oord, A., Alemi, A., & Tucker, G. (2019, juin). On Variational Bounds of Mutual Information. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning* (Vol. 97, pp. 5171–5180). PMLR.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017, juillet). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Quarterly, C. (1985). *Almanac: 1984* (Vol. XL). Washington, D.C. : Congressional Quarterly, Incorporated.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine learning*, *1*, 81–106. (Publisher: Springer)

Quinlan, J. R. (2014). *C4. 5: Programs for Machine Learning*. Elsevier.

Raftery, A. E., & Dean, N. (2006). Variable Selection for Model-Based Clustering. *Journal of the American Statistical Association*, *101*(473), 168–178. (Publisher: Taylor & Francis)

Ramazzotti, D., Lal, A., Wang, B., Batzoglou, S., & Sidow, A. (2018). Multi-Omic Tumor Data Reveal Diversity of Molecular Mechanisms that Correlate with Survival. *Nature Communications*, *9*(1), 1–14. (Publisher: Nature Publishing Group)

Reese, S. E., Archer, K. J., Therneau, T. M., Atkinson, E. J., Vachon, C. M., De Andrade, M., ... Eckel-Passow, J. E. (2013). A New Statistic for Identifying Batch Effects in High-Throughput Genomic Data that uses Guided Principal Component Analysis. *Bioinformatics*, *29*(22), 2877–2883. (Publisher: Oxford University Press)

Ren, Y., Pu, J., Yang, Z., Xu, J., Li, G., Pu, X., ... He, L. (2022). Deep Clustering: A Comprehensive Survey. *arXiv preprint arXiv:2210.04142*.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014, juin). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In E. P. Xing & T. Jebara (Eds.), *Proceedings of the 31st International Conference on Machine Learning* (Vol. 32, pp. 1278–1286). PMLR.

Ronen, M., Finder, S. E., & Freifeld, O. (2022). DeepDPM: Deep Clustering With an Unknown Number of Clusters. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 9851–9860). doi: 10.1109/CVPR52688.2022.00963

Roth, G. A., Mensah, G. A., Johnson, C. O., Addolorato, G., Ammirati, E., Baddour, L. M., ... Benziger, C. P. (2020). Global Burden of Cardiovascular Diseases and Risk Factors, 1990–2019: Update from the Gbd 2019 Study. *Journal of the American College of Cardiology*, *76*(25), 2982–3021. (Publisher: American College of Cardiology Foundation Washington DC)

Rousseeuw, P. J. (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, *20*, 53–65. (Publisher: Elsevier)

Roux, M. (2018). A Comparative Study of Divisive and Agglomerative Hierarchical Clustering Algorithms. *Journal of Classification*, *35*, 345–366. (Publisher: Springer)

Roy, B. (1959). Transitivité Et Connexité. *CR Acad. Sci. Paris*, *249*(216-218), 182.

Rényi, A. (1961). On Measures of Entropy and Information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics* (Vol. 4, pp. 547–562). University of California Press.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved Techniques for Training GANs. *Advances in Neural Information Processing Systems*, *29*.

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics*, *6*(2), 461 – 464. (Publisher: Institute of Mathematical Statistics) doi: 10.1214/aos/1176344136

Scrucca, L. (2010). Dimension Reduction for Model-Based Clustering. *Statistics and Computing*, *20*, 471–484. (Publisher: Springer)

Scrucca, L., Fop, M., Murphy, T. B., & Raftery, A. E. (2016). mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models. *The R journal*, *8*(1), 289. (Publisher: NIH Public Access)

Scrucca, L., & Raftery, A. E. (2018, avril). clustvarsel: A Package Implementing Variable Selection for Gaussian Model-Based Clustering in R. *J. Stat. Soft.*, *84*(1), 1 – 28. doi: 10.18637/jss.v084.i01

Sellers, S. L., Turner, C. T., Sathananthan, J., Cartlidge, T. R., Sin, F., Bouchareb, R., ... Bernatchez, P. N. (2019). Transcatheter Aortic Heart Valves: Histological Analysis Providing Insight to Leaflet Thickening and Structural Valve Degeneration. *JACC: Cardiovascular Imaging*, *12*(1), 135–145. (Publisher: American College of Cardiology Foundation Washington, DC) doi: 10.1016/j.jcmg.2018.06.028

Sengupta, P. P., Shrestha, S., Kagiyama, N., Hamirani, Y., Kulkarni, H., Yanamala, N., ... Cavalcante, J. L. (2021, septembre). A Machine-Learning Framework to Identify Distinct Phenotypes of Aortic Stenosis Severity. *JACC: Cardiovascular Imaging*, *14*(9), 1707–1720. (Publisher: American College of Cardiology Foundation) doi: 10.1016/j.jcmg.2021.03.020

Sermesant, M., Delingette, H., Cochet, H., Jais, P., & Ayache, N. (2021). Applications of Artificial Intelligence in Cardiovascular Imaging. *Nature Reviews Cardiology*, *18*(8), 600–609. (Publisher: Nature Publishing Group UK London)

Sevilla, T., Revilla-Orodea, A., & San Román, J. A. (2021). Timing of Intervention in Asymptomatic Patients with Aortic Stenosis. *European Cardiology Review*, *16*. (Publisher: Radcliffe Cardiology)

Shamir, O., Sabato, S., & Tishby, N. (2010). Learning and Generalization with the Information Bottleneck. *Theoretical Computer Science*, *411*(29-30), 2696–2711. (Publisher: Elsevier)

Sharma, A., López, Y., & Tsunoda, T. (2017). Divisive Hierarchical Maximum Likelihood Clustering. *Bmc Bioinformatics*, *18*(16), 139–147. (Publisher: BioMed Central)

Shen, M., Tastet, L., Capoulade, R., Arsenault, M., Bédard, E., Clavel, M.-A., & Pibarot, P. (2020). Effect of Bicuspid Aortic Valve Phenotype on Progression of Aortic Stenosis. *European Heart Journal-Cardiovascular Imaging*, *21*(7), 727–734. (Publisher: Oxford University Press)

Shen, R., Mo, Q., Schultz, N., Seshan, V. E., Olshen, A. B., Huse, J., … Sander, C. (2012). Integrative Subtype Discovery in Glioblastoma using iCluster. *Plos One*, *7*(4), e35236. (Publisher: Public Library of Science San Francisco, USA)

Shi, J., & Malik, J. (2000). Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888–905. (Publisher: IEEE)

Shirdhonkar, S., & Jacobs, D. W. (2008). Approximate Earth Mover's Distance in Linear Time. In *2008 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8). IEEE.

Sibson, R. (1973). SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method. *The Computer Journal*, *16*(1), 30–34. (Publisher: Oxford University Press)

Simard, L., Cote, N., Dagenais, F., Mathieu, P., Couture, C., Trahan, S., … Joubert, P. (2017). Sex-Related Discordance Between Aortic Valve Calcification and Hemodynamic Severity of Aortic Stenosis: Is Valvular Fibrosis the Explanation? *Circulation research*, *120*(4), 681–691. (Publisher: Am Heart Assoc)

Slonim, N., Atwal, G. S., Tkačik, G., & Bialek, W. (2005). Information-Based Clustering. *Proceedings of the National Academy of Sciences*, *102*(51), 18297–18302. (Publisher: National Academy of Sciences)

Solorio-Fernández, S., Carrasco-Ochoa, J. A., & Martínez-Trinidad, J. F. (2020). A Review of Unsupervised Feature Selection Methods. *Artificial Intelligence Review*, *53*(2), 907–948. (Publisher: Springer)

Springenberg, J. T. (2015). Unsupervised and Semi-Supervised Learning with Categorical Generative Adversarial Networks. *arXiv preprint arXiv:1511.06390*.

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., & Lanckriet, G. R. (2009). On Integral Probability Metrics,phi-Divergences and Binary Classification. *arXiv preprint arXiv:0901.2698*.

Strehl, A., & Ghosh, J. (2002). Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of machine learning research*, *3*(Dec), 583–617.

Sturn, A., Quackenbush, J., & Trajanoski, Z. (2002). Genesis: Cluster Analysis of Microarray Data. *Bioinformatics*, *18*(1), 207–208. (Publisher: Oxford University Press)

Tadesse, M. G., Sha, N., & Vannucci, M. (2005). Bayesian Variable Selection in Clustering High-Dimensional Data. *Journal of the American Statistical Association*, *100*(470), 602–617. (Publisher: Taylor & Francis)

Tao, Y., Takagi, K., & Nakata, K. (2021). Clustering-Friendly Representation Learning via Instance Discrimination and Feature Decorrelation. In *International Conference on Learning Representations.*

Tastet, L., Capoulade, R., Clavel, M.-A., Larose, E., Shen, M., Dahou, A., … Dumesnil, J. G. (2017). Systolic Hypertension and Progression of Aortic Valve Calcification in Patients with Aortic Stenosis: Results from the PROGRESSA Study. *European Heart Journal-Cardiovascular Imaging*, *18*(1), 70–78. (Publisher: Oxford University Press)

Tastet, L., Kwiecinski, J., Pibarot, P., Capoulade, R., Everett, R. J., Newby, D. E., ... Clavel, M.-A. (2020). Sex-Related Differences in the Extent of Myocardial Fibrosis in Patients with Aortic Valve Stenosis. *JACC: Cardiovascular Imaging*, *13*(3), 699-711. doi: 10.1016/j.jcmg.2019.06.014

Tavallali, P., Tavallali, P., & Singhal, M. (2021). K-means Tree: An Optimal Clustering Tree for Unsupervised Learning. *The Journal of Supercomputing*, *77*, 5239–5266. (Publisher: Springer)

Thaden, J. J., Nkomo, V. T., & Enriquez-Sarano, M. (2014). The Global Burden of Aortic Stenosis. *Progress in Cardiovascular Diseases*, *56*(6), 565–571. (Publisher: Elsevier)

Thoenes, M., Bramlage, P., Zamorano, P., Messika-Zeitoun, D., Wendt, D., Kasel, M., ... Steeds, R. P. (2018). Patient Screening for Early Detection of Aortic Stenosis (AS)—Review of Current Practice and Future Perspectives. *Journal of Thoracic Disease*, *10*(9), 5584. (Publisher: AME Publications)

Thrun, M. C., & Stier, Q. (2021). Fundamental Clustering Algorithms Suite. *SoftwareX*, *13*, 100642. (Publisher: Elsevier)

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the Number of Clusters in a Data Set via the Gap Statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*(2), 411–423. doi: 10.1111/1467-9868.00293

Tishby, N., Pereira, F. C., & Bialek, W. (2000). The Information Bottleneck Method. *arXiv preprint physics/0004057*.

Tokodi, M., Schwertner, W. R., Kovács, A., Tősér, Z., Staub, L., Sárkány, A., ... Perge, P. (2020). Machine Learning-Based Mortality Prediction of Patients Undergoing Cardiac Resynchronization Therapy: The SEMMELWEIS-CRT Score. *European Heart Journal*, *41*(18), 1747–1756. (Publisher: Oxford University Press)

Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., & Lucic, M. (2020). On Mutual Information Maximization for Representation Learning. In *International Conference on Learning Representations.*

Van den Oord, A., Li, Y., & Vinyals, O. (2018). Representation Learning with Contrastive Predictive Coding. *arXiv e-prints*, arXiv–1807.

Vandewalle, V. (2020). Multi-Partitions Subspace Clustering. *Mathematics*, *8*(4), 597. (Publisher: Multidisciplinary Digital Publishing Institute)

Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., & Van Gool, L. (2020). Scan: Learning to Classify Images without Labels. In *European Conference on Computer Vision* (pp. 268–285). Springer.

Varshavsky, R., Gottlieb, A., Linial, M., & Horn, D. (2006, juillet). Novel Unsupervised Feature Filtering of Biological Data. *Bioinformatics*, *22*(14), e507–e513. doi: 10.1093/bioinformatics/btl214

Veronesi, C., Beccagutti, G., Corbo, M., Blini, V., & Degli Esposti, L. (2015). Cost Of Illness In Aortic Stenosis Patients. *Value in Health*, *18*(7), A386. (Publisher: Elsevier)

Ver Steeg, G., Galstyan, A., Sha, F., & DeDeo, S. (2014). Demystifying Information-Theoretic Clustering. In *International Conference on Machine Learning* (pp. 19–27). PMLR.

Villani, C. (2009). *Optimal Transport: Old and New* (Vol. 338). Springer.

Vittrant, B., Leclercq, M., Martin-Magniette, M.-L., Collins, C., Bergeron, A., Fradet, Y., & Droit, A. (2020). Identification of a Transcriptomic Prognostic Signature by Machine Learning

Using a Combination of Small Cohorts of Prostate Cancer. *Frontiers in Genetics*, *11*. doi: 10.3389/fgene.2020.550894

von Luxburg, U. (2007, décembre). A Tutorial on Spectral Clustering. *Statistics and Computing*, *17*(4), 395–416. doi: 10.1007/s11222-007-9033-z

Wagstaff, K., & Cardie, C. (2000). Clustering with Instance-level Constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 1103–1110).

Wang, Y., Zhang, D., Du, G., Du, R., Zhao, J., Jin, Y., … Lu, Q. (2020). Remdesivir in Adults with Severe COVID-19: A Randomised, Double-Blind, Placebo-Controlled, Multicentre Trial. *The Lancet*, *395*(10236), 1569–1578. (Publisher: Elsevier)

Ward Jr, J. H. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American statistical association*, *58*(301), 236–244. (Publisher: Taylor & Francis)

Warshall, S. (1962). A Theorem on Boolean Matrices. *Journal of the ACM (JACM)*, *9*(1), 11–12. (Publisher: ACM New York, NY, USA)

Williams, W. T., & Lambert, J. M. (1959). Multivariate Methods in Plant Ecology: I. Association-Analysis in Plant Communities. *The Journal of Ecology*, 83–101. (Publisher: JSTOR)

Witten, D. M., & Tibshirani, R. (2010). A Framework for Feature Selection in Clustering. *Journal of the American Statistical Association*, *105*(490), 713–726. (Publisher: Taylor & Francis)

Witten, D. M., Tibshirani, R., & Witten, M. D. (2013). *Package 'sparcl'*.

Wu, C., Khan, Z., Ioannidis, S., & Dy, J. G. (2020). Deep Kernel Learning for Clustering. In *Proceedings of the 2020 SIAM International Conference on Data Mining* (pp. 640–648). SIAM.

Xin, R., Zhong, C., Chen, Z., Takagi, T., Seltzer, M., & Rudin, C. (2022). Exploring the Whole Rashomon Set of Sparse Decision Trees. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in Neural Information Processing Systems* (Vol. 35, pp. 14071–14084). Curran Associates, Inc.

Yang, C., Ojha, B. D., Aranoff, N. D., Green, P., & Tavassolian, N. (2020, octobre). Classification of Aortic Stenosis Using Conventional Machine Learning and Deep Learning Methods Based on Multi-Dimensional Cardio-Mechanical Signals. *Scientific Reports*, *10*(1), 17521. doi: 10.1038/s41598-020-74519-6

Yang, J., Parikh, D., & Batra, D. (2016). Joint Unsupervised Learning of Deep Representations and Image Clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5147–5156).

Yang, L., Fan, W., & Bouguila, N. (2021). Deep Clustering Analysis via Dual Variational Autoencoder with Spherical Latent Embeddings. *IEEE Transactions on Neural Networks and Learning Systems*. (Publisher: IEEE)

Yang, Y., Morillo, I. G., & Hospedales, T. M. (2018). Deep Neural Decision Trees. *arXiv preprint arXiv:1806.06988*.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., & Smola, A. J. (2017). Deep Sets. In I. Guyon et al. (Eds.), *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc.

Zhang, Z., & Boykov, Y. (2023). Revisiting Discriminative Entropy Clustering and its Relation to K-means. *arXiv preprint arXiv:2301.11405*.

Zhao, Z., & Liu, H. (2007). Spectral Feature Selection for Supervised and Unsupervised Learning. In *Proceedings of the 24th international conference on Machine learning* (pp. 1151–1157).

Zhou, P., Hou, Y., & Feng, J. (2018). Deep Adversarial Subspace Clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1596–1604).

Zografos, V., Ellis, L., & Mester, R. (2013). Discriminative Subspace Clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2107–2114).

