VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Scalable Probabilistic Approximation Method in Applications

Metoda Škálovatelné Pravděpodobnostní Aproximace v aplikacích

Matěj Frič

Bachelor Thesis

Supervisor: Ing. Lukáš Pospíšil, Ph.D.

Ostrava, 2023

VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Ref. no.VSB/22/080965
vsboee22026BD0

Bachelor Thesis Assignment

Student: **Matěj Frič**

Study Programme: B0541A170008 Computational and Applied Mathematics

Title: Scalable Propabilistic Approximation method in Applications
Metoda Škálovatelné Pravděpodobnostní Aproximace v aplikacích

The thesis language: English

Description:

The components of data-processing pipelines in the machine learning are characterized by computational costs that grow quickly with the increasing data complexity and size – accompanied by a very moderate improvement of performance. Typical key prerequisite consists of finding reliable discrete approximations / clustering of complex systems, however, common discretization approaches are crucially limited in terms of quality and cost.
The recently developed a low-cost improved-quality Scalable Probabilistic Approximation (SPA) framework allows a simultaneous joint optimal solution of datadriven feature selection, discretization and Bayesian/Markovian model inference problems. The aim of the thesis is to understand this methodology and compare it with standard methods on selected applications.

References:

- Gerber S., Pospisil L., Navandar M., Horenko I.: Low-cost scalable discretization, prediction and feature selection for complex systems, 2019.
- Dostal Z., Beremlijski P.: Metody optimalizace. skriptum Matematika pro inženýry 21. století, 2012.
- Litschmannová M.: Úvod do statistiky. skriptum Matematika pro inženýry 21. století, 2012.

Extent and terms of a thesis are specified in directions for its elaboration that are opened to the public on the web sites of the faculty.

Supervisor: **Ing. Lukáš Pospíšil, Ph.D.**

Date of issue: 01.09.2022

Date of submission: 30.04.2023

Study programme guarantor: prof. RNDr. Jiří Bouchala, Ph.D.

In IS EDISON assigned: 16.11.2022 15:50:47

Abstrakt

Cílem této práce je rozšířit výzkum nově vyvinuté metody Škálovatelné Pravděpodobnostní Aproximace (SPA) s důrazem převážně na klasifikační problémy. Metoda SPA je uplatněna k diskretizaci spojitých stochastických procesů a v kombinaci s bayesovským modelováním kauzální inference vede k vícekriteriálnímu optimalizačnímu problému, který umožňuje splnit obě kritéria současně. Řešení tohoto problému je navrženo jako algoritmus strojového učení s učitelem, který je vhodný pro různé klasifikační úlohy. Ačkoli je tento algoritmus omezen z hlediska výpočetní náročnosti, navržený odhad problému, který je úzce spjat s široce známým algoritmem K-means, je použitelný i pro velké soubory dat. Předběžné experimenty ukazují, že tuto metodiku lze přizpůsobit k vybrané aplikaci detekce koroze z obrazových dat.

Klíčová slova

analýza obrazu; detekce koroze; diskretizace; klasifikace; K-means; MATLAB; optimalizace; počítačové vidění; SPA; strojové učení; vícekriteriální optimalizace

Abstract

This thesis aims to extend the research on the newly developed Scalable Probabilistic Approximation (SPA) method, with emphasis predominantly on classification problems. The SPA method is utilized to discretize continuous stochastic processes and, in conjunction with Bayesian causal inference modeling, leads to a multiobjective optimization problem that is capable of simultaneously resolving both objectives. The solution to this problem is formulated as a supervised machine learning algorithm that is suitable for various classification tasks. Although the algorithm is limited in terms of computational cost, a proposed estimation of the problem, which is closely related to the widely known K-means algorithm, is applicable even for large datasets. Preliminary experiments demonstrate that this framework is adaptable to the selected application of corrosion detection from image data.

Keywords

classification; computer vision; corrosion detection; discretization; image analysis; K-means; machine learning; MATLAB; multiobjective optimization; optimization; SPA

Acknowledgement

First and foremost, I wish to express my sincere gratitude to my thesis supervisor, Ing. Lukáš Pospíšil, Ph.D., for his invaluable guidance and support throughout the entire bachelor thesis. His willingness to clarify all my queries and correct my errors, guiding me in the right direction, was an instrumental part of this work.