Şenbabaoğlu, Y., Michailidis, G., & Li, J. Z. (2014). Critical Limitations of Consensus Clustering in Class Discovery. *Scientific Reports*, *4*(1), 6207. (Publisher: Nature Publishing Group UK London)

List of Figures

1.1 Apical four chambers drawing of the heart . . . 2

2.1 The generative and discriminative modelling frameworks . . . 15
2.2 Example of one-dimensional Gaussian mixture . . . 16

3.1 Examples of decision boundaries for a one-dimensional Gaussian mixture . . . 35
3.2 Value of mutual information for Gaussian mixture separation . . . 36
3.3 Synthetic example motivating the one-vs-one GEMINI . . . 43
3.4 Evolution of the MSE for GEMINI estimates per batch size . . . 45
3.5 Average performance time of GEMINIs on synthetic data . . . 46
3.6 Nonparametric clustering of a Gaussian mixture with GEMINI . . . 49
3.7 Entropy of the clustering predictions for the GSTM dataset . . . 49
3.8 Clustering of the moon dataset . . . 51
3.9 ARI and non-empty clusters for MNIST clustering with GEMINI . . . 53
3.10 Evolution of the number of clusters with increasing epochs for GEMINI . . . 54
3.11 Visualisation of the Enron email dataset . . . 59
3.12 Interaction matrices of the clusters on the Enron email dataset . . . 60
3.13 Consensus distribution between clusterings on the Enron email dataset . . . 60

4.1 Overview of Sparse GEMINI . . . 63
4.2 Example of weight convergence curves for the synthetic dataset . . . 71
4.3 Feature importance map for MNIST variations dataset . . . 72
4.4 Training curves for the US Congress dataset . . . 73
4.5 Training curves for the Heart Statlog dataset . . . 74

5.1 Overview of the end-to-end clustering tree framework . . . 80
5.2 Example of dataset motivating the need for unsupervised clustering trees . . . 82
5.3 Toy example of kernel stocks . . . 88
5.4 Graphical interpretation of the α and β variables . . . 93
5.5 Explanation scores of Kauri on the Digits and Mice datasets . . . 102
5.6 Explanation scores for a rotating Gaussian mixtures . . . 102
5.7 PCA and clustering of the wine dataset . . . 104
5.8 Number of non-empty clusters for Kauri and kernel K-means . . . 105
5.9 Example of obtained Kauri structure on the US Congress dataset . . . 110

6.1 Graphical summary of the clustering pipeline for PROGRESSA . . . 118
6.2 Graphical summary of the PROGRESSA structure . . . 119
6.3 Heatmap of the clinicopathological dataset with merged clusters . . . 126
6.4 Heatmap of the fused dataset with merged clusters . . . 127
6.5 Kaplan-Meier curves for clinical endpoints per cluster . . . 132
6.6 Cox models curves for clinical endpoints per cluster . . . 132

D.1 OvA MMD-GEMINI forward pass 200

List of Tables

3.1 Definitions of the GEMINI . . . 44
3.2 Clustering performances on the GSTM dataset . . . 50
3.3 Clustering performances on the MNIST dataset . . . 52
3.4 Clustering performances on the CIFAR10 dataset . . . 55
3.5 ARI scores of the clustering models on the Enron email dataset . . . 56

4.1 Brief description of datasets involved in experiments . . . 66
4.2 Synthetic clustering performances with dynamic training (OvO) . . . 69
4.3 Synthetic clustering performances with dynamic training (OvA) . . . 70
4.4 Synthetic clustering performances with dynamic training . . . 71
4.5 Clustering performance son two OpenML datasets . . . 75
4.6 Clustering performances on the Prostate BCR dataset . . . 76

5.1 Advantages and disadvantages of the Kauri and Douglas models . . . 98
5.2 Summary of the datasets used for benchmarking clustering trees . . . 99
5.3 Clustering performances benchmark for small clustering trees . . . 100
5.4 K-means score performances benchmark for small clustering trees . . . 101
5.5 Depth performances benchmark for small clustering trees . . . 103
5.6 Clustering performances benchmark for large clustering trees . . . 104
5.7 Explanation performances benchmark for large clustering trees . . . 105
5.8 Clustering performances for kernel variations . . . 106
5.9 K-means scores for kernel variations . . . 107
5.10 Clustering performances of Douglas on the benchmark . . . 109
5.11 K-means scores of Douglas on the benchmark . . . 109

6.1 Dimensions of the considered datasets for clustering . . . 120
6.2 Accuracy of the clustering models per dataset . . . 124
6.3 Proportion of ambiguous clusters for the ensembles of clustering models . . . 124
6.4 Clinical characteristics of the clinicopathological clusters . . . 128
6.5 Metabolic characteristics of the clinicopathological clusters . . . 129
6.6 Echocardiographic characteristics of the clinicopathological clusters . . . 130

List of Algorithms

1 Split choosing algorithm for Kauri 92
2 Split computing algorithm for Kauri 95
3 Training algorithm for Kauri 96

Appendix

APPENDIX A

Proof of the mutual information convergence for the separation of two Gaussian distributions

We consider the data distribution from Section 3.2.2:

$$p(x|y=0) = \mathcal{N}(x|\mu_0, \sigma^2), p(x|y=1) = \mathcal{N}(x|\mu_1, \sigma^2), \tag{A.1}$$

where y is the cluster assignment. We take balanced clusters proportions, *i.e.* $p(y=0) = p(y=1) = \frac{1}{2}$. We defined two different clustering models: one which splits evenly the data space called p_A (good boundary) and another which splits it on a closed set p_B (misplaced boundary):

$$p_A(y=1|x) = \begin{cases} 1-\epsilon & x > \frac{\mu_1-\mu_0}{2} \\ \epsilon & \text{otherwise} \end{cases}, \tag{A.2}$$

$$p_B(y=1|x) = \begin{cases} 1-\epsilon & x \in [\mu_0, \mu_1] \\ \epsilon & \text{otherwise} \end{cases}. \tag{A.3}$$

Our goal is to show that both models p_A and p_B will converge to the same value of mutual information as ϵ converges to 0. For this demonstration, we will compute the respective Kullback-Leibler divergence D_{KL} between cluster distributions of each model, then compute the mutual information using the expectation of the divergences over the data distribution:

$$\mathcal{I}(x;y) = \mathbb{E}_{x\sim p(x)}\left[D_{\text{KL}}\left(p_\theta(y|x)\|p_\theta(y)\right)\right]. \tag{A.4}$$

A.1 Mutual information of the good boundary model

To compute the cluster proportions, we estimate with samples x from the distribution $p_{\text{data}}(x)$. Since we are aware for this demonstration of the true nature of the data distribution, we can use $p(x)$ for sampling. Consequently, we can compute the two marginals:

$$p_A(y=1) = \int_{\mathcal{X}} p(x)p_A(y=1|x)\mathrm{d}x, \tag{A.5}$$

$$= \int_{-\infty}^{\frac{\mu_1-\mu_0}{2}} p(x)\epsilon\mathrm{d}x + \int_{\frac{\mu_1-\mu_0}{2}}^{+\infty} p(x)(1-\epsilon)\mathrm{d}x, \tag{A.6}$$

$$= \epsilon\left(\int_{-\infty}^{\frac{\mu_1-\mu_0}{2}} p(x)\mathrm{d}x\right) + (1-\epsilon)\left(\int_{\frac{\mu_1-\mu_0}{2}}^{+\infty} p(x)\mathrm{d}x\right), \tag{A.7}$$

$$= \frac{1}{2}. \tag{A.8}$$

We first start by computing the Kullback-Leibler divergence for some arbitrary value of $x \in \mathbb{R}$:

$$D_{\mathrm{KL}}\left(p_A(y|x)\|p_A(y)\right) = \sum_{i=0}^{1} p_A(y=i|x)\log\frac{p_A(y=i|x)}{p_A(y=i)}. \tag{A.9}$$

We now need to detail the specific cases because the value of $p_A(y=i|x)$ is dependent on x. We start $\forall x < \frac{\mu_1-\mu_0}{2}$:

$$D_{\mathrm{KL}}\left(p_A(y|x)\|p_A(y)\right) = p_A(y=0|x)\log\frac{p_A(y=0|x)}{\frac{1}{2}} + p_A(y=1|x)\log\frac{p_A(y=1|x)}{\frac{1}{2}}, \tag{A.10}$$

$$= (1-\epsilon)\log 2(1-\epsilon) + \epsilon\log 2\epsilon. \tag{A.11}$$

The opposite case, $\forall x \geq \frac{\mu_1-\mu_0}{2}$ yields:

$$D_{\mathrm{KL}}\left(p_A(y|x)\|p_A(y)\right) = p_A(y=0|x)\log\frac{p_A(y=0|x)}{\frac{1}{2}} + p_A(y=1|x)\log\frac{p_A(y=1|x)}{\frac{1}{2}}, \tag{A.12}$$

$$= \epsilon\log 2\epsilon + (1-\epsilon)\log 2(1-\epsilon). \tag{A.13}$$

Since both cases are equal, we can write down:

$$D_{\mathrm{KL}}\left(p_A(y|x)\|p_A(y)\right) = \epsilon\log 2\epsilon + (1-\epsilon)\log 2(1-\epsilon), \forall x \in \mathbb{R}. \tag{A.14}$$

We inject the value of the Kullback-Leibler divergence from Eq. (A.14) inside an expectation performed over the data distribution $p(x)$:

$$\mathcal{I}_A\left(x;y\right) = \mathbb{E}_{x\sim p(x)}\left[D_{\mathrm{KL}}\left(p_A(y|x)\|p_A(y)\right)\right], \tag{A.15}$$

$$= \int_{\mathcal{X}} p(x)\left(\epsilon\log(2\epsilon) + (1-\epsilon)\log(2(1-\epsilon))\right)\mathrm{d}x, \tag{A.16}$$

$$= \epsilon\log(2\epsilon) + (1-\epsilon)\log(2(1-\epsilon)). \tag{A.17}$$

Since the KL divergence was independent of x, we could leave the constant outside of the integral which is equal to 1.

We can assess the coherence of Eq. (A.17) since its limit as ϵ approaches 0 is $\log 2$. In terms of bits, this is the same as saying that the information on x directly gives us information on the y of the cluster.

A.2 Mutual information of the misplaced boundary model

For the misplaced decision boundary, the marginal is different:

$$p_B(y=1) = \int_{\mathcal{X}} p(x)p_B(y=1|x)\mathrm{d}x, \tag{A.18}$$

$$= \epsilon\left(\int_{-\infty}^{\mu_0} p(x)\mathrm{d}x + \int_{\mu_1}^{+\infty} p(x)\mathrm{d}x\right) + (1-\epsilon)\int_{\mu_0}^{\mu_1} p(x)\mathrm{d}x, \tag{A.19}$$

$$= \epsilon\left(1 - \int_{\mu_0}^{\mu_1} p(x)\mathrm{d}x\right) + (1-\epsilon)\int_{\mu_0}^{\mu_1} p(x)\mathrm{d}x. \tag{A.20}$$

Here, we simply introduce a new variable named β that will be a shortcut for noting the proportion of data between μ_0 and μ_1:

$$\beta = \int_{\mu_0}^{\mu_1} p(x)\mathrm{d}x. \tag{A.21}$$

And so can we simply write the cluster proportion of decision boundary model B as:

$$p_B(y=1) = \epsilon(1-\beta) + (1-\epsilon)\beta. \tag{A.22}$$

For the Kullback-Leibler divergence, we proceed to the same detailing as before. We start with the set $x \in [\mu_0, \mu_1]$:

$$D_{\mathrm{KL}}\left(p_B(y|x)\|p_B(y)\right) = p_B(y=0|x)\log\frac{p_B(y=0|x)}{p_B(y=0)} + p_B(y=1|x)\log\frac{p_B(y=1|x)}{p_B(y=1)}, \tag{A.23}$$

$$= \epsilon\log\frac{\epsilon}{p_B(y=0)} + (1-\epsilon)\log\frac{1-\epsilon}{p_B(y=1)}. \tag{A.24}$$

When x is out of this set, the divergence becomes:

$$D_{\mathrm{KL}}\left(p_B(y|x)\|p_B(y)\right) = p_B(y=0|x)\log\frac{p_B(y=0|x)}{p_B(y=0)} + p_B(y=1|x)\log\frac{p_B(y=1|x)}{p_B(y=1)}, \tag{A.25}$$

$$= (1-\epsilon)\log\frac{1-\epsilon}{p_B(y=0)} + \epsilon\log\frac{\epsilon}{p_B(y=1)}. \tag{A.26}$$

To fuse the two results, we will write the KL divergence as such:

$$D_{\mathrm{KL}}\left(p_B(y|x)\|p_B(y)\right) = \epsilon\log\epsilon + (1-\epsilon)\log(1-\epsilon) - C(x), \forall x \in \mathbb{R}, \tag{A.27}$$

where $C(x)$ is a constant term depending on x defined by:

$$C(x) = \begin{cases} \epsilon\log p_B(y=0) + (1-\epsilon)\log p_B(y=1) & x \in [\mu_0, \mu_1] \\ \epsilon\log p_B(y=1) + (1-\epsilon)\log p_B(y=0) & x \in \mathbb{R}\setminus[\mu_0, \mu_1] \end{cases}. \tag{A.28}$$

For simplicity of later writings, we will shorten the notations by:

$$C(x) = \begin{cases} \alpha_1 & x \in [\mu_0, \mu_1] \\ \alpha_0 & x \in \mathbb{R} \setminus [\mu_0, \mu_1] \end{cases} . \tag{A.29}$$

We inject the value of the KL divergence from Eq. (A.27) inside the expectation of the mutual information:

$$\begin{aligned} \mathcal{I}_B(x;y) &= \mathbb{E}_{x\sim p(x)}\left[D_{\mathrm{KL}}\left(p_B(y|x)\|p_B(y)\right)\right], && \text{(A.30)}\\ &= \int_{\mathcal{X}} p(x)\left(\epsilon\log\epsilon + (1-\epsilon)\log(1-\epsilon) - C(x)\right)\mathrm{d}x, && \text{(A.31)}\\ &= \epsilon\log\epsilon + (1-\epsilon)\log(1-\epsilon) - \int_{\mathcal{X}} p(x)C(x)\mathrm{d}x. && \text{(A.32)} \end{aligned}$$

The first terms are constant with respect to x and the integral of $p(x)$ over $\mathcal{X}$ adds up to 1. We finally need to detail the expectation of the constant $C(x)$ from Eq. (A.28):

$$\begin{aligned} \mathbb{E}_x\left[C(x)\right] &= \int_{-\infty}^{\mu_0} C(x)p(x)\mathrm{d}x + \int_{\mu_0}^{\mu_1} C(x)p(x)\mathrm{d}x + \int_{\mu_1}^{+\infty} C(x)p(x)\mathrm{d}x, && \text{(A.33)}\\ &= \alpha_0\left(\int_{-\infty}^{\mu_0} p(x)\mathrm{d}x + \int_{\mu_1}^{+\infty} p(x)\mathrm{d}x\right) + \alpha_1\int_{\mu_0}^{\mu_1} p(x)\mathrm{d}x, && \text{(A.34)}\\ &= \alpha_0(1-\beta) + \alpha_1\beta. && \text{(A.35)} \end{aligned}$$

This can be further improved by unfolding the description of α_0 and α_1 from Eq. (A.28):

$$\begin{aligned} \alpha_0(1-\beta) + \beta\alpha_1 &= \alpha_0 + \beta(\alpha_1 - \alpha_0), && \text{(A.36)}\\ &= \epsilon\log p_B(y=1) + (1-\epsilon)\log p_B(y=0) + \beta\left[\epsilon\log p_B(y=0)\right.\\ &\quad \left.+(1-\epsilon)\log p_B(y=1) - \epsilon\log p_B(y=1) - (1-\epsilon)\log p_B(y=0)\right], && \text{(A.37)}\\ &= \left[1-\epsilon+\beta\epsilon-\beta+\beta\epsilon\right]\log p_B(y=0) + \left[\epsilon+\beta-\beta\epsilon-\beta\epsilon\right]\log p_B(y=1), && \text{(A.38)}\\ &= \log p_B(y=0) + \left[2\beta\epsilon-\beta-\epsilon\right]\log\frac{p_B(y=0)}{p_B(y=1)}. && \text{(A.39)} \end{aligned}$$

We can finally write down the mutual information for the model B:

$$\mathcal{I}_B(x;y) = \epsilon\log\epsilon + (1-\epsilon)\log(1-\epsilon) - \log p_B(y=0) - \left[2\beta\epsilon-\beta-\epsilon\right]\log\frac{p_B(y=0)}{p_B(y=1)}. \tag{A.40}$$

A.3 Differences of mutual information

Now that we have the exact value of both mutual informations, we can compute their differences:

$$\Delta_{\mathcal{I}} = \mathcal{I}_A(x;y) - \mathcal{I}_B(x;y), \tag{A.41}$$

$$= \epsilon\log(2\epsilon) + (1-\epsilon)\log(2(1-\epsilon)) - \epsilon\log\epsilon - (1-\epsilon)\log(1-\epsilon) + \log p_B(y=0) + [2\beta\epsilon - \beta - \epsilon]\log\frac{p_B(y=0)}{p_B(y=1)}, \tag{A.42}$$

$$= \epsilon\log 2 + (1-\epsilon)\log 2 + \log p_B(y=0) + [2\beta\epsilon - \beta - \epsilon]\log\frac{p_B(y=0)}{p_B(y=1)}. \tag{A.43}$$

We then deduce how the difference of mutual information evolves as the decision boundary becomes sharper, *i.e.* when ϵ approaches 0:

$$\lim_{\epsilon\to 0}\Delta_{\mathcal{I}} = \log 2 + \log p_B(y=0) - \beta\log\frac{p_B(y=0)}{p_B(y=1)}. \tag{A.44}$$

However, the cluster proportions by B $p_B(y=1)$ also take a different value as ϵ approaches 0. Recalling Eq. (A.3):

$$\lim_{\epsilon\to 0} p_B(y=1) = \beta, \lim_{\epsilon\to 0} p_B(y=0) = 1-\beta. \tag{A.45}$$

And finally can we write that:

$$\lim_{\epsilon\to 0}\Delta_{\mathcal{I}} = \log 2 + \log(1-\beta) - \beta\log\frac{1-\beta}{\beta}, \tag{A.46}$$

$$= \log 2 + (1-\beta)\log(1-\beta) + \beta\log\beta, \tag{A.47}$$

$$= \log 2 - \mathbb{H}(\beta). \tag{A.48}$$

APPENDIX B

Proofs for the GEMINI propositions

B.1 Proof of Prop. 3.3.1

Proof. For the sake of clarity, we will use the notations $\pi_k \equiv p_\theta(y=k)$ during the demonstration.

Modelisation of the conditional distribution We will consider two types of models. The first one is the *generic* clustering model, where the cluster assignment follows a categorical distribution:

$$y|\boldsymbol{x} \sim \text{Categorical}(\psi_\theta(\boldsymbol{x})), \tag{B.1}$$

where $\psi_\theta : \mathcal{X} \to \Delta_K$ is a learnable function of parameters θ and Δ_K is a K-simplex.

The second model is a Dirac distribution where the data space $\mathcal{X}$ is divided into a partition $\mathcal{X}_k, \forall k \in \{1, \cdots, K\}$:

$$p(y|\boldsymbol{x}) = \mathbb{1}[\boldsymbol{x} \in \mathcal{X}_k]\,, \tag{B.2}$$

with $\mathbb{1}$ the indicator function. This is simply a sub-case of the generic model.

For both models, we consider that clusters are not empty and that the model is not degenerate, *i.e.* $\pi_k \in]0,1[\forall k \in \{1, K\}$.

Value of OvA GEMINI and upper bounds We first unfold the OvA GEMINI for the generic model and $\alpha \in \mathbb{R} \setminus \{0,1\}$:

$$\mathcal{I}^{\text{ova}}_{D_\alpha}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[f_\alpha\left(\frac{p(\boldsymbol{x}|y)}{p(\boldsymbol{x})}\right)\right]\right], \tag{B.3}$$

$$= \sum_{k=1}^{K} \pi_k \int_{\mathcal{X}} p(\boldsymbol{x}) \left(\frac{p(\boldsymbol{x}|y=k)^\alpha p(\boldsymbol{x})^{-\alpha}}{\alpha(\alpha-1)} - \frac{p(\boldsymbol{x}|y=k)p(\boldsymbol{x})^{-1}}{\alpha-1} + \frac{1}{\alpha}\right) \mathrm{d}\boldsymbol{x}, \tag{B.4}$$

$$= \sum_{k=1}^{K} \pi_k \int_{\mathcal{X}} \left(\frac{p(\boldsymbol{x})p(y=k|\boldsymbol{x})^\alpha}{\pi_k^\alpha \alpha(\alpha-1)} - \frac{p(\boldsymbol{x}|y=k)}{\alpha-1} + \frac{p(\boldsymbol{x})}{\alpha}\right) \mathrm{d}\boldsymbol{x}. \tag{B.5}$$

After distributing the factor $p(\boldsymbol{x})$ in the integral, we can notice that the two last terms will be summed to 1, up to a factor depending on α.

$$\begin{aligned}\mathcal{I}_{D_\alpha}^{\text{ova}}(\boldsymbol{x};y) &= \sum_{k=1}^{K}\pi_k\left(\frac{1}{\alpha}-\frac{1}{\alpha-1}+\frac{1}{\pi_k^\alpha\alpha(\alpha-1)}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[p(y=k|\boldsymbol{x})^\alpha\right]\right),\\ &= \frac{-1}{\alpha(\alpha-1)}+\frac{1}{\alpha(\alpha-1)}\sum_{k=1}^{K}\pi_k^{1-\alpha}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[p(y=k|\boldsymbol{x})^\alpha\right],\\ &= (\alpha(\alpha-1))^\alpha\left[-1+\sum_{k=1}^{K}\pi_k^{1-\alpha}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[p(y=k|\boldsymbol{x})^\alpha\right]\right].\end{aligned}$$

Since we face a categorical distribution, we can affirm that $p(y|\boldsymbol{x}) \in [0,1]$. Therefore, depending on the value of α, we have either $p(y=k|\boldsymbol{x})^\alpha \in [1,\infty[$ if α is negative, and $p(y=k|\boldsymbol{x}) \in [0,1]$ for α positive. We now inspect these different cases.

Case of $\alpha \in]1,+\infty[$ The upper bound we can get on the expectation involves the inequality $p(y=k|\boldsymbol{x})^\alpha \leq p(y=k|\boldsymbol{x})$. Owing to the linearity of the expectation, we can affirm that:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[p(y=k|\boldsymbol{x})^\alpha\right] &\leq \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[p(y=k|\boldsymbol{x})\right], &\text{(B.6)}\\ &\leq \pi_k. &\text{(B.7)}\end{aligned}$$

This allows us to derive the following upper bound on the OvA GEMINI:

$$\begin{aligned}\mathcal{I}_{D_\alpha}^{\text{ova}}(\boldsymbol{x};y) &= (\alpha(\alpha-1))^{-1}\left[-1+\sum_{k=1}^{K}\pi_k^{1-\alpha}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[p(y=k|\boldsymbol{x})^\alpha\right]\right], &\text{(B.8)}\\ &\leq (\alpha(\alpha-1))^{-1}\left[-1+\sum_{k=1}^{K}\pi_k^{2-\alpha}\right], &\text{(B.9)}\\ &\leq \mathcal{B}_{]1,+\infty[}(\pi_1,\cdots,\pi_K). &\text{(B.10)}\end{aligned}$$

The upper bound $\mathcal{B}_{]1,+\infty[}(\pi_1,\cdots,\pi_K)$ is a convex function that is invariant to permutations of π_k. Interestingly, this upper bound only depends on the proportions of the clusters. Its maximimum is reached when $\pi_k = K^{-1}$. However, any solution is acceptable for the special case of $\alpha = 2$, *i.e.* the Pearson χ^2-divergence. In this case, the upper bound is a constant: $\mathcal{B}_2^+ = \frac{K-1}{2}$. In this situation, the proportions of the clusters do not matter. Only getting a Dirac model is sufficient to maximise the OvA GEMINI.

We can further conclude that the bound is tight when considering a Dirac model since $1^\alpha = 1$ and $0^\alpha = 0$, leading to:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[\mathbb{1}[\boldsymbol{x}\in\mathcal{X}_k]^\alpha\right] &= \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[\mathbb{1}[\boldsymbol{x}\in\mathcal{X}_k]\right], &\text{(B.11)}\\ &= \pi_k. &\text{(B.12)}\end{aligned}$$

And so do we conclude:

$$\mathcal{I}_{D_\alpha}^{\text{ova}}(\boldsymbol{x}; y) = (\alpha(\alpha-1))^{-1}\left[-1 + \sum_{k=1}^{K} \pi_k^{1-\alpha}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[p(y=k|\boldsymbol{x})^\alpha\right]\right], \tag{B.13}$$

$$= (\alpha(\alpha-1))^{-1}\left[-1 + \sum_{k=1}^{K} \pi_k^{2-\alpha}\right], \tag{B.14}$$

$$= \mathcal{B}_{]1,+\infty[}(\pi_1, \cdots, \pi_K). \tag{B.15}$$

Case of a $\alpha \in]0,1[$ In this case, the front factor $(\alpha(\alpha-1))^{-1}$ is negative. Thus, we are interested in minimising the second term and finding the lower bound. We can already infer:

$$p(y=k|\boldsymbol{x}) \leq p(y=k|\boldsymbol{x})^\alpha, \tag{B.16}$$

$$\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[p(y=k|\boldsymbol{x})\right] \leq \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[p(y=k|\boldsymbol{x})^\alpha\right], \tag{B.17}$$

$$\pi_k \leq \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[p(y=k|\boldsymbol{x})^\alpha\right]. \tag{B.18}$$

This lower bound is tight for a Dirac model. We can finally compute for the OvA GEMINI that:

$$\mathcal{I}_{D_\alpha}^{\text{ova}}(\boldsymbol{x}; y) = (\alpha(\alpha-1))^{-1}\left[-1 + \sum_{k=1}^{K} \pi_k^{1-\alpha}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[p(y=k|\boldsymbol{x})^\alpha\right]\right], \tag{B.19}$$

$$\leq (\alpha(\alpha-1))^{-1}\left[-1 + \sum_{k=1}^{K} \pi_k^{2-\alpha}\right], \tag{B.20}$$

$$\leq \mathcal{B}_{]0,1[}. \tag{B.21}$$

Hence, we conclude that $\mathcal{B}_{]0,1[} = \mathcal{B}_{]1,+\infty[} = \mathcal{B}_{\mathbb{R}^{+*}\setminus\{1\}}$. In both cases, using a Dirac model implies that the OvA GEMINI reaches its upper bound.

Case of a negative α The upper bound of the OvA GEMINI in this case is the infinity. Indeed, taking the example of the Dirac model is sufficient to consider regions of the data space $\mathcal{X}$ where the clustering distribution has no support. Thus, the expectation is undefined, or rather drifts towards infinity.

Specific case of $\alpha = 1$**, the KL divergence** In this case, we need to start the computations all over again using the definition $f(t) = t\log t$. Indeed, we can skip the term $-t+1$ since it does not affect the value of an f-divergence, *i.e.* for any convex function f s.t. $f(1) = 0$ and for any real constant c:

$$D_{f(t)}(p\|q) = D_{f(t)+c(t-1)}(p\|q). \tag{B.22}$$

We thus get:

$$\mathcal{I}_{D_1}^{\text{ova}}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p(y)}\left[\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[\frac{p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}\log\frac{p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}\right]\right], \tag{B.23}$$

$$= \sum_{k=1}^{K}\pi_k \int_{\mathcal{X}} p(\boldsymbol{x}|y=k)\log\left(\frac{p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}\right)\mathrm{d}\boldsymbol{x}, \tag{B.24}$$

$$= \sum_{k=1}^{K}\int_{\mathcal{X}} p(y=k|\boldsymbol{x})p(\boldsymbol{x})\log\left(\frac{p(y=k|\boldsymbol{x})}{\pi_k}\right)\mathrm{d}\boldsymbol{x}. \tag{B.25}$$

We can then separate the log term to make appear the two different entropies contributing to mutual information:

$$\mathcal{I}_{D_1}^{\text{ova}}(\boldsymbol{x}; y) = \sum_{k=1}^{K}\int_{\mathcal{X}} p(y=k|\boldsymbol{x})p(\boldsymbol{x})\log(p(y=k|\boldsymbol{x}))\mathrm{d}\boldsymbol{x} - \log\pi_k\int_{\mathcal{X}} p(y=k|\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \tag{B.26}$$

$$= \sum_{k=1}^{K}\int_{\mathcal{X}} p(y=k|\boldsymbol{x})p(\boldsymbol{x})\log(p(y=k|\boldsymbol{x}))\mathrm{d}\boldsymbol{x} - \pi_k\log\pi_k \tag{B.27}$$

We find once again an upper bound on the integral depending on $p(y|\boldsymbol{x})$. We know that the function $g : t \mapsto t\log t$ is convex and below 0 for $t \in [0, 1]$. Hence:

$$p(y=k|\boldsymbol{x})\log p(y=k|\boldsymbol{x}) \leq 0, \tag{B.28}$$

with strict equality iff $p(y=k|\boldsymbol{x}) \in \{0, 1\}$. This implies that the Dirac model maximises the left integral. We deduce the upper bound of mutual information:

$$\mathcal{I}_{D_1}^{\text{ova}}(\boldsymbol{x}; y) = \sum_{k=1}^{K}\int_{\mathcal{X}} p(y=k|\boldsymbol{x})p(\boldsymbol{x})\log(p(y=k|\boldsymbol{x}))\mathrm{d}\boldsymbol{x} - \pi_k\log\pi_k, \tag{B.29}$$

$$\leq \sum_{k=1}^{K} -\pi_k\log\pi_k, \tag{B.30}$$

$$\leq \mathcal{B}_1. \tag{B.31}$$

This shows that the cluster proportion entropy is the upper bound of mutual information. It is reached for any Dirac model.

Last subcase: the null alpha In this case, the α-divergence is defined by the function $f(t) = -\log t$. Let us derive again the OvA GEMINI:

$$\mathcal{I}_{D_0}^{\text{ova}}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p(y)}\left[\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[-\log\frac{p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}\right]\right], \tag{B.32}$$

$$= -\sum_{k=1}^{K}\pi_k\int_{\mathcal{X}} p(\boldsymbol{x})\log\left(\frac{p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}\right)\mathrm{d}\boldsymbol{x}, \tag{B.33}$$

$$= -\sum_{k=1}^{K}\pi_k\int_{\mathcal{X}} p(\boldsymbol{x})\log\left(\frac{p(y=k|\boldsymbol{x})}{\pi_k}\right)\mathrm{d}\boldsymbol{x}. \tag{B.34}$$

We expand again the logarithm and compute the integral over constant terms factorised by $p(\boldsymbol{x})$:

$$\mathcal{I}_{D_0}^{\text{ova}}(\boldsymbol{x}; y) = \sum_{k=1}^{K}\pi_k\log\pi_k - \pi_k\int_{\mathcal{X}} p(\boldsymbol{x})\log p(y=k|\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \tag{B.35}$$

Now we can see that this OvA GEMINI may converge to infinity. Indeed, for the example of the Dirac model, we evaluate the integral with terms worth $\lim_{t\to 0}\log t$. We cannot conclude on the upper bounds of this case.

Maximal upper bound We have shown so far that for $\alpha > 0$, we can derive two different upper bounds that only depend on the proportions of the clusters π_k. These upper bounds can be reached by Dirac model of type $p(y=k|\boldsymbol{x}) = \mathbb{1}[\boldsymbol{x}\in\mathcal{X}_k]$.

We can now question for the two upper bounds, $\mathcal{B}_{\mathbb{R}^{+*}\setminus\{1\}}$ and $\mathcal{B}_1$ what are the optimal cluster proportions π_k. By adding a Lagrangian constraint to enforce $\sum_{k=1}^{K}\pi_k = 1$ in each upper bound, we can show that the maximal upper bound is reached iff $\pi_k = K^{-1}\forall k$. This concludes our proof. □

B.2 Proof of Prop. 3.3.3

Proof. Let us consider the value of OvO GEMINI when the distance D is an f-divergence or an IPM.

Demonstration for f-divergences We first need to highlight that f-divergences come with a conjugate convex function g. This conjugate enables the inversion of the arguments of the f-divergence:

$$D_f(p\|q) = D_g(q\|p), \tag{B.36}$$

for any distributions p and q. We can use this trick to revert first the f-divergence between the distribution $p_\theta(y=k|\boldsymbol{x})$ and $p(\boldsymbol{x})$:

$$D_f(p_\theta(\boldsymbol{x}|y=k)\|p(\boldsymbol{x})) = D_g(p(\boldsymbol{x})\|p_\theta(\boldsymbol{x}|y=k)). \tag{B.37}$$

We then write $p(\boldsymbol{x})$ as a sum marginalising the y variable. Using the convexity of the function g, we get a weighted upper bound of this divergence:

$$D_g\left(p(\boldsymbol{x})\|p_\theta(\boldsymbol{x}|y=k)\right) = D_g\left(\sum_{k'=1}^{K} p(y=k')p(\boldsymbol{x}|y=k')\|\left(\sum_{k'=1}^{K} p(y=k')\right)p_\theta(\boldsymbol{x}|y=k)\right), \tag{B.38}$$

$$\leq \sum_{k'=1}^{K} p(y=k')D_g\left(p(\boldsymbol{x}|y=k'\|p_\theta(\boldsymbol{x}|y=k)\right), \tag{B.39}$$

$$\leq \mathbb{E}_{k'\sim p(y)}\left[D_g\left(p(\boldsymbol{x}|y=k'\|p_\theta(\boldsymbol{x}|y=k)\right)\right]. \tag{B.40}$$

To retrieve the OvO form, we can compute the expectation of this inequality over all possible combinations of $p(y)$:

$$\mathbb{E}_{y\sim p(y)}\left[D_f\left(p(\boldsymbol{x})\|p_\theta(\boldsymbol{x}|y=k)\right)\right)] \leq \mathbb{E}_{y_1,y_2\sim p(y)}\left[D_g\left(p(\boldsymbol{x}|y_1)\|p(\boldsymbol{x}|y_2)\right)\right], \tag{B.41}$$

$$\mathcal{I}_{D_f}^{\text{ova}}(\boldsymbol{x};y) \leq \mathcal{I}_{D_g}^{\text{ovo}}(\boldsymbol{x};y). \tag{B.42}$$

Then, owing to the conjugate convex functions, we can observe that for any $k, k' \in \{1,\cdots,K\}$:

$$\begin{aligned} &D_g\left(p_\theta(\boldsymbol{x}|y=k)\|p(\boldsymbol{x}|y=k')\right) + D_g\left(p(\boldsymbol{x}|y=k')\|p_\theta(\boldsymbol{x}|y=k)\right) \\ &\qquad = D_f\left(p(\boldsymbol{x}|y=k'\|p_\theta(\boldsymbol{x}|y=k)\right) + D_f\left(p_\theta(\boldsymbol{x}|y=k)\|p(\boldsymbol{x}|y=k')\right). \end{aligned} \tag{B.43}$$

Consequently, the symmetry of OvO in its double expectation implies that:

$$\mathcal{I}_{D_f}^{\text{ovo}}(\boldsymbol{x};y) = \mathcal{I}_{D_g}^{\text{ovo}}(\boldsymbol{x};y). \tag{B.44}$$

And so do we conclude that:

$$\mathcal{I}_{D_f}^{\text{ova}}(\boldsymbol{x};y) \leq \mathcal{I}_{D_f}^{\text{ovo}}(\boldsymbol{x};y). \tag{B.45}$$

Demonstration for IPMs For IPMs, we start from the OvA distance between the distribution of an arbitrary cluster i among K:

$$D_{\text{IPM}}\left(p(\boldsymbol{x}|y=i)\|p(\boldsymbol{x})\right) = \sup_{f\in\mathcal{F}} \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=i)}\left[f(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[f(\boldsymbol{x})\right]. \tag{B.46}$$

We can extend the expectation of the data distribution over all clusters using the sum rules of probabilities:

$$D_{\text{IPM}}\left(p(\boldsymbol{x}|y=i)\|p(\boldsymbol{x})\right) = \sup_{f\in\mathcal{F}} \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=i)}\left[f(\boldsymbol{x})\right] - \sum_{k=1}^{K} p(y=k)\mathbb{E}_{\boldsymbol{x}\sim p_\theta(\boldsymbol{x}|y=k)}\left[f(\boldsymbol{x})\right], \tag{B.47}$$

$$= \sup_{f\in\mathcal{F}} \sum_{k=1}^{K} p(y=k)\left(\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=i)}\left[f(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p_\theta(\boldsymbol{x}|y=k)}\left[f(\boldsymbol{x})\right]\right). \tag{B.48}$$

We used in the second line the fact that the sum of $\sum_{k=1}^{K} p(y=k) = 1$ to factorise the first expectation. Finally, we can use the property that the supremum of a sum is lower or equal than a sum of suprema:

$$D_{\text{IPM}}\left(p(\boldsymbol{x}|y=i)\|p(\boldsymbol{x})\right) \leq \sum_{k=1}^{K} \sup_{f\in\mathcal{F}} p(y=k)\left(\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=i)}\left[f(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p_\theta(\boldsymbol{x}|y=k)}\left[f(\boldsymbol{x})\right]\right), \tag{B.49}$$

$$\leq \sum_{k=1}^{K} p(y=k) D_{\text{IPM}}\left(p(\boldsymbol{x}|y=i)\|p_\theta(\boldsymbol{x}|y=k)\right). \tag{B.50}$$

This upper bound corresponds to the expectation of the IPM between a specific cluster distribution i and all other cluster distributions. Finally, by performing the expectation over all cluster of index i, we can conclude that:

$$\mathcal{I}^{\text{ova}}_{D_{\text{IPM}}}(\boldsymbol{x}; y) \leq \mathcal{I}^{\text{ovo}}_{D_{\text{IPM}}}(\boldsymbol{x}; y). \tag{B.51}$$

□

B.3 Proof of Prop. 3.3.4

Proof. We will show here that when the cluster variable y is binary, the OvA and OvO GEMINIs are equal if we use IPMs for distance between distributions. Indeed we can first unfold both equations:

$$\mathcal{I}^{\text{ova}}_{D_{\text{IPM}}}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p(y)}\left[D_{\text{IPM}}\left(p(\boldsymbol{x}|y)\|p(\boldsymbol{x})\right)\right], \tag{B.52}$$

$$= p(y=0) D_{\text{IPM}}\left(p(\boldsymbol{x}|y=0)\|p(\boldsymbol{x})\right) + p(y=1) D_{\text{IPM}}\left(p(\boldsymbol{x}|y=1)\|p(\boldsymbol{x})\right), \tag{B.53}$$

$$= p(y=0) \sup_{f\in\mathcal{F}} \{\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=0)}\left[f(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[f(\boldsymbol{x})\right]\}$$

$$+ p(y=1) \sup_{g\in\mathcal{F}} \{\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=1)}\left[g(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[g(\boldsymbol{x})\right]\}, \tag{B.54}$$

and for the OvO, we simply use the symmetric property of IPMs:

$$\mathcal{I}^{\text{ovo}}_{D_{\text{IPM}}}(\boldsymbol{x}; y) = \mathbb{E}_{y_a, y_b\sim p(y)}\left[D_{\text{IPM}}\left(p(\boldsymbol{x}|y_a)\|p(\boldsymbol{x}|y_b)\right)\right], \tag{B.55}$$

$$= p(y=0)p(y=1) D_{\text{IPM}}\left(p(\boldsymbol{x}|y=0)\|p(\boldsymbol{x}|y=1)\right)$$

$$+ p(y=1)p(y=0) D_{\text{IPM}}\left(p(\boldsymbol{x}|y=1)\|p(\boldsymbol{x}|y=0)\right), \tag{B.56}$$

$$= 2p(y=0)p(y=1) D_{\text{IPM}}\left(p(\boldsymbol{x}|y=0)\|p(\boldsymbol{x}|y=1)\right). \tag{B.57}$$

Notice that in Eq. (B.55), we skipped the terms where both the random variables y_1 and y_2 are equal, since the implied distance is necessarily 0.

Now, to show the equivalence of both equations (B.52) and (B.55), we simply need to write the sum rule of probabilities leading to the marginalisation of $\boldsymbol{x}$:

$$p(\boldsymbol{x}) = p(\boldsymbol{x}|y=0)p(y=0) + p(\boldsymbol{x}|y=1)p(y=1). \tag{B.58}$$

Thus, we can rewrite the expectations depending on the distribution $p(\boldsymbol{x})$ with other distributions for any function f:

$$\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[f(\boldsymbol{x})\right] = p(y=0)\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=0)}\left[f(\boldsymbol{x})\right] + p(y=1)\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=1)}\left[f(\boldsymbol{x})\right], \tag{B.59}$$

which we can incorporate back into Eq. (B.52) to get:

$$\begin{aligned}\mathcal{I}^{\text{ova}}_{D_{\text{IPM}}}(\boldsymbol{x}; y) = p(y=0)\sup_{f\in\mathcal{F}}\{(1-p(y=0))\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=0)}\left[f(\boldsymbol{x})\right] - p(y=1)\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=1)}\left[f(\boldsymbol{x})\right]\}\\ + p(y=1)\sup_{g\in\mathcal{F}}\{(1-p(y=1))\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=1)}\left[g(\boldsymbol{x})\right] - p(y=0)\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=0)}\left[g(\boldsymbol{x})\right]\}.\end{aligned} \tag{B.60}$$

Since we only use two clusters, we know that $p(y=1) = 1 - p(y=0)$. This helps us factorising terms inside the sup expressions:

$$\begin{aligned}\mathcal{I}^{\text{ova}}_{D_{\text{IPM}}}(\boldsymbol{x}; y) = p(y=0)\sup_{f\in\mathcal{F}}\left\{p(y=1)\left[\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=0)}\left[f(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=1)}\left[f(\boldsymbol{x})\right]\right]\right\}\\ + p(y=1)\sup_{g\in\mathcal{F}}\left\{p(y=0)\left[\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=1)}\left[g(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=0)}\left[g(\boldsymbol{x})\right]\right]\right\}.\end{aligned} \tag{B.61}$$

Eventually, the factors $p(y=0)$ and $p(y=1)$ do not depend on the functions f and g, so we can pull them out of the supremum. The remaining expressions are then symmetric and can be thus factorised:

$$\begin{aligned}\mathcal{I}^{\text{ova}}_{D_{\text{IPM}}}(\boldsymbol{x}; y) &= 2p(y=0)p(y=1)\sup_{f\in\mathcal{F}}\left\{\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=0)}\left[f(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|y=1)}\left[f(\boldsymbol{x})\right]\right\}, &\text{(B.62)}\\ &= 2p(y=0)p(y=1)D_{\text{IPM}}\left(p(\boldsymbol{x}|y=0)\|p(\boldsymbol{x}|y=1)\right), &\text{(B.63)}\\ &= \mathcal{I}^{\text{ovo}}_{D_{\text{IPM}}}(\boldsymbol{x}; y). &\text{(B.64)}\end{aligned}$$

This concludes the proof. □

B.4 Proof of Prop. 3.3.5

Proof. For any f-divergence and two distribution p and q taking value in the space $\mathcal{X}$, then disjoint support between p and q implies the maximisation of the f-divergence. Indeed, the bounds of an f-divergence are:

$$0 \leq D_f(p\|q) \leq f(0) + g(0), \tag{B.65}$$

where the upper bound can be infinity depending on f and its convex conjugate $g : t \longrightarrow tf(1/t)$. Thus, for any two different clusters $k \neq k'$:

$$0 \leq D_f\left(p(\boldsymbol{x}|y=k)\|p(\boldsymbol{x}|y=k')\right) \leq f(0)+g(0). \tag{B.66}$$

However, distributions for the same clusters have an f-divergence of 0. We can therefore sum all the terms and their respective upper bounds:

$$\sum_{k\neq k'}^{K} p(y=k)p(y=k')D_f\left(p(\boldsymbol{x}|y=k)\|p(\boldsymbol{x}|y=k')\right) \leq \sum_{k\neq k'}^{K} p(y=k)p(y=k')(f(0)+g(0)), \tag{B.67}$$

$$\mathcal{I}^{\text{ovo}}_{D_f}(\boldsymbol{x};y) \leq \sum_{k=1}^{K} p(y=k)p(y\neq k)(f(0)+g(0)), \tag{B.68}$$

Following (Caglar, 2014, theorem 5), disjoint supports between the distribution $p(\boldsymbol{x}|y=k)$ and $p(\boldsymbol{x}|y=k')$ implies the equality with the upper bound. Assume that the data space $\mathcal{X}$ is separated into K disjoint and supplementary spaces $\mathcal{X}_k$. To each subspace $\mathcal{X}_k$ corresponds a cluster distribution $p(\boldsymbol{x}|y=k)$. This disjoint supports are achieved for any model of the form:

$$p(y=k|\boldsymbol{x}) = \mathbb{1}[\boldsymbol{x}\in\mathcal{X}_k]\,, \tag{B.69}$$

which implies the disjoint distributions:

$$p(\boldsymbol{x}|y=k) \propto \mathbb{1}[\boldsymbol{x}\in\mathcal{X}_k]\,. \tag{B.70}$$

Each of these spaces control the proportion of data in the cluster k, and hence controls $p(y=k)$. Thus, the OvO GEMINI is equal to its upper bound owing to disjoint supports:

$$\mathcal{I}^{\text{ovo}}_{D_f}(\boldsymbol{x};y) = \sum_{k=1}^{K} p(y=k)p(y\neq k)(f(0)+g(0)). \tag{B.71}$$

We need to maximise the upper bound. This will be the maximum value of OvO GEMINI reachable for models with disjoint supports. Adding a Lagrangian term to respect the contraint of $\sum_{k=1}^{K} p(y=k)=1$ leads to the optimal solution $p(y=k)=\frac{1}{K}$. This concludes the proof. □

APPENDIX C

Derivation and gradients of GEMINIs

We show in this appendix how to derive all estimable forms of the GEMINI and their gradients. In the specific case of the Wasserstein-GEMINI, we consider that the proof of Proposition 3.3.2 is sufficient to provide the estimable form. For every case, we start with the estimation, then follow with the gradient.

C.1 f-divergence GEMINI

We detail here the derivation for 3 f-divergences that we previously chose: the KL divergence, the TV distance and the squared Hellinger distance. To do so, we first describe the generic scenario for any function f.

C.1.1 Generic f function

First, we recall that the definition of an f-divergence involves a convex function:

$$f : \mathbb{R}^+ \to \mathbb{R}, \tag{C.1}$$
$$x \to f(x), \tag{C.2}$$
$$\text{s.t.} \quad f(1) = 0, \tag{C.3}$$

between two distributions p and q as described:

$$D_f\left(p \| q\right) = \mathbb{E}_{\boldsymbol{z} \sim q}\left[f\left(\frac{p(\boldsymbol{z})}{q(\boldsymbol{z})}\right)\right]. \tag{C.4}$$

C.1.1.1 Derivation

We simply inject this definition in the OvA GEMINI and directly obtain both an expectation on the cluster assignment y and on the data variable $\boldsymbol{x}$. We then merge the writing of the two expectations for the sake of clarity.

$$\mathcal{I}_{D_f}^{\text{OvA}}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p_\theta(y)}\left[D_f\left(p_\theta(\boldsymbol{x}|y)\|p_{\text{data}}(\boldsymbol{x})\right)\right], \tag{C.5}$$

$$= \mathbb{E}_{y\sim p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[f\left(\frac{p_\theta(\boldsymbol{x}|y)}{p_{\text{data}}(\boldsymbol{x})}\right)\right]\right], \tag{C.6}$$

$$= \mathbb{E}_{y\sim p_\theta(y),\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right)\right]. \tag{C.7}$$

Injecting the f-divergence in the OvO GEMINI first yields:

$$\mathcal{I}_{D_f}^{\text{OvO}}(\boldsymbol{x}; y) = \mathbb{E}_{y_a,y_b\sim p_\theta(y)}\left[D_f\left(p_\theta(\boldsymbol{x}|y_a)\|p_\theta(\boldsymbol{x}|y_b)\right)\right], \tag{C.8}$$

$$= \mathbb{E}_{y_a,y_b\sim p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x}\sim p_\theta(\boldsymbol{x}|y_b)}\left[f\left(\frac{p_\theta(\boldsymbol{x}|y_a)}{p_\theta(\boldsymbol{x}|y_b)}\right)\right]\right]. \tag{C.9}$$

Now, by using Bayes theorem, we can perform the inner expectation over the data distribution. We then merge the expectations for the sake of clarity.

$$\mathcal{I}_{D_f}^{\text{OvO}}(\boldsymbol{x}; y) = \mathbb{E}_{y_a,y_b\sim p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)}f\left(\frac{p_\theta(\boldsymbol{x}|y_a)}{p_\theta(\boldsymbol{x}|y_b)}\right)\right]\right], \tag{C.10}$$

$$= \mathbb{E}_{y_a,y_b\sim p_\theta(y),\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)}f\left(\frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})p_\theta(y_a)}\right)\right]. \tag{C.11}$$

Notice that we also changed the ratio of conditional distributions inside the function by a ratio of posteriors through Bayes' theorem, weighted by the relative cluster proportions.

C.1.1.2 Gradients

We will show in this section the derivations of the gradient w.r.t. the parameters θ of the posterior distribution $p_\theta(y|\boldsymbol{x})$ for the f-divergence GEMINIs. Note that the discrete variable y for the cluster assignment has a distribution depending on the parameters θ whereas the $p_{\text{data}}(\boldsymbol{x})$ distribution does not.

Starting with OvA For any f-divergence, we can rewrite the one-vs-all GEMINI by extending the expectation over the discrete cluster assignments:

$$\mathcal{I}_{D_f}^{\text{OvA}}(\boldsymbol{x}; y) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}),y\sim p_\theta(y)}\left[f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right)\right], \tag{C.12}$$

$$= \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\sum_{y=1}^{K} p_\theta(y) f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right)\right]. \tag{C.13}$$

We can now start applying the gradient operator:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvA}}(\boldsymbol{x}; y)}{\partial \theta} = \frac{\partial \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\sum_{y=1}^{K} p_\theta(y) f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right)\right]}{\partial \theta}. \tag{C.14}$$

First, as the data distribution $p_{\text{data}}(\boldsymbol{x})$ is constant w.r.t. the parameters θ, we can perform the derivation inside the expectation:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvA}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\sum_{y=1}^{K} \frac{\partial p_\theta(y) f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right)}{\partial \theta} \right], \tag{C.15}$$

$$= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\sum_{y=1}^{K} p_\theta(y) \frac{\partial f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right)}{\partial \theta} + f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right) \frac{\partial p_\theta(y)}{\partial \theta} \right]. \tag{C.16}$$

We unfold the derivatives of the first term:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvA}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\sum_{k=1}^{K} p_\theta(y) f'\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right) \frac{\partial \frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}}{\partial \theta} + f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right) \frac{\partial p_\theta(y)}{\partial \theta} \right], \tag{C.17}$$

$$= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\sum_{k=1}^{K} p_\theta(y) f'\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right) \frac{p_\theta(y) \frac{\partial p_\theta(y|\boldsymbol{x})}{\partial \theta} - p_\theta(y|\boldsymbol{x}) \frac{\partial p_\theta(y)}{\partial \theta}}{p_\theta(y)^2} + f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right) \frac{\partial p_\theta(y)}{\partial \theta} \right], \tag{C.18}$$

$$= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\sum_{k=1}^{K} f'\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right) \frac{\partial p_\theta(y|\boldsymbol{x})}{\partial \theta} + \left\{ f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right) - \frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)} f'\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right) \right\} \frac{\partial p_\theta(y)}{\partial \theta} \right]. \tag{C.19}$$

The marginal is estimated directly as the expectation of the posterior over the data. In practice, we use an unbiased estimate for the marginal. Therefore, the gradient of the marginal w.r.t θ becomes:

$$\frac{\partial p_\theta(y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial \theta} \right], \tag{C.20}$$

which we can re-inject in the equation above:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvA}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\sum_{k=1}^{K} f'\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right) \frac{\partial p_\theta(y|\boldsymbol{x})}{\partial \theta} + \left\{ f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right) - \frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)} f'\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right) \right\} \mathbb{E}_{\boldsymbol{x}' p_{\text{data}}(\boldsymbol{x}')} \left[\frac{\partial p_\theta(y|\boldsymbol{x}')}{\partial \theta} \right] \right]. \tag{C.21}$$

We then realise that since $\boldsymbol{x}$ and $\boldsymbol{x}'$ are drawn from the same distribution, it is equivalent to rearranging the equation thanks to expectations' linearity in order to factor everything by the same posterior gradient $\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial \theta}$:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvA}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\sum_{k=1}^{K}\left\{f'\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right)\right.\right.$$
$$\left.\left.+\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x}')}\left[f\left(\frac{p_\theta(y|\boldsymbol{x}')}{p_\theta(y)}\right) - \frac{p_\theta(y|\boldsymbol{x}')}{p_\theta(y)} f'\left(\frac{p_\theta(y|\boldsymbol{x}')}{p_\theta(y)}\right)\right]\right\}\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial \theta}\right], \tag{C.22}$$

$$= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}), y\sim p_\theta(y)}\left[\frac{1}{p_\theta(y)}\left\{f'\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right)\right.\right.$$
$$\left.\left.+\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x}')}\left[f\left(\frac{p_\theta(y|\boldsymbol{x}')}{p_\theta(y)}\right) - \frac{p_\theta(y|\boldsymbol{x}')}{p_\theta(y)} f'\left(\frac{p_\theta(y|\boldsymbol{x}')}{p_\theta(y)}\right)\right]\right\}\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial \theta}\right]. \tag{C.23}$$

Therefore the gradient of the OvA f-divergence-GEMINI consists in two terms. The first one depends on both the value of the data $\boldsymbol{x}$ and the cluster assignment y, whereas the second only depends on the cluster assignment y since it consists in an expectation over the data.

Concluding with OvO Once again, the data distribution does not undergo the gradient operator for it does not depend on the parameters θ. Therefore, we pass the gradient operator inside the data expectation from equation C.11:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvO}}(\boldsymbol{x}; y)}{\partial \theta} = \frac{\partial \mathbb{E}_{y_a, y_b\sim p_\theta(y), \boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} f\left(\frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})p_\theta(y_a)}\right)\right]}{\partial \theta}, \tag{C.24}$$

$$= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\sum_{\substack{y_a=1\\y_b=1}}^{K,K} \frac{\partial p_\theta(y_a)p_\theta(y_b|\boldsymbol{x}) f\left(\frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})p_\theta(y_a)}\right)}{\partial \theta}\right]. \tag{C.25}$$

For the sake of brevity, we will note the inner fraction:

$$\gamma = \frac{p_\theta(\boldsymbol{x}|y_a)}{p_\theta(\boldsymbol{x}|y_b)} = \frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})p_\theta(y_a)}, \tag{C.26}$$

which derivative w.r.t. θ is:

$$\frac{\partial \gamma}{\partial \theta} = \frac{\gamma}{p_\theta(y_b)}\frac{\partial p_\theta(y_b)}{\partial \theta} + \frac{\gamma}{p_\theta(y_a|\boldsymbol{x})}\frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial \theta} - \frac{\gamma}{p_\theta(y_b|\boldsymbol{x})}\frac{\partial p_\theta(y_b|\boldsymbol{x})}{\partial \theta} - \frac{\gamma}{p_\theta(y_a)}\frac{\partial p_\theta(y_a)}{\partial \theta}. \tag{C.27}$$

We can now inject these elements inside the main derivative:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvO}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\sum_{\substack{y_a=1\\y_b=1}}^{K,K} \frac{\partial p_\theta(y_a) p_\theta(y_b|\boldsymbol{x}) f(\gamma)}{\partial \theta}\right], \tag{C.28}$$

$$= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\sum_{\substack{y_a=1\\y_b=1}}^{K,K} p_\theta(y_b|\boldsymbol{x}) f(\gamma) \frac{\partial p_\theta(y_a)}{\partial \theta} + p_\theta(y_a) f(\gamma) \frac{\partial p_\theta(y_b|\boldsymbol{x})}{\partial \theta}\right.$$

$$\left. + p_\theta(y_a) p_\theta(y_b|\boldsymbol{x}) f'(\gamma) \frac{\partial \gamma}{\partial \theta}\right]. \tag{C.29}$$

We detail the derivation of the last term inside the expectation by first noticing that the front factor can be rewritten with the inverse of γ, then develop and simplify the fractions:

$$p_\theta(y_a) p_\theta(y_b|\boldsymbol{x}) f'(\gamma) \frac{\partial \gamma}{\partial \theta} = \frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b)}{\gamma} f'(\gamma) \frac{\partial \gamma}{\partial \gamma}, \tag{C.30}$$

$$= \frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b)}{\gamma} f'(\gamma) \left(\frac{\gamma}{p_\theta(y_b)} \frac{\partial p_\theta(y_b)}{\partial \theta} + \frac{\gamma}{p_\theta(y_a|\boldsymbol{x})} \frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial \theta}\right.$$

$$\left. - \frac{\gamma}{p_\theta(y_b|\boldsymbol{x})} \frac{\partial p_\theta(y_b|\boldsymbol{x})}{\partial \theta} - \frac{\gamma}{p_\theta(y_a)} \frac{\partial p_\theta(y_a)}{\partial \theta}\right), \tag{C.31}$$

$$= f'(\gamma) \left(p_\theta(y_a|\boldsymbol{x}) \frac{\partial p_\theta(y_b)}{\partial \theta} + p_\theta(y_b) \frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial \theta} - p_\theta(y_a) \gamma \frac{\partial p_\theta(y_b|\boldsymbol{x})}{\partial \theta}\right.$$

$$\left. - p_\theta(y_b|\boldsymbol{x}) \gamma \frac{\partial p_\theta(y_a)}{\partial \theta}\right). \tag{C.32}$$

We can now rewrite all inner terms of the expectation factorised with their respective gradient term:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvO}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\sum_{\substack{y_a=1\\y_b=1}}^{K,K} \left(f(\gamma) - \gamma f'(\gamma)\right) p_\theta(y_b|\boldsymbol{x}) \frac{\partial p_\theta(y_a)}{\partial \theta}\right.$$

$$\left. + p_\theta(y_a|\boldsymbol{x}) f'(\gamma) \frac{\partial p_\theta(y_b)}{\partial \theta} + p_\theta(y_b) f'(\gamma) \frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial \theta} + \left(f(\gamma) - \gamma f'(\gamma)\right) p_\theta(y_a) \frac{\partial p_\theta(y_b|\boldsymbol{x})}{\partial \theta}\right]. \tag{C.33}$$

Since the two random variables y_a and y_b are independent, all terms have a symmetric version which simply consists in swapping the position of the index a or b. Note that the swapped version of γ is $\frac{1}{\gamma}$. We can therefore write a new version of the gradient that only depends on the gradient of the marginal and posterior of one single cluster assignment y_a:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvO}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\sum_{y_a=1}^{K} \left\{ \sum_{y_b=1}^{K} p_\theta(y_b|\boldsymbol{x}) \left(f(\gamma) - f'(\gamma) + f'(\frac{1}{\gamma}) \right) \frac{\partial p_\theta(y_a)}{\partial \theta} + \sum_{y_b=1}^{K} p_\theta(y_b) \left(f'(\gamma) + f(\frac{1}{\gamma}) - \frac{1}{\gamma} f'(\frac{1}{\gamma}) \right) \frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial \theta} \right\} \right]. \quad \text{(C.34)}$$

We can now introduce the function h defined as:

$$h(t) = f(t) - t f'(t) + f'(\frac{1}{t}), \quad \text{(C.35)}$$

which we can recognize in our gradient and thus simplify the notation:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvO}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\sum_{y_a=1}^{K} \sum_{y_b=1}^{K} p_\theta(y_b|\boldsymbol{x}) h(\gamma) \frac{\partial p_\theta(y_a)}{\partial \theta} + \sum_{y_b=1}^{K} p_\theta(y_b) h(\frac{1}{\gamma}) \frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial \theta} \right]. \quad \text{(C.36)}$$

We finish this step by replacing sums in the equation by expectations:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvO}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x}), y_a \sim p_\theta(y)} \left[\frac{1}{p_\theta(y_a)} \mathbb{E}_{y_b \sim p_\theta(y)} \left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} h(\gamma) \right] \frac{\partial p_\theta(y_a)}{\partial \theta} + \frac{1}{p_\theta(y_a)} \mathbb{E}_{y_b \sim p_\theta(y)} \left[h(\frac{1}{\gamma}) \right] \frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial \theta} \right]. \quad \text{(C.37)}$$

In order to finish this demonstration, we perform the same trick as in the OvA scenario. In practice, the cluster proportion are estimated via the posterior distribution, and so must we unfold the expression $\frac{\partial p_\theta(y_a)}{\partial \theta}$:

$$\frac{\partial p_\theta(y_a)}{\partial \theta} = \frac{\partial \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[p_\theta(y_a|\boldsymbol{x}) \right]}{\partial \theta} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial \theta} \right], \quad \text{(C.38)}$$

leading to:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvO}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x}), y_a \sim p_\theta(y)} \left[\frac{1}{p_\theta(y_a)} \mathbb{E}_{y_b \sim p_\theta(y)} \left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} h(\gamma) \right] \times \mathbb{E}_{\boldsymbol{x}' \sim p_{\text{data}}(\boldsymbol{x}')} \left[\frac{\partial p_\theta(y_a|\boldsymbol{x}')}{\partial \theta} \right] + \frac{1}{p_\theta(y_a)} \mathbb{E}_{y_b \sim p_\theta(y)} \left[h(\frac{1}{\gamma}) \right] \frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial \theta} \right]. \quad \text{(C.39)}$$

Then, we can reorder the elements thanks to the linearity of the expectation:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvO}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x}), y_a \sim p_\theta(y)} \left[\frac{1}{p_\theta(y_a)} \mathbb{E}_{y_b \sim p_\theta(y)} \left[h(\frac{1}{\gamma}) - \mathbb{E}_{\boldsymbol{x}' \sim p_{\text{data}}(\boldsymbol{x}')} \left[\frac{p_\theta(y_b|\boldsymbol{x}')}{y_b \sim p_\theta(y)} h(\gamma) \right] \right] \frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial \theta} \right]. \quad \text{(C.40)}$$

For the final equation, we can expand again the definition of γ:

$$\frac{\partial \mathcal{I}_{D_f}^{\text{OvO}}(\boldsymbol{x}; y)}{\partial \theta} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}), y_a \sim p_\theta(y)} \left[\frac{1}{p_\theta(y_a)} \mathbb{E}_{y_b \sim p_\theta(y)} \left[h\left(\frac{p_\theta(\boldsymbol{x}|y_b)}{p_\theta(\boldsymbol{x}|y_a)} \right) - \mathbb{E}_{\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x}')} \left[\frac{p_\theta(y_b|\boldsymbol{x}')}{p_\theta(y_b)} h\left(\frac{p_\theta(\boldsymbol{x}'|y_a)}{p_\theta(\boldsymbol{x}'|y_b)} \right) \right] \right] \frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial \theta} \right]. \quad \text{(C.41)}$$

Thus, we highlighted for both the OvA and OvO GEMINIs based on f-divergences that gradients consists in two terms within an expectation. The first one varies both with the values of the cluster assignment y and the data sample $\boldsymbol{x}$, whereas the second one only varies with y. For the remainder of this section, we will not give the explicit formula for the gradient of each specific GEMINI and we will only focus on their estimate.

C.1.2 Kullback-Leibler divergence

The function for Kullback-Leibler is $f(t) = t \log t$ and its derivative $f'(t) = \log t + 1$. We do not need to write the OvA equation: it is straightforwardly the usual MI. For the OvO, we inject in Eq. (C.11) the function definition by replacing:

$$t = \frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x}) p_\theta(y_a)}, \quad \text{(C.42)}$$

in order to get:

$$\mathcal{I}_{D_{\text{KL}}}^{\text{OvO}}(\boldsymbol{x}; y) = \mathbb{E}_{y_a, y_b \sim p_\theta(y), \boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} \times \frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x}) p_\theta(y_a)} \log \frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x}) p_\theta(y_a)} \right]. \quad \text{(C.43)}$$

We can first simplify the factors outside of the logs:

$$\mathcal{I}_{D_{\text{KL}}}^{\text{OvO}}(\boldsymbol{x}; y) = \mathbb{E}_{y_a, y_b \sim p_\theta(y), \boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \log \frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x}) p_\theta(y_a)} \right]. \quad \text{(C.44)}$$

If we use the properties of the log, we can separate the inner term in two sub-expressions:

$$\mathcal{I}_{D_{\text{KL}}}^{\text{OvO}}(\boldsymbol{x}; y) = \mathbb{E}_{y_a, y_b \sim p_\theta(y), \boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \log \frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} + \frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \log \frac{p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})} \right]. \quad \text{(C.45)}$$

Hence, we can use the linearity of the expectation to separate the two terms above. The first term is constant w.r.t. y_b, so we can remove this variable from the expectation among the subscripts:

$$\mathcal{I}_{D_{\text{KL}}}^{\text{OvO}}(\boldsymbol{x}; y) = \mathbb{E}_{y_a \sim p_\theta(y), \boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \log \frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \right] + \mathbb{E}_{y_a, y_b \sim p_\theta(y), \boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \log \frac{p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})} \right]. \quad \text{(C.46)}$$

Since the variables y_a and y_b are independent, we can use the fact that:

$$\mathbb{E}_{y_a \sim p_\theta(y)} \left[\frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \right] = \int p_\theta(y_a) \frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} dy_a = 1, \tag{C.47}$$

inside the second term to reveal the final form of the equation:

$$\mathcal{I}^{\text{OvO}}_{D_{\text{KL}}}(\boldsymbol{x}; y) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x}), y \sim p_\theta(y)} \left[\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)} \log \frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)} \right] + \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x}), y \sim p_\theta(y)} \left[\log \frac{p_\theta(y)}{p_\theta(y|\boldsymbol{x})} \right]. \tag{C.48}$$

Notice that since both terms did not compare one cluster assignment y_a against another y_b, we can switch to the same common variable y. Both terms are in fact KL divergences depending on the cluster assignment y. The first is the reverse of the second. This sum of KL divergences is sometimes called the *symmetric* KL, and so can we write in two ways the OvO KL-GEMINI:

$$\begin{aligned} \mathcal{I}^{\text{OvO}}_{D_{\text{KL}}}(\boldsymbol{x}; y) &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[D_{\text{KL}} \left(p_\theta(y|\boldsymbol{x}) \| p_\theta(y) \right) \right] + \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[D_{\text{KL}} \left(p_\theta(y) \| p_\theta(y|\boldsymbol{x}) \right) \right], & \text{(C.49)} \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[D_{\text{KL-sym}} \left(p_\theta(y|\boldsymbol{x}) \| p_\theta(y) \right) \right]. & \text{(C.50)} \end{aligned}$$

We can also think of this equation as the usual MI with an additional term based on the reversed KL divergence.

C.1.3 Total Variation distance

For the total variation, the function is $f(t) = \frac{1}{2}|t-1|$. Thus, the OvA GEMINI is:

$$\mathcal{I}^{\text{OvA}}_{D_{\text{TV}}}(\boldsymbol{x}; y) = \frac{1}{2} \mathbb{E}_{y \sim p_\theta(y), \boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\left| \frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)} - 1 \right| \right]. \tag{C.51}$$

And the OvO is:

$$\begin{aligned} \mathcal{I}^{\text{OvO}}_{D_{\text{TV}}}(\boldsymbol{x}; y) &= \frac{1}{2} \mathbb{E}_{y_a, y_b \sim p_\theta(y), \boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} \left| \frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x}) p_\theta(y_a)} - 1 \right| \right], & \text{(C.52)} \\ &= \frac{1}{2} \mathbb{E}_{y_a, y_b \sim p_\theta(y), \boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\left| \frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} - \frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} \right| \right]. & \text{(C.53)} \end{aligned}$$

We did not find any further simplification of these equations.

C.1.4 Squared Hellinger distance

Finally, the squared Hellinger distance is based on $f(t) = 2(1 - \sqrt{t})$. Hence the OvA unfolds as:

$$\begin{aligned} \mathcal{I}^{\text{OvA}}_{D_{H^2}}(\boldsymbol{x}; y) &= \mathbb{E}_{y \sim p_\theta(y), \boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[2 \left(1 - \sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}} \right) \right], & \text{(C.54)} \\ &= 2 - 2 \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x}), y \sim p_\theta(y)} \left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}} \right]. & \text{(C.55)} \end{aligned}$$

The idea of the squared OvA Hellinger-GEMINI is therefore to minimise the expected square root of the relative certainty between the posterior and cluster proportion.

For the OvO setting, the definition yields:

$$\mathcal{I}^{\text{OvO}}_{D_{H^2}}(\boldsymbol{x}; y) = \mathbb{E}_{y_a, y_b \sim p_\theta(y), \boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} \times 2 \times \left(1 - \sqrt{\frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x}) p_\theta(y_a)}} \right) \right], \tag{C.56}$$

which we can already simplify by putting the constant 2 outside of the expectation, and by inserting all factors inside the square root before simplifying and separating the expectation:

$$\mathcal{I}^{\text{OvO}}_{D_{H^2}}(\boldsymbol{x}; y) = 2\mathbb{E}_{y_a, y_b \sim p_\theta(y), \boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} - \frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} \sqrt{\frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b)}{p_\theta(y_a) p_\theta(y_b|\boldsymbol{x})}} \right], \tag{C.57}$$

$$= 2\mathbb{E}_{y_a, y_b \sim p_\theta(y), \boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} \right] - 2\mathbb{E}_{y_a, y_b \sim p_\theta(y), \boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\sqrt{\frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_a) p_\theta(y_b)}} \right]. \tag{C.58}$$

We can replace the first term by the constant 1, as shown for the OvO KL derivation. Since we can split the square root into the product of two square roots, we can apply twice the expectation over y_a and y_b because these variables are independent:

$$\mathcal{I}^{\text{OvO}}_{D_{H^2}}(\boldsymbol{x}; y) = 2 - 2\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\mathbb{E}_{y \sim p_\theta(y)} \left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}} \right]^2 \right]. \tag{C.59}$$

To avoid computing this squared expectation, we use the equation of the variance $\mathbb{V}$ to replace it. Thus:

$$\mathcal{I}^{\text{OvO}}_{D_{H^2}}(\boldsymbol{x}; y) = 2 - 2\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\mathbb{E}_{y \sim p_\theta(y)} \left[\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)} \right] - \mathbb{V}_{y \sim p_\theta(y)} \left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}} \right] \right], \tag{C.60}$$

$$= 2 - 2\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\mathbb{E}_{y \sim p_\theta(y)} \left[\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)} \right] \right] + 2\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\mathbb{V}_{y \sim p_\theta(y)} \left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}} \right] \right]. \tag{C.61}$$

Then, for the same reason as before, the second term is worth 1, which cancels the first constant. We therefore end up with:

$$\mathcal{I}^{\text{OvO}}_{D_{H^2}}(\boldsymbol{x}; y) = 2\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\mathbb{V}_{y \sim p_\theta(y)} \left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}} \right] \right]. \tag{C.62}$$

Similar to the OvO KL case, the OvO squared Hellinger converges to an OvA setting, *i.e.* we only need information about the cluster distribution itself without comparing it to another. Furthermore, the idea of maximising the variance of the cluster assignments is straightforward for clustering.

C.2 Maximum Mean Discrepancy

When using an IPM with a family of functions that project an input of $\mathcal{X}$ to the unit ball of an RKHS $\mathcal{H}$, the IPM becomes the MMD distance.

$$D_{\text{MMD}}\left(p\|q\right) = \sup_{f:\|f\|_{\mathcal{H}}\leq 1} \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[f(\boldsymbol{z})\right] - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}\left[f(\boldsymbol{z})\right], \tag{C.63}$$

$$= \|\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\varphi(\boldsymbol{z})\right] - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}\left[\varphi(\boldsymbol{z})\right]\|_{\mathcal{H}}, \tag{C.64}$$

$$\tag{C.65}$$

where φ is a embedding function of the RKHS.

By using a kernel function $\kappa(\boldsymbol{z}_a, \boldsymbol{z}_b) =< \varphi(\boldsymbol{z}_a), \varphi(\boldsymbol{z}_b) >$, we can express the square of this distance thanks to inner product space properties (Gretton et al., 2012):

$$D_{\text{MMD}^2}\left(p\|q\right) = \mathbb{E}_{\boldsymbol{z}_a,\boldsymbol{z}_b\sim p(\boldsymbol{z})}\left[\kappa(\boldsymbol{z}_a,\boldsymbol{z}_b)\right] + \mathbb{E}_{\boldsymbol{z}_a,\boldsymbol{z}_b\sim q(\boldsymbol{z})}\left[\kappa(\boldsymbol{z}_a,\boldsymbol{z}_b)\right] - 2\mathbb{E}_{\boldsymbol{z}_a\sim p(\boldsymbol{z}),\boldsymbol{z}_b\sim q(\boldsymbol{z})}\left[\kappa(\boldsymbol{z}_a,\boldsymbol{z}_b)\right]. \tag{C.66}$$

Throughout this section, we will use the following shortcut notation:

$$D_{\text{MMD}^2}\left(p\|q\right) = (\alpha + \beta - 2\gamma)^2, \tag{C.67}$$

where we purposefully omit the dependence on $\boldsymbol{x}$, y and θ in the terms α, β and γ for the sake of brevity.

Now, we can derive each term of this equation using our distributions $p \equiv p_\theta(\boldsymbol{x}|y)$ and $q \equiv p_{\text{data}}(\boldsymbol{x})$ for the OvA case, and $p \equiv p_\theta(\boldsymbol{x}|y_a), q \equiv p_\theta(\boldsymbol{x}|y_b)$ for the OvO case. In both scenarios, we aim at finding an expectation over the data variable $\boldsymbol{x}$ using only the respectively known and estimable terms $p_\theta(y|\boldsymbol{x})$ and $p_\theta(y)$.

C.2.1 OvA scenario

C.2.1.1 Estimation

For the first term, we use Bayes' theorem twice to get an expectation over two variables $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ drawn from the data distribution.

$$\alpha = \mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_\theta(\boldsymbol{x}|y)}\left[\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right] = \mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y|\boldsymbol{x}_a)p_\theta(y|\boldsymbol{x}_b)}{p_\theta(y)^2}\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right]. \tag{C.68}$$

For the second term, we do not need to perform anything particular as we directly get an expectation over the data variables $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$. The last term only needs Bayes theorem once, for the distribution q is directly replaced by the data distribution $p_{\text{data}}(\boldsymbol{x})$:

$$\gamma = \mathbb{E}_{\boldsymbol{x}_a\sim p_\theta(\boldsymbol{x}|y),\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right] = \mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y|\boldsymbol{x}_a)}{p_\theta(y)}\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right]. \tag{C.69}$$

Note that for the last term, we could replace $p_\theta(y|\boldsymbol{x}_a)$ by $p_\theta(y|\boldsymbol{x}_b)$; that would not affect the result since $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are independently drawn from $p_{\text{data}}(\boldsymbol{x})$. We thus replace all terms, and do not forget to put a square root on the entire sum since we have computed so far the squared MMD:

$$\mathcal{I}^{\text{OvA}}_{D_{\text{MMD}}}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p_\theta(y)}\left[D_{\text{MMD}}\left(p_\theta(\boldsymbol{x}|y)\|p_{\text{data}}(\boldsymbol{x})\right)\right], \tag{C.70}$$

$$= \mathbb{E}_{y\sim p_\theta(y)}\left[\left(\mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y|\boldsymbol{x}_a)p_\theta(y|\boldsymbol{x}_b)}{p_\theta(y)^2}\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right]\right.\right.$$
$$\left.\left.+\mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right] - 2\mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y|\boldsymbol{x}_a)}{p_\theta(y)}\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right]\right)^{\frac{1}{2}}\right]. \tag{C.71}$$

Since all variables $\boldsymbol{x}_a$, $\boldsymbol{x}'_a$, $\boldsymbol{x}_b$ and $\boldsymbol{x}'_b$ are independently drawn from the same distribution $p_{\text{data}}(\boldsymbol{x})$, we can replace all of them by the variables $\boldsymbol{x}$ and $\boldsymbol{x}'$. We then use the linearity of the expectation and factorise by the kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}')$:

$$\mathcal{I}^{\text{OvA}}_{D_{\text{MMD}}}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')\left(\frac{p_\theta(y|\boldsymbol{x})p_\theta(y|\boldsymbol{x}')}{p_\theta(y)^2} + 1 - 2\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right)\right]^{\frac{1}{2}}\right]. \tag{C.72}$$

C.2.1.2 Gradient

To compute the gradient, we will first unfold the expectation over the cluster proportions to be able to differentiate:

$$\mathcal{I}^{\text{OvA}}_{D_{\text{MMD}}}(\boldsymbol{x}; y) = \sum_{y=1}^{K}\sqrt{p_\theta(y)^2 D_{\text{MMD}^2}\left(p_\theta(\boldsymbol{x}|y)\|p_{\text{data}}(\boldsymbol{x})\right)}, \tag{C.73}$$

$$= \sum_{y=1}^{K}\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')\left(p_\theta(y|\boldsymbol{x})p_\theta(y|\boldsymbol{x}') + p_\theta(y)^2 - 2p_\theta(y|\boldsymbol{x})p_\theta(y)\right)\right]^{\frac{1}{2}}. \tag{C.74}$$

There, we can first differentiate with respect to the distance, and then backpropagate for each term within the expectation because the data distribution does not depend on θ:

$$\frac{\partial\mathcal{I}^{\text{OvA}}_{D_{\text{MMD}}}(\boldsymbol{x}; y)}{\partial\theta} = \sum_{y=1}^{K}\frac{1}{2p_\theta(y)D_{\text{MMD}}\left(p_\theta(\boldsymbol{x}|y)\|p_{\text{data}}(\boldsymbol{x})\right)}\times$$
$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')\left(\frac{\partial p_\theta(y|\boldsymbol{x})p_\theta(y|\boldsymbol{x}')}{\partial\theta} + \frac{\partial p_\theta(y)^2}{\partial\theta} - 2\frac{\partial p_\theta(y|\boldsymbol{x})p_\theta(y)}{\partial\theta}\right)\right]. \tag{C.75}$$

Each term corresponds then to the respective gradients of α, β and γ. For the first term, we will use the fact that $\boldsymbol{x}$ and $\boldsymbol{x}'$ are sampled independently and the symmetry of the kernel to swap the expectations definitions and get a factor 2:

$$\frac{\partial \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\alpha\right]}{\partial\theta} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')\frac{\partial p_\theta(y|\boldsymbol{x})p_\theta(y|\boldsymbol{x}')}{\partial\theta}\right], \tag{C.76}$$

$$= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')\left(p_\theta(y|\boldsymbol{x})\frac{\partial p_\theta(y|\boldsymbol{x}')}{\partial\theta} + p_\theta(y|\boldsymbol{x}')\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial\theta}\right)\right], \tag{C.77}$$

$$= 2\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')p_\theta(y|\boldsymbol{x}')\right]\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial\theta}\right]. \tag{C.78}$$

The second term β is easier since it depends only on the cluster proportions which is constant throughout the double expectations. Thus:

$$\frac{\partial \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\beta\right]}{\partial\theta} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')\frac{\partial p_\theta(y)^2}{\partial\theta}\right], \tag{C.79}$$

$$= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')\right] \times 2p_\theta(y)\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial\theta}\right], \tag{C.80}$$

$$= 2\bar{\kappa}p_\theta(y)\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial\theta}\right], \tag{C.81}$$

where $\bar{\kappa}$ is the mean kernel of the data. We then finish the last term:

$$\frac{\partial \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\gamma\right]}{\partial\theta} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')\frac{\partial p_\theta(y|\boldsymbol{x})p_\theta(y)}{\partial\theta}\right], \tag{C.82}$$

$$= p_\theta(y)\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial\theta}\right] + \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')p_\theta(y|\boldsymbol{x}) \times \mathbb{E}_{\boldsymbol{x}''\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{\partial p_\theta(y|\boldsymbol{x}'')}{\partial\theta}\right]\right], \tag{C.83}$$

$$= p_\theta(y)\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\cdot)\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial\theta}\right] + \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x}',\cdot)p_\theta(y|\boldsymbol{x}')\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial\theta}\right]. \tag{C.84}$$

Finally, we can merge these three terms to obtain the complete OvA-MMD gradient:

$$\frac{\partial \mathcal{I}^{\text{OvA}}_{D_{\text{MMD}}}(\boldsymbol{x};y)}{\partial\theta} = \sum_{y=1}^{K}\frac{1}{2p_\theta(y)D_{\text{MMD}}\left(p_\theta(\boldsymbol{x}|y)||p_{\text{data}}(\boldsymbol{x})\right)}\times \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\left\{2\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')p_\theta(y|\boldsymbol{x}')\right] + 2\bar{\kappa}p_\theta(y) - 2p_\theta(y)\kappa(\boldsymbol{x},\cdot) - 2\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x}',\cdot)p_\theta(y|\boldsymbol{x}')\right]\right\}\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial\theta}\right]. \tag{C.85}$$

To conclude, we simplify the factor 2 and factorise a bit the equation to obtain:

$$\frac{\partial \mathcal{I}^{\text{OvA}}_{D_{\text{MMD}}}(\boldsymbol{x}; y)}{\partial \theta} = \sum_{y=1}^{K} \frac{1}{p_\theta(y) D_{\text{MMD}}\left(p_\theta(\boldsymbol{x}|y) \| p_{\text{data}}(\boldsymbol{x})\right)} \times$$
$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\left\{\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[[\kappa(\boldsymbol{x},\boldsymbol{x}') - \kappa(\boldsymbol{x}',\cdot)]p_\theta(y|\boldsymbol{x}')\right] + [\bar{\kappa} - \kappa(\boldsymbol{x},\cdot)]p_\theta(y)\right\}\frac{\partial p_\theta(y|\boldsymbol{x})}{\partial \theta}\right] \quad \text{(C.86)}$$

C.2.2 OvO scenario

C.2.2.1 Estimation

The two first terms α and β of the OvO MMD are the same as the first term α of the OvA setting, with a careful attention to swapping the subscript a and b as appropriate. Only the negative term γ changes. We once again use Bayes' theorem twice:

$$\gamma = \mathbb{E}_{\boldsymbol{x}_a\sim p_\theta(\boldsymbol{x}|y_a),\boldsymbol{x}_b p_\theta(\boldsymbol{x}|y_b)}\left[\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right] = \mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y_a|\boldsymbol{x}_a)}{p_\theta(y_a)}\frac{p_\theta(y_b|\boldsymbol{x}_b)}{p_\theta(y_b)}\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right]. \quad \text{(C.87)}$$

The final sum is hence similar to the OvA:

$$\mathcal{I}^{\text{OvO}}_{D_{\text{MMD}}}(\boldsymbol{x}; y) = \mathbb{E}_{y_a,y_b\sim p_\theta(y)}\left[D_{\text{MMD}}\left(p_\theta(\boldsymbol{x}|y_a)\|p_\theta(\boldsymbol{x}|y_b)\right)\right], \quad \text{(C.88)}$$
$$= \mathbb{E}_{y_a,y_b\sim p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')\left(\frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_a|\boldsymbol{x}')}{p_\theta(y_a)^2}\right.\right.\right.$$
$$\left.\left.\left.+\frac{p_\theta(y_b|\boldsymbol{x})p_\theta(y_b|\boldsymbol{x}')}{p_\theta(y_b)^2} - 2\frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b|\boldsymbol{x}')}{p_\theta(y_a)p_\theta(y_b)}\right)\right]^{\frac{1}{2}}\right]. \quad \text{(C.89)}$$

C.2.2.2 Gradient

Similarly to the OvA case, we start by unfolding the OvO GEMINI and incorporating the cluster proportions within the square root:

$$\mathcal{I}^{\text{OvO}}_{D_{\text{MMD}}}(\boldsymbol{x}; y) = \sum_{\substack{y_a=1\\y_b=1}}^{K,K} \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')\left(p_\theta(y_a|\boldsymbol{x})p_\theta(y_a|\boldsymbol{x}')p_\theta(y_b)^2\right.\right.$$
$$\left.\left.+p_\theta(y_b|\boldsymbol{x})p_\theta(y_b|\boldsymbol{x}')p_\theta(y_a)^2 - 2p_\theta(y_a|\boldsymbol{x})p_\theta(y_b|\boldsymbol{x}')p_\theta(y_a)p_\theta(y_b)\right)\right]^{\frac{1}{2}}. \quad \text{(C.90)}$$

The gradient will then be a sum of terms proportional to their respective inverse MMD and cluster proportions. Skipping this step as it its identical to the OvA case, we directly focus on the gradient of the 3 inner terms. For the first term, we have:

$$\frac{\partial p_\theta(y_a|\boldsymbol{x})p_\theta(y_a|\boldsymbol{x}')p_\theta(y_b)^2}{\partial\theta} = p_\theta(y_a|\boldsymbol{x}')p_\theta(y_b)^2\frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial\theta} + p_\theta(y_a|\boldsymbol{x})p_\theta(y_b)^2\frac{\partial p_\theta(y_a|\boldsymbol{x}')}{\partial\theta}$$
$$+ 2p_\theta(y_a|\boldsymbol{x})p_\theta(y_a|\boldsymbol{x}')p_\theta(y_b)\mathbb{E}_{\boldsymbol{x}''\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{\partial p_\theta(y_b|\boldsymbol{x}'')}{\partial\theta}\right]. \quad \text{(C.91)}$$

Incorporating it in the double expectation over the data distribution and multipliying with the kernel simplifies the first terms:

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{\partial\alpha}{\partial\theta}\right] = 2p_\theta(y_b)^2\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')p_\theta(y_a|\boldsymbol{x}')\right]\frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial\theta}\right] + 2p_\theta(y_b)\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{\boldsymbol{x}',\boldsymbol{x}''\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x}'.\boldsymbol{x}'')p_\theta(y_a|\boldsymbol{x}')p_\theta(y_a|\boldsymbol{x}'')\right]\frac{\partial p_\theta(y_b|\boldsymbol{x})}{\partial\theta}\right]. \quad \text{(C.92)}$$

The second term β has the exact same equation with a permutation of the a and b subscripts. We can compute the last term:

$$\begin{aligned}\frac{\partial p_\theta(y_a|\boldsymbol{x})p_\theta(y_b|\boldsymbol{x}')p_\theta(y_a)p_\theta(y_b)}{\partial\theta} &= p_\theta(y_b|\boldsymbol{x}')p_\theta(y_a)p_\theta(y_b)\frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial\theta} \\ &+ p_\theta(y_a|\boldsymbol{x})p_\theta(y_a)p_\theta(y_b)\frac{\partial p_\theta(y_b|\boldsymbol{x}')}{\partial\theta} \\ &+ p_\theta(y_a|\boldsymbol{x})p_\theta(y_b|\boldsymbol{x}')p_\theta(y_b)\mathbb{E}_{\boldsymbol{x}''\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{\partial p_\theta(y_a|\boldsymbol{x}'')}{\partial\theta}\right] \\ &+ p_\theta(y_a|\boldsymbol{x})p_\theta(y_b|\boldsymbol{x}')p_\theta(y_a)\mathbb{E}_{\boldsymbol{x}''\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{\partial p_\theta(y_b|\boldsymbol{x}'')}{\partial\theta}\right]. \quad \text{(C.93)}\end{aligned}$$

Once inserted inside the double expectation over the data distribution, we obtain:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{\partial\gamma}{\partial\theta}\right] &= p_\theta(y_a)p_\theta(y_b)\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')p_\theta(y_b|\boldsymbol{x}')\right]\frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial\theta}\right. \\ &\left.+\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')p_\theta(y_a|\boldsymbol{x}')\right]\frac{\partial p_\theta(y_b|\boldsymbol{x})}{\partial\theta}\right] + \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')p_\theta(y_a|\boldsymbol{x})p_\theta(y_b|\boldsymbol{x}')\right] \\ &\times\left(p_\theta(y_b)\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial\theta}\right] + p_\theta(y_a)\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{\partial p_\theta(y_b|\boldsymbol{x})}{\partial\theta}\right]\right). \quad \text{(C.94)}\end{aligned}$$

Now, we can add all terms together and factorise with respect to each differential factor. To lighten the expression, we will only write the factors in front of the differential $\frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial\theta}$ because those weighting the differential $\frac{\partial p_\theta(y_b|\boldsymbol{x})}{\partial\theta}$ are the same through permutation of the a and b subscripts:

$$\begin{aligned}\frac{\partial\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\alpha+\beta-2\gamma\right]}{\partial\theta} &= 2\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\left\{\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x},\boldsymbol{x}')p_\theta(y_b)(p_\theta(y_b)p_\theta(y_a|\boldsymbol{x})\right.\right.\right. \\ &\left.\left.-p_\theta(y_b|\boldsymbol{x})p_\theta(y_a))\right] + \mathbb{E}_{\boldsymbol{x}',\boldsymbol{x}''\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x}',\boldsymbol{x}'')(p_\theta(y_a)p_\theta(y_b|\boldsymbol{x}')p_\theta(y_b|\boldsymbol{x}'')\right. \\ &\left.\left.\left.-p_\theta(y_b)p_\theta(y_a|\boldsymbol{x}')p_\theta(y_b|\boldsymbol{x}''))\right]\right\}\frac{\partial p_\theta(y_a|\boldsymbol{x})}{\partial\theta} + \{\ldots\}\frac{\partial p_\theta(y_b|\boldsymbol{x})}{\partial\theta}\right]. \quad \text{(C.95)}\end{aligned}$$

The final gradient of the OvO-MMD-GEMINI therefore the incorporation of the equation above in a sum weighted by the inverse MMD distance and the matching cluster proportions of the GEMINI:

$$\frac{\partial \mathcal{I}^{\text{OvO}}_{D_{\text{MMD}}}(\boldsymbol{x}; y)}{\partial \theta} = \sum_{\substack{y_a=1 \\ y_b=1}}^{K,K} \frac{1}{2p_\theta(y_a)p_\theta(y_b)D_{\text{MMD}}\left(p_\theta(\boldsymbol{x}|y_a)\|p_\theta(\boldsymbol{x}|y_b)\right)} \times \frac{\partial \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p_{\text{data}}(\boldsymbol{x})}\left[\alpha+\beta-2\gamma\right]}{\partial \theta}. \quad \text{(C.96)}$$

Naturally, the gradient to optimise upon implementation would merge the differential over the different variables y_a and y_b into a single variable y. However, the length of such expression given the inverse MMD distance would be too long to write out here. Instead, we propose to take a look at the effective implementation of the MMD-GEMINI using matrix multiplications and element-wise operations in App. D which provides an automated easy-to-write gradient.

C.3 Gradients for the Wasserstein-GEMINI

We seek the expression of the gradient of the Wasserstein-GEMINI for some output $\boldsymbol{\tau} \in \mathbb{R}^{N\times K}$ of some probabilistic model. The matrix $\boldsymbol{\tau}$ is therefore row-stochastic. The expression of the GEMINI in the one-vs-all context is then:

$$\mathcal{I}^{\text{OvA}}_{D_{\mathcal{W}}}(\boldsymbol{x}; y|\theta) = \mathbb{E}_{y\sim p_\theta(y)}\left[D_{\mathcal{W}}\left(p_\theta(\boldsymbol{x}|y)\|p_{\text{data}}(x)\right)\right], \quad \text{(C.97)}$$

and the one-vs-one variant simply replaces the data distribution with another cluster distribution on which to perform the expectation as well. The distance between the samples is noted δ. During training, the model does not see continuous distribution and only gets batches of samples. Hence, the problem is discretised and the Wasserstein distance can be then evaluated using histogram vectors. We demonstrated in Section 3.3.2.3 that these histogram vectors consist in a cluster-wise normalisation of the predictions which arises from importance sampling. Thus, the discrete approximation of the Wasserstein-GEMINI is:

$$\hat{\mathcal{I}}^{\text{OvA}}_{D_{\mathcal{W}}}(\boldsymbol{x}; y|\theta) = \sum_{k=1}^{K} \pi_k \min_{\boldsymbol{P}\in U(\boldsymbol{\omega}_k, \mathbf{1}_N/N)} \sum_{\substack{i=1 \\ j=1}}^{N,N} \boldsymbol{P}_{i,j}\delta(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \text{(C.98)}$$

where $\boldsymbol{P}$ is constrained in a set that forces it to have rows summing to the values of $\boldsymbol{\omega}_{\cdot k}$ and columns summing to $\mathbf{1}_N/N$. The vector $\boldsymbol{\omega}_{\cdot k} = \boldsymbol{\tau}_{\cdot k}/\sum_{i=1}^{N}\boldsymbol{\tau}_{ik}$ is the normalised cluster predictions.

C.3.1 Gradient for the Wasserstein distance

The new formulation of the discrete Wasserstein distance corresponds to a linear program and is often referred to as the Kantorovich problem. This problem admits the following dual (Peyré & Cuturi, 2019):

$$D_{\mathcal{W}}\left(\boldsymbol{\omega}_{\cdot 1}\|\boldsymbol{\omega}_{\cdot 2}\right)) = \max_{\substack{(\boldsymbol{u},\boldsymbol{v})\in\mathbb{R}^N\times\mathbb{R}^N \\ \boldsymbol{u}_i+\boldsymbol{v}_j\leq\delta_{ij},\forall i,j\leq N}} \langle \boldsymbol{u}, \boldsymbol{\omega}_{\cdot 1}\rangle + \langle \boldsymbol{v}, \boldsymbol{\omega}_{\cdot 2}\rangle, \quad \text{(C.99)}$$

thanks to the strong duality for linear programs (Bertsimas & Tsitsiklis, 1997, p 148, Theorem 4.4). It immediately appears that once we found the optimal "Kantorovich potentials" $\boldsymbol{u}^\star$ and $\boldsymbol{v}^\star$ for each

respective histogram vector $\boldsymbol{\omega}_{\cdot 1}$ and $\boldsymbol{\omega}_{\cdot 2}$ we can compute the gradient of the distance using these optimal values because we remove the $\max$ term. However, as we want to remain in the simplex, we need to recenter the mass of a gradient and thus substract the mean of the dual variables:

$$\frac{\partial D_{\mathcal{W}}\left(\boldsymbol{\omega}_{\cdot 1} \| \boldsymbol{\omega}_{\cdot 2}\right))}{\partial \boldsymbol{\omega}_{\cdot 1}} = \boldsymbol{u}^{\star} - \sum_{i=1}^{N} \frac{\boldsymbol{u}_i^{\star}}{N} = \bar{\boldsymbol{u}}, \tag{C.100}$$

$$\frac{\partial D_{\mathcal{W}}\left(\boldsymbol{\omega}_{\cdot 1} \| \boldsymbol{\omega}_{\cdot 2}\right))}{\partial \boldsymbol{\omega}_{\cdot 2}} = \boldsymbol{v}^{\star} - \sum_{i=1}^{N} \frac{\boldsymbol{v}_i^{\star}}{N} = \bar{\boldsymbol{v}}. \tag{C.101}$$

C.3.2 Complete gradient for the OvA Wasserstein

We can now simply unfold the rules of derivation w.r.t. $\boldsymbol{\tau}_{ik}$ between the product of the cluster proportions π_k and the Wasserstein distance. However, we must take into account that due to the self-normalisation of $\boldsymbol{\tau}$ to produce the histogram vectors $\boldsymbol{\omega}$, we have to sum its derivative over all normalised samples. Thus:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\tau}_{ik}} = \sum_{k'=1}^{K} D_{\mathcal{W}}\left(\boldsymbol{\omega}_{\cdot k'} \| \mathbf{1}_N / N\right) \frac{\partial \pi_{k'}}{\partial \boldsymbol{\tau}_{ik}} + \sum_{j=1}^{N} \pi_{k'} \frac{\partial D_{\mathcal{W}}\left(\boldsymbol{\omega}_{\cdot k'} \| \mathbf{1}_N / N\right)}{\partial \boldsymbol{\omega}_{jk'}} \frac{\partial \boldsymbol{\omega}_{jk'}}{\partial \boldsymbol{\tau}_{ik}}, \tag{C.102}$$

$$= \frac{D_{\mathcal{W}}\left(\boldsymbol{\omega}_{\cdot k} \| \mathbf{1}_N\right)}{N} + \pi_k \sum_{j=1}^{N} \bar{\boldsymbol{u}}_{jk} \left(\frac{\mathbb{1}[i == j]}{N \pi_k} - \frac{\boldsymbol{\tau}_{jk}}{N^2 \pi_k^2} \right), \tag{C.103}$$

where $\mathbb{1}$ is the indicator function resulting from the derivative of the self normalisation. After summing over all samples, we can conclude that the gradient of the one-vs-all Wasserstein-GEMINI w.r.t. model predictions $\boldsymbol{\tau}$ is:

$$\frac{\partial \hat{\mathcal{I}}_{D_{\mathcal{W}}}^{\text{OvA}}}{\partial \boldsymbol{\tau}_{ik}} = \frac{D_{\mathcal{W}}\left(\boldsymbol{\omega}_{\cdot k} \| \mathbf{1}_N\right)}{N} + \frac{\bar{\boldsymbol{u}}_{ik}}{N} - \frac{\langle \bar{\boldsymbol{u}}_{\cdot k}, \boldsymbol{\tau}_k \rangle}{N^2 \pi_k}. \tag{C.104}$$

C.3.3 Complete gradient for the OvO Wasserstein

The demonstration follows the same rules as before. We add as well the fact that the Wasserstein distance is symmetric, and hence its gradient is as well so we can permute the names $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{v}}$ when changing $D_{\mathcal{W}}\left(\boldsymbol{\omega}_1 \| \boldsymbol{\omega}_2\right)$ for $D_{\mathcal{W}}\left(\boldsymbol{\omega}_2 \| \boldsymbol{\omega}_1\right)$. Therefore, we sum twice the gradients of the Wasserstein distances, as well as twice the gradients for the proportions due to the symmetric nature of this function. We can thus arrive to a final gradient that is very similar to the OvO scenario with an additional summing over adversarial proportions:

$$\frac{\partial \hat{\mathcal{I}}_{D_{\mathcal{W}}}^{\text{OvO}}}{\partial \boldsymbol{\tau}_{ik}} = \sum_{k'=1}^{K} 2\frac{\pi_k D_{\mathcal{W}}\left(\boldsymbol{\omega}_{\cdot k} \| \boldsymbol{\omega}_{\cdot k'}\right)}{N} + 2\frac{\pi_{k'} \bar{\boldsymbol{u}}_{j,k/k'}}{N} - 2\frac{\langle \bar{\boldsymbol{u}}_{\cdot k/k'} \boldsymbol{\tau}_k \rangle}{N^2 \pi_k}. \tag{C.105}$$

Notice that we detailed in subscript for which Wasserstein evaluation a dual variable emerges using the notation k/k'. Since the one-vs-one GEMINI makes K^2 distance evaluation, we have K^2 dual variables as well when removing duplicate dual variables due to symmetry.

APPENDIX D

Implementing MMD-GEMINI using matrix multiplications

In this appendix, we provide a practical view on the implementation of the MMD and its gradient when we have a fixed batch of size N and K clusters. We consider the computations starting from a row-stochastic matrix $\boldsymbol{\tau} \in \mathbb{R}^{N\times K}$, typically the softmax output of a model. The provided gradients are of course not as generic as the ones provided in App. C. We assume in this appendix that the reader is familiar with the definition of the OvA and OvO MMD-GEMINIs.

D.1 OvA scenario

D.1.1 Alternative computation of the forward pass

We focus here only on the computations of the objective function, the OvA MMD. First, we can compute the cluster proportions:

$$\boldsymbol{\pi} = \frac{1}{N}\mathbf{1}_N^\top \boldsymbol{\tau}. \tag{D.1}$$

Our goal is to compute the vector $\boldsymbol{\Delta} \in \mathbb{R}^K$ where the k-the component is the squared distance in the Hilbert space between one cluster distribution and the data distribution:

$$\boldsymbol{\Delta}_k = \sum_{i,j}^{N,N} \tilde{\kappa}_{i,j} \left[\frac{\boldsymbol{\tau}_{ki}\boldsymbol{\tau}_{kj}}{\pi_k^2} + 1 - 2\frac{\boldsymbol{\tau}_{ki}}{\pi_k} \right]. \tag{D.2}$$

To that end, we introduce the matrix $\boldsymbol{\alpha} \in \mathbb{R}^{N\times K}$ which is the element-wise division of $\boldsymbol{\tau}$ by the proportions of the matching cluster.

$$\boldsymbol{\alpha} = \boldsymbol{\tau} \oslash (\mathbf{1}_N \boldsymbol{\pi}^\top) = \left[\frac{\boldsymbol{\tau}_{ki}}{\pi_k} \right]. \tag{D.3}$$

Individually, we can interprete the value of $\alpha_i k$ as the ratio $p(y = k|\boldsymbol{x}_i)/p(y = k)$ or $p(\boldsymbol{x}_i|y = k)/p(\boldsymbol{x}_i)$. This represents the relative strength of the sample in the cluster distribution. We can then deduce the writing of $\boldsymbol{\Delta}$:

$$\boldsymbol{\Delta} = \operatorname{diag}\left(\boldsymbol{\alpha}^\top \tilde{\boldsymbol{\kappa}} \boldsymbol{\alpha}\right) + \mathbf{1}_{K\times N}\tilde{\boldsymbol{\kappa}}\mathbf{1}_N - 2\boldsymbol{\alpha}^\top \tilde{\boldsymbol{\kappa}}\mathbf{1}_N = \boldsymbol{a} + \boldsymbol{c} - 2\boldsymbol{b}. \tag{D.4}$$

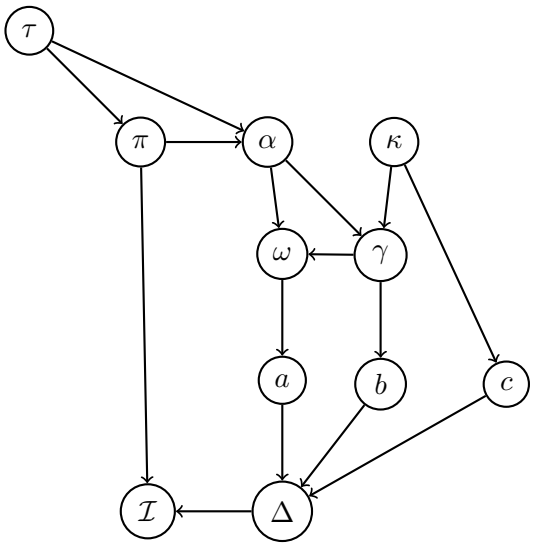


Figure D.1 – Summary of computations for the forward pass of the OvA MMD-GEMINI using matrices

To name the elements, $\boldsymbol{a}$ is the cluster contribution: how the distribution of the cluster contributes to increase the distance, and the same goes for the constant $\boldsymbol{c}$ which represents the agnostic data strength. Finally, $\boldsymbol{b}$ is the agreement between the two distributions $p(y| = k|\boldsymbol{x})$ and $p(\boldsymbol{x})$ which diminishes the value of the MMD: the more the cluster distribution takes to the data (by having everything in the same cluster), the stronger $\boldsymbol{b}$ is and the lower the MMD. Yet, to simplify the derivatives to compute later, we introduce two intermediary variables:

$$\boldsymbol{\gamma} = \tilde{\boldsymbol{\kappa}}\boldsymbol{\alpha}, \tag{D.5}$$

and

$$\boldsymbol{\omega} = \boldsymbol{\alpha}^\top \boldsymbol{\gamma}, \tag{D.6}$$

of respective shapes $N \times K$ and $K \times K$. Thus, we simply rewrite:

$$\boldsymbol{\Delta} = \boldsymbol{a} + \boldsymbol{c} - 2\boldsymbol{b} = \text{diag}(\boldsymbol{\omega}) + \mathbf{1}_{K\times N}\tilde{\boldsymbol{\kappa}}\mathbf{1}_N - 2\boldsymbol{\gamma}^\top \mathbf{1}_N. \tag{D.7}$$

Finally, assuming the square root is applied element-wise, we can write the final objective as:

$$\hat{\mathcal{I}}^{\text{OvA}}_{\text{MMD}}(\boldsymbol{x}, y|\theta) = \boldsymbol{\pi}^\top \sqrt{\boldsymbol{\Delta}}. \tag{D.8}$$

The graph of computations is summarised in Figure D.1.

D.1.2 Gradient

We can now compute the derivatives of each part of the graph with respect to the conditional probabilities: $\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\tau}}$. By reversing the graph, we get the list of the following sorted derivatives to compute: (1) $\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\Delta}}$ (2) $\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{a}}$ (3) $\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{b}}$ (4) $\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\omega}}$ (5) $\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\gamma}}$ (6) $\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\alpha}}$ (7) $\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\pi}}$ (8) $\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\tau}}$.

To compute these derivatives, we will follow an automatic differentiation procedure. All derivatives correspond to the gradient of a scalar with respect to a matrix or vector, hence all derivatives will keep the same shape as the denominator. Notice that because c and $\tilde{\kappa}$ do not depend on $\boldsymbol{\tau}$, they will not produce any gradient.

D.1.2.1 Deriving for Δ

We simply take the vector π for this derivative that summed the square root of Δ. Additionally, the element-wise square root is differentiated. Thus:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\Delta}} = \frac{\boldsymbol{\pi}}{2\sqrt{\Delta}}. \tag{D.9}$$

D.1.2.2 Deriving for $\boldsymbol{a}$ and $\boldsymbol{b}$

The contributions of $\boldsymbol{a}$ and $\boldsymbol{b}$ are simple element-wise sums of vectors. They have the same shape as the previous gradient. Therefore:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{a}} = \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\Delta}} = \frac{\boldsymbol{\pi}}{2\sqrt{\Delta}}, \tag{D.10}$$

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{b}} = -2\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\Delta}} = -\frac{\boldsymbol{\pi}}{\sqrt{\Delta}}. \tag{D.11}$$

D.1.2.3 Deriving for ω

Since we took only the diagonal of ω for computation, the gradient with respect to ω will be a diagonal matrix, which diagonal is exactly the previously committed error:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\omega}} = \text{diag}\left(\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{a}}\right) = \frac{1}{2}\text{diag}\left(\frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}}\right). \tag{D.12}$$

D.1.2.4 Deriving for γ

The vector γ contributed two times in the computation graph: once to $\boldsymbol{\omega}$ and another time for $\boldsymbol{b}$. Both cases involve simple matrix multiplications. We can sum the matrix gradient to both errors to get:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\gamma}} = \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\omega}}\boldsymbol{\alpha}^\top + \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{b}}\mathbf{1}_N^\top, \tag{D.13}$$

$$= \frac{1}{2}\boldsymbol{\alpha}\text{diag}\left(\frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}}\right) - \mathbf{1}_N\left(\frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}}\right)^\top. \tag{D.14}$$

Notice that for the derivative from $\boldsymbol{b}$, we had to transpose the error since $\boldsymbol{b}$ is computed using $\boldsymbol{\gamma}^\top$. Here, we can remark that by unfolding the definition of $\boldsymbol{\alpha}$, and thanks to matrix product with the diagonal product, the values of $\boldsymbol{\pi}$ gets cancelled. Therefore:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\gamma}} = \frac{1}{2}\frac{\boldsymbol{\tau}}{\mathbf{1}_N\sqrt{\boldsymbol{\Delta}}^\top} - \mathbf{1}_N\left(\frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}}\right)^\top. \tag{D.15}$$

D.1.2.5 Deriving for α

As we did for $\boldsymbol{\gamma}$, we need to sum here the gradient contributions of $\boldsymbol{\alpha}$ to $\boldsymbol{\omega}$ and $\boldsymbol{\gamma}$. Both are matrix multiplications, however we add a transposition in the case of ω. Thus:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\alpha}} = \left(\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\omega}}\boldsymbol{\gamma}^\top\right)^\top + \tilde{\boldsymbol{\kappa}}^\top\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\gamma}}, \tag{D.16}$$

$$= \frac{1}{2}\boldsymbol{\gamma}\text{diag}\left(\frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}}\right) + \tilde{\boldsymbol{\kappa}}\left(\frac{1}{2}\frac{\boldsymbol{\tau}}{\mathbf{1}_N\sqrt{\boldsymbol{\Delta}}^\top} - \mathbf{1}_N\left(\frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}}\right)^\top\right). \tag{D.17}$$

Here, we can unfold the definition of $\boldsymbol{\gamma}$ to make a common factor $\tilde{\boldsymbol{\kappa}}$ appear on the left matrix multiplication. Thus:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\alpha}} = \tilde{\boldsymbol{\kappa}}\left[\frac{1}{2}\boldsymbol{\alpha}\text{diag}\left(\frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}}\right) + \frac{1}{2}\frac{\boldsymbol{\tau}}{\mathbf{1}_N\sqrt{\boldsymbol{\Delta}}^\top} - \mathbf{1}_N\left(\frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}}\right)^\top\right]. \tag{D.18}$$

We notice the exact same simplification on the left term between $\boldsymbol{\alpha}$ and the diagonal matrix as we had for the gradient w.r.t. $\boldsymbol{\gamma}$. Rewriting this term is exactly equal to the second, and thus:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\alpha}} = \tilde{\boldsymbol{\kappa}}\left[\frac{\boldsymbol{\tau}}{\mathbf{1}_N\sqrt{\boldsymbol{\Delta}}^\top} - \mathbf{1}_N\left(\frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}}\right)^\top\right]. \tag{D.19}$$

D.1.2.6 Deriving for π

Same procedure for $\boldsymbol{\pi}$ by summing the contributions of the gradient from $\boldsymbol{\alpha}$ and the dot product with $\sqrt{\boldsymbol{\Delta}}$ in $\mathcal{I}$. For the derivative from $\boldsymbol{\alpha}$, we multiply the rows of the previous error by the squared inverse of $\boldsymbol{\pi}$ and $\boldsymbol{\beta}$ and sum them. Hence:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\pi}} = \sqrt{\boldsymbol{\Delta}} - \left[\left[\frac{\boldsymbol{\tau}^\top}{\sqrt{\boldsymbol{\Delta}}\mathbf{1}_N^\top} - \frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}}\mathbf{1}_N^\top\right]\tilde{\boldsymbol{\kappa}} \odot \frac{\boldsymbol{\beta}}{\boldsymbol{\pi}^2\mathbf{1}_N^\top}\right]\mathbf{1}_N, \tag{D.20}$$

$$= \sqrt{\boldsymbol{\Delta}} - \left[\left[\frac{\boldsymbol{\tau}^\top}{\sqrt{\boldsymbol{\Delta}}\mathbf{1}_N^\top} - \frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}}\mathbf{1}_N^\top\right]\tilde{\boldsymbol{\kappa}} \odot \frac{\boldsymbol{\alpha}}{\boldsymbol{\pi}\mathbf{1}_N^\top}\right]\mathbf{1}_N. \tag{D.21}$$

Since the inverse factor $1/\boldsymbol{\pi}\mathbf{1}_N^\top$ is constant row-wise, we can incorporate it directly to the left term of the matrix multiplication. This simplifies again the notations:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\pi}} = \sqrt{\boldsymbol{\Delta}} + \left[\left(\frac{\boldsymbol{\alpha}^\top\tilde{\boldsymbol{\kappa}}}{\sqrt{\boldsymbol{\Delta}}\mathbf{1}_N^\top} - \frac{\mathbf{1}_N^\top\tilde{\boldsymbol{\kappa}}}{\sqrt{\boldsymbol{\Delta}}\mathbf{1}_N^\top}\right) \odot \boldsymbol{\alpha}\right]\mathbf{1}_N. \tag{D.22}$$

Here, the combination of the element-wise multiplication by $\boldsymbol{\alpha}$ followed by a sum over all samples is in fact equal to the respective distance terms $\boldsymbol{a}$ and $\boldsymbol{b}$. First first simplification yields:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\pi}} = \sqrt{\boldsymbol{\Delta}} - \frac{\boldsymbol{a} - \boldsymbol{b}}{\sqrt{\Delta}}. \tag{D.23}$$

To finally go further, we can use the definition of $\boldsymbol{\Delta}$ to replace the left term by another. Indeed, since:

$$\boldsymbol{a} - \boldsymbol{b} = \boldsymbol{\Delta} + \boldsymbol{b} - \boldsymbol{c}, \tag{D.24}$$

and the denominator $\sqrt{\Delta}$ gets cancelled by $\boldsymbol{\Delta}$, we obtain:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\pi}} = \frac{\boldsymbol{c} - \boldsymbol{b}}{\sqrt{\Delta}}. \tag{D.25}$$

D.1.2.7 Deriving for τ

Finally, the gradient for $\boldsymbol{\tau}$ sums contributions from both $\boldsymbol{\alpha}$ and $\boldsymbol{\pi}$. In both cases, we just consider element-wise operations, so the global gradient will just be element-wise multiplication of the errors, with a specific repetition over all rows for the gradient from $\boldsymbol{\pi}$:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\tau}} = \frac{\mathbf{1}_N}{N} \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\pi}}^\top + \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\alpha}} \odot \frac{1}{\mathbf{1}_N \boldsymbol{\pi}^\top}, \tag{D.26}$$

$$= \mathbf{1}_N \frac{\boldsymbol{c} - \boldsymbol{b}}{N\sqrt{\Delta}}^\top + \tilde{\boldsymbol{\kappa}} \left[\frac{\boldsymbol{\tau}}{\mathbf{1}_N \sqrt{\boldsymbol{\Delta}}^\top} - \mathbf{1}_N \left(\frac{\boldsymbol{\pi}}{\sqrt{\boldsymbol{\Delta}}} \right)^\top \right] \odot \frac{1}{\mathbf{1}_N \boldsymbol{\pi}^\top}, \tag{D.27}$$

$$= \mathbf{1}_N \frac{\boldsymbol{c} - \boldsymbol{b}}{N\sqrt{\Delta}}^\top + \tilde{\boldsymbol{\kappa}} \left[\frac{\boldsymbol{\alpha}}{\mathbf{1}_N \sqrt{\boldsymbol{\Delta}}^\top} - \frac{1}{\mathbf{1}_N \sqrt{\boldsymbol{\Delta}}^\top} \right]. \tag{D.28}$$

To conclude, we can factorise all terms by the common denominator:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\tau}} = \frac{1}{\mathbf{1}_N \sqrt{\boldsymbol{\Delta}}^\top} \odot \left[\frac{\mathbf{1}_N}{N} (\boldsymbol{c} - \boldsymbol{b})^\top + \tilde{\boldsymbol{\kappa}} (\boldsymbol{\alpha} - \mathbf{1}_{N \times K}) \right]. \tag{D.29}$$

For further simplification of the gradients, we can unfold again the definition of $\boldsymbol{b}$ and $\boldsymbol{c}$ as follows:

$$\frac{\mathbf{1}_N}{N} (\boldsymbol{c} - \boldsymbol{b})^\top = \frac{1}{N} \left[\mathbf{1}_{N \times N} \tilde{\boldsymbol{\kappa}} \mathbf{1}_{N \times K} - \mathbf{1}_{N \times N} \tilde{\boldsymbol{\kappa}} \boldsymbol{\alpha} \right], \tag{D.30}$$

$$= \frac{1}{N} \left[\mathbf{1}_{N \times N} \tilde{\boldsymbol{\kappa}} \left(\mathbf{1}_{N \times K} - \boldsymbol{\alpha} \right) \right]. \tag{D.31}$$

Thus, we can conclude that the final equation for the gradient of the OvA MMD is:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\tau}} = \frac{1}{\mathbf{1}_N \sqrt{\boldsymbol{\Delta}}^\top} \odot \left[\left(I_N - \mathbf{1}_{N \times N} / N \right) \tilde{\boldsymbol{\kappa}} \left(\boldsymbol{\alpha} - \mathbf{1}_{N \times K} \right) \right]. \tag{D.32}$$

To be more precise, we can even express the value for a component at position i, k:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\tau}_{i,k}} = \left[\frac{1}{\sqrt{\Delta_k}} \sum_{j=1}^{N} \left(\tilde{\kappa}_{ij} - \frac{1}{N} \sum_{l=1}^{N} \tilde{\kappa}_{jl} \right) (\alpha_{jk} - 1) \right], \tag{D.33}$$

$$= \left[\frac{1}{\sqrt{\Delta_k}} (c - b_k + \gamma_{ik} - \bar{\kappa}_i) \right], \tag{D.34}$$

with $\bar{\kappa}_i = \sum_{j=1}^{N} \tilde{\kappa}_{ij}$.

D.2 OvO scenario

D.2.1 Alternative computation of the forward pass

We will proceed here to the exact same reasoning as in the OvA MMD. We first compute the distance $\boldsymbol{\Delta}$ before summing them with $\boldsymbol{\pi}$. Contrary to the OvA MMD, $\boldsymbol{\Delta}$ is now a matrix of shape $K \times K$ where each entry describes the distance between two clusters k and k':

$$\hat{\mathcal{I}}^{\text{OvO}}_{\text{MMD}}(\boldsymbol{x}, y|\theta) = \boldsymbol{\pi}^{\top} \sqrt{\boldsymbol{\Delta}} \boldsymbol{\pi}. \tag{D.35}$$

As previously done, we can express the squared distance as the sum of two self-contributions minus a cross-contribution. These contributions will be here matrices of shape $K \times K$. Yet, we can notice that in the OvO MMD, the matrix $\boldsymbol{\Delta}$ is symmetric. Simply put, the cross-contribution is symmetric, and the two self-contributions are the transposed of each other:

$$\boldsymbol{\Delta} = \boldsymbol{A} + \boldsymbol{C} - 2\boldsymbol{B}, \tag{D.36}$$

$$= \boldsymbol{A} + \boldsymbol{A}^{\top} - 2\boldsymbol{B}. \tag{D.37}$$

We can here realise that the matrix $\boldsymbol{A}$ is in fact a column-wise copy of the vector $\boldsymbol{a}$ from the previous computations with OvA MMD. Similarly, $\boldsymbol{B}$ is the entire matrix $\boldsymbol{\omega}$ while $\boldsymbol{A}$ only consists in its diagonal. Therefore:

$$\boldsymbol{\Delta} = \text{diag}(\boldsymbol{\omega}) \mathbf{1}_K^{\top} + \mathbf{1}_K \text{diag}(\boldsymbol{\omega})^{\top} - 2\boldsymbol{\omega}. \tag{D.38}$$

The remaining of the definition of $\boldsymbol{\omega}$ strictly unfolds from the OvA MMD forward pass.

D.2.2 Gradient

In the specific case of the OvO, we have square roots of values of $\boldsymbol{\Delta}$ which can be equal to 0, hence undifferentiable. This is in fact not a burden since in principle, these 0 only happen when we evaluate the MMD between a cluster and itself. Thus, we can discard easily the null components of $\boldsymbol{\Delta}$ during the final sum (expectation over π) and adopt locally the small convention that the derivative of $\mathcal{I}$ w.r.t. $\boldsymbol{\Delta}$ will be equal to 0 on the diagonal, despite the square root computation.

D.2.2.1 Deriving for $\boldsymbol{\Delta}$

We start simple, the derivative is simply a square matrix where all components are the cartesian product of the vector $\boldsymbol{\pi}$:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\Delta}} = \frac{\pi\pi^\top}{2\sqrt{\Delta}}. \tag{D.39}$$

From now on, we will arbitrarily say that $\left(\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\Delta}}\right)_{k,k} = 0$ because it was not summed at the end of the forward pass in the OvO MMD. Thus, we will write for clarity:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\Delta}} = \frac{\pi\pi^\top}{2\sqrt{\Delta}} \odot (\mathbf{1}_{K\times K} - \boldsymbol{I}_K). \tag{D.40}$$

D.2.2.2 Deriving for $\boldsymbol{\omega}$

For the gradient w.r.t. $\boldsymbol{\omega}$, we have two contributions to sum, one which comes from the diagonal of $\boldsymbol{\omega}$ times 2, and another from the complete matrix $\boldsymbol{\omega}$:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\omega}} = \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\Delta}} \left(\frac{\partial \boldsymbol{\Delta}}{\partial \text{diag}(\boldsymbol{\omega})} \frac{\partial \text{diag}(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} + \frac{\partial \boldsymbol{\Delta}}{\partial \text{diag}(\boldsymbol{\omega})^\top} \frac{\partial \text{diag}(\boldsymbol{\omega})^\top}{\partial \boldsymbol{\omega}} \right) - 2 \times \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\Delta}} \frac{\partial \boldsymbol{\Delta}}{\partial \boldsymbol{\omega}}, \tag{D.41}$$

$$= 2\text{diag}\left(\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\Delta}} \mathbf{1}_K \right) - 2\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\Delta}}. \tag{D.42}$$

We can simplify the factor 2 to get:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\omega}} = \text{diag}\left(\left[\frac{\boldsymbol{\pi}\boldsymbol{\pi}^\top}{\sqrt{\boldsymbol{\Delta}}} \odot (\mathbf{1}_{K\times K} - \boldsymbol{I}_K) \right] \mathbf{1}_K \right) - \frac{\boldsymbol{\pi}\boldsymbol{\pi}^\top}{\sqrt{\boldsymbol{\Delta}}} \odot (\mathbf{1}_{K\times K} - \boldsymbol{I}_K). \tag{D.43}$$

This gradient says that on all parts of the matrix except the diagonal, we backpropagate the cross-contribution from $\boldsymbol{\omega}$, but sum all this contributions as well on the diagonal. To ease later writings, we introduce $\boldsymbol{\Lambda}$:

$$\boldsymbol{\Lambda} = \frac{\boldsymbol{\pi}\boldsymbol{\pi}^\top}{\sqrt{\boldsymbol{\Delta}}} \odot (\mathbf{1}_{K\times K} - \boldsymbol{I}_K). \tag{D.44}$$

Thanks to $\boldsymbol{\Delta}$ and the cross-product of the vector $\boldsymbol{\pi}$, $\boldsymbol{\Lambda}$ is positive and symetric.

D.2.2.3 Deriving for $\boldsymbol{\gamma}$

Now, we can backpropagate as we did for the OvA MMD, except that γ only contributed once to the computation of $\boldsymbol{\omega}$. Thus:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\gamma}} = \boldsymbol{\alpha} \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\omega}}, \tag{D.45}$$

$$= \boldsymbol{\alpha} \left[\text{diag}\left(\boldsymbol{\Lambda} \mathbf{1}_K \right) - \boldsymbol{\Lambda} \right]. \tag{D.46}$$

D.2.2.4 Deriving for $\boldsymbol{\alpha}$

The derivative w.r.t. $\boldsymbol{\alpha}$ is a backpropagation through two matrix multiplications: one in $\boldsymbol{\gamma}$ and one in $\boldsymbol{\omega}$. Hence:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\alpha}} = \tilde{\boldsymbol{\kappa}} \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\gamma}} + \boldsymbol{\gamma} \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\omega}}, \tag{D.47}$$

$$= \tilde{\boldsymbol{\kappa}} \boldsymbol{\alpha} \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\omega}} + \boldsymbol{\gamma} \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\omega}}, \tag{D.48}$$

$$= 2\boldsymbol{\gamma} \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\omega}}. \tag{D.49}$$

Thus:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\alpha}} = 2\boldsymbol{\gamma} \left[\text{diag} \left(\boldsymbol{\Lambda} \mathbf{1}_K \right) - \boldsymbol{\Lambda} \right]. \tag{D.50}$$

Note that we can write the first term differently because we multiply the matrix $\boldsymbol{\gamma}$ by a diagonal matrix. This is equivalent to doing an element-wise multiplication of $\boldsymbol{\gamma}$ by the vector inside the diag function repeated row-wise:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\alpha}} = 2\mathbf{1}_{N \times K} \boldsymbol{\Lambda} \odot \boldsymbol{\gamma} - 2\boldsymbol{\gamma}\boldsymbol{\Lambda}. \tag{D.51}$$

D.2.2.5 Deriving for $\boldsymbol{\pi}$

The proportions contributed in the final expectation with $\boldsymbol{\Delta}$ and in the computations of $\boldsymbol{\alpha}$. This looks like what we had in the OvA MMD backpropagation. Therefore:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\pi}} = \frac{\partial \boldsymbol{\pi}^\top \sqrt{\Delta} \boldsymbol{\pi}}{\partial \boldsymbol{\pi}} - \mathbf{1}_N^\top \left[\frac{\boldsymbol{\alpha}}{\mathbf{1}_N \boldsymbol{\pi}^\top} \odot \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\alpha}} \right], \tag{D.52}$$

$$= 2\boldsymbol{\pi}^\top \sqrt{\Delta} - 2 \times \mathbf{1}_N^\top \left[\frac{\boldsymbol{\alpha}}{\mathbf{1}_N \boldsymbol{\pi}^\top} \odot \left(\mathbf{1}_{N \times K} \boldsymbol{\Lambda} \odot \boldsymbol{\gamma} - \boldsymbol{\gamma}\boldsymbol{\Lambda} \right) \right]. \tag{D.53}$$

By noticing that the matrix $\mathbf{1}_{N \times K} \boldsymbol{\Lambda}$ is constant row-wise, we can easily permute the element-wise operation with $\boldsymbol{\gamma}$ and perform the matrix multiplication with $\mathbf{1}_N^\top$ before thanks to factorisation. The element-wise product of $\boldsymbol{\alpha}$ with $\boldsymbol{\gamma}$ summed over all samples is equal to the diagonal of $\boldsymbol{\omega}$. We can rewrite:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\pi}} = 2\boldsymbol{\pi}^\top \sqrt{\Delta} - 2 \frac{\text{diag}(\boldsymbol{\omega})}{\boldsymbol{\pi}^\top} \odot \mathbf{1}_K \boldsymbol{\Lambda} + \frac{2}{\boldsymbol{\pi}^\top} \odot \left(\mathbf{1}_N^\top \left[\boldsymbol{\alpha} \odot \boldsymbol{\gamma}\boldsymbol{\Lambda} \right] \right). \tag{D.54}$$

D.2.2.6 Deriving for $\boldsymbol{\tau}$

We can finally draw a conclusion to this backpropagation by summing the gradient of the two contributions of $\boldsymbol{\tau}$: one from $\boldsymbol{\alpha}$ and another one from $\boldsymbol{\pi}$:

$$\frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\tau}} = \frac{1}{\mathbf{1}_N \boldsymbol{\pi}^\top} \odot \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\alpha}} + \frac{1}{N} \mathbf{1}_N \frac{\partial \hat{\mathcal{I}}}{\partial \boldsymbol{\pi}}, \tag{D.55}$$

$$= \frac{2}{\mathbf{1}_N \boldsymbol{\pi}^\top} \odot [\boldsymbol{\gamma} \odot \mathbf{1}_{N\times K} \boldsymbol{\Lambda} - \boldsymbol{\gamma}\boldsymbol{\Lambda}] + \frac{2}{N} \mathbf{1}_N \boldsymbol{\pi}^\top \sqrt{\boldsymbol{\Delta}} - \frac{2}{N} \mathbf{1}_N \left[\frac{\operatorname{diag}(\boldsymbol{\omega})}{\boldsymbol{\pi}^\top} \odot \mathbf{1}_K \boldsymbol{\Lambda} \right] + \frac{2}{N} \mathbf{1}_N \left[\frac{1}{\boldsymbol{\pi}^\top} \odot \left(\mathbf{1}_N^\top [\boldsymbol{\alpha} \odot \boldsymbol{\gamma}\boldsymbol{\Lambda}] \right) \right], \tag{D.56}$$

$$= \frac{2}{N} \mathbf{1}_N \boldsymbol{\pi}^\top \sqrt{\boldsymbol{\Delta}} + \frac{2}{\mathbf{1}_N \boldsymbol{\pi}^\top} \odot \left[\boldsymbol{\gamma} \odot \mathbf{1}_{N\times K} \boldsymbol{\Lambda} - \boldsymbol{\gamma}\boldsymbol{\Lambda} - \frac{\mathbf{1}_N}{N} (\operatorname{diag}(\boldsymbol{\omega}) \odot \mathbf{1}_K \boldsymbol{\Lambda}) + \frac{\mathbf{1}_{N\times N}}{N} (\boldsymbol{\alpha} \odot \boldsymbol{\gamma}\boldsymbol{\Lambda}) \right]. \tag{D.57}$$

Thus, we computed the gradient of the OvO-MMD-GEMINI using only matrices for batches of fixed size N.

Apprentissage statistique appliqué à la cardiologie

Clustering discriminant et phénogroupes de la sténose aortique

Louis OHL

Résumé

La sténose de la valve aortique (SA) est une maladie chronique progressive dont la prévalence risque de tripler dans les décennies à venir en Amérique du Nord et par conséquent ses impacts en santé et économie. À l'heure actuelle, aucun médicament contre la SA n'est disponible. La nécessité de pharmacothérapies adaptées pousse donc à l'exploration des différentes causes de la progression de la SA chez les patients. Bien qu'il existe déjà certaines sous-catégories de la SA, ces dernières sont difficiles à identifier et par conséquent à cibler par une thérapie.

Afin de découvrir et identifier des causes potentielles de la SA, nous formulons la recherche de ces phénogroupes en tant que problème de partitionement. Le partitionnement est un problème issu du domaine d'apprentissage automatique consistant à répartitr de multiples observations en groupes nommés *clusters* selon leurs similarités. Afin d'accompagner ce problème d'apprentissage automatique, nous utilisons l'étude sur le progression des déterminants métaboliques de la SA (étude PROGRESSA). L'étude PROGRESSA comprend trois modalités: clinicopathologique, protéomique et radiomique pour 351 patients avec suivi annuel. La structure de PROGRESSA est complexe: elle est de grande dimension avec des variables de natures différentes. De plus, les différentes modalités ne se recouvrent pas nécessairement.

Dans ce contexte, nous formulons le problème de partitionnement à travers un prisme discriminatif, ce qui permet d'intégrer avec facilité des modèles d'apprentissage profond, notamment pour manipuler des données grande dimensions. Ces dernières années ont été marquées par l'arrivée de méthodes de partitionnement profonds, souvent basés sur la maximisation de l'information mutuellee. Cependant, les récents succès de ces méthodes sont souvent spécifique à un type unique de données et ne permettent donc pas d'anticiper leur applicabilité à un problème multi-source.

Afin de construire une solution pour le problème de partitionnement multi-source, cette thèse s'orchestre autour du développement d'un ensemble de méthodes de clustering nommé information mutuelle généralisée (GEMINI) à partir du Chapitre 3. Cet ensemble de méthodes permet d'utiliser n'importe quelle architecture de réseau de neurones profonds sur des données de natures variées. Nous montrons également comment cette méthode peut être améliorée pour incorporer des méchanismes de sélections de vaiables afin de faciliter l'interprétation des clusters au Chapitre 4: Sparse GEMINI. Puis nous complètons le spectre des modèles entraînables par GEMINI avec l'introduction d'arbres non supervisés donnant un clustering avec explication intégrée dans le chapitre 5.

Enfin, nous terminons cette thèse avec un pipeline intégrant divers variants de GEMINI pour la découverte de phénogroupes de la SA dans l'étude PROGRESSA au Chapitre 6. Certains de ces phénogroupes montrent une mortalité accentuée et sont caractérisés par des marqueurs spécifiques, par exemple liés aux lipoprotéines, au diabète ou à la bicuspidie des valves aortiques. Ces phénogroupes peuvent ainsi être ciblés par des thérapies spécifiques afin de réduire le risque de progression de la maladie.

Mots-clés : Partitionnement discriminatif, apprentissage non supervisé, cardiologie, sténose aortique, phénogroupes

Abstract

Aortic valve stenosis (AS) is a chronic progressive disease whose prevalence is likely to triple in the coming decades in North America, with a consequent impact on health and the economy. However, efficient drug therapies for this disease are not available. The need for appropriate medication is therefore driving the exploration of the various causes of AS progression in patients. There exist a few sub-categories of the disease that could be differently targeted by drugs, but they are hard to define and identify.

To alleviate the finding of different possible causes of AS, we formulate the search of phenogroup (i.e. disease subtypes) as a clustering problem. Clustering is a family of approaches from machine learning that consists in gathering multiple observations deemed similar in categories called clusters. To support this machine learning problem instance, we employ the metabolic determinants of the progression of AS study (PROGRESSA study). The PROGRESSA dataset comprises 3 modalities: clinicopathological, proteomics and radiomics data for 351 patients with yearly follow-ups. The structure of the PROGRESSA study is challenging for current clustering algorithms: it is high-dimensional with mixed data types. Moreover, the different modalities of the data do not necessarily overlap, making it to a multi-source clustering problem.

In this context, we formulate the clustering problem through the lens of discriminative clustering: a point of view that leverages the easy integration of deep learning models for handling and concatenating high-dimensional data. Within this framework, the last decade witnessed the impressive rise of deep clustering methods that often involves the maximisation of mutual information. However, the recent success of deep clustering models are often over-specified for one type of data and therefore hardly account for multi-modal data.

To pave the way for a multi-source discriminative clustering algorithm, we developed a set of discriminative clustering methods called generalised mutual information (GEMINI) in Chapter 3. Thanks to its discriminative construction, this set of methods can be used with any deep neural network architecture on data of various types. We also show how this method can be improved to incorporate variable selection mechanisms to facilitate the interpretation of clusters in Chapter 4: Sparse GEMINI. Then, we complete the spectrum of models trainable by GEMINI in Chapter 5 with the introduction of unsupervised trees giving a clustering with integrated explanation.

Finally, we conclude this thesis in Chapter 6 with a pipeline integrating various GEMINI variants for the discovery of AS phenogroups in the PROGRESSA study. Some of these phenogroups show increased mortality and are characterised by specific markers, for example linked to lipoproteins, diabetes or bicuspid aortic valves. These phenogroups can therefore be targeted by specific therapies to reduce the risk of disease progression.

Keywords: Discriminative clustering, unsupervised learning, cardiology, aortic stenosis, phenogroups