In addition, I would like to thank the same person for providing me with the opportunity to gain hands-on research experience and develop new skills that will undoubtedly benefit my future academic pursuits. I appreciate the generosity with which he shared his knowledge and expertise, allowing me to broaden my horizons and learn something new.

Last but not least, many thanks to the professors and instructors of the Department of Applied Mathematics at VSB – Technical University of Ostrava for providing me with a strong foundation in mathematics and equipping me with the skills necessary to undertake this work.

Contents

List of symbols and abbreviations

GLCM	–	Gray-level co-occurrence matrix
KKT	–	Karush-Kuhn-Tucker
KLD	–	Kullback–Leibler Divergence
KNN	–	k-nearest neighbor algorithm
MCC	–	Matthew's correlation coefficient
MOP	–	Multiobjective optimization problem
PCA	–	Principal component analysis
SML	–	Supervised machine learning
SPA	–	Scalable Probabilistic Approximation
SPG	–	Spectral Projected Gradient
SSE	–	Sum of squares error
SVM	–	Support vector machine
WSM	–	Weighted sum method
$\mathbb{N}$	–	The set of all natural numbers
$\mathbb{R}$	–	Real vector space
$\mathbb{R}^+$	–	$\{x \in \mathbb{R} \mid x > 0\}$
$\mathbb{R}_0^+$	–	$\mathbb{R}^+ \cup \{0\}$
$\mathbb{R}^n$	–	The set of all n-dimensional real vectors ($n \in \mathbb{N}$)
$\ln(x)$	–	Natural logarithm of x
$:=$	–	Defined to be equal to
$\langle \mathbf{u}, \mathbf{v} \rangle$	–	Standard scalar product defined as $\langle \mathbf{u}, \mathbf{v} \rangle := \sum_{i=1}^{n} \mathbf{u}_i \mathbf{v}_i$ ($n \in \mathbb{N}$)
$\|\mathbf{u}\|_2$	–	Euclidean norm defined as $\|\mathbf{u}\|_2 := \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}$

List of Figures

List of Tables

Chapter 1

Introduction

This thesis proposes an innovative methodology of a recently developed combination of two techniques, namely *Scalable Probabilistic Approximation* (SPA) and *Bayesian causal inference modeling*. The methodology is then applied to real-world *classification* problems, with the central goal of developing a *machine learning* model capable of accurately detecting corrosion from features extracted from image data using various *computer vision* methods. This is a significant problem in practice, as evidenced by recent studies [1, 2].

In forthcoming Chapter 2, a review of selected foundations of *probability theory* is provided, which will serve as the basis for subsequent sections. Afterwards, the formulation of a *Bayesian model* and a method for determining the solution of this model are presented.

Chapter 3 focuses on *discretization*, mainly on adaptive discretization, which is an essential step in the preprocessing of continuous data for the application of Bayesian modeling. The chapter comprises two parts, with the first one dedicated to K-means and clustering, while the second part explains the utilization of SPA for discretization purposes.

An integral part of this work is *multiobjective optimization*, which is the subject of Chapter 4. The chapter provides an overview of the underlying mathematical principles in this field and subsequently delves into the application of multiobjective optimization in combining SPA and Bayesian modeling into a single composite optimization problem.

Finally, Chapter 5, applies the developed methodology to real-world problems. In addition, the chapter explains selected key elements of machine learning, such as *feature engineering*, *model evaluation*, and *hyperparameter tuning*. The applications were implemented in MATLAB and are freely available on GitHub [3].

Chapter 2

Bayesian Causal Inference Modeling

In this chapter, we give a brief review of some of the fundamentals of probability theory that will be useful for the task at hand. Afterwards, we proceed to Bayesian modeling of causal inference, where we formulate a Bayesian linear model that will guide us throughout this work, and propose a method for finding the solution to this model.

2.1 Probability Theory

Since we discuss topics related to probability, we remind the definition of a probability space.

Definition 2.1 *A probability space is a triplet $(\Omega, \mathcal{A}, \mathcal{P})$, where Ω is a sample space, $\mathcal{A}$ is a σ-algebra, and $\mathcal{P} : \mathcal{A} \to \mathbb{R}$ is a probability function.*

An essential building block of Bayesian inference is the *Law of Total Probability*. Before delving into this significant theorem, let us first review the definitions of *conditional probability* and *finite partition*.

Definition 2.2 *Let $(\Omega, \mathcal{A}, \mathcal{P})$ be a probability space. The* conditional probability *of event $A \in \mathcal{A}$, given that event $B \in \mathcal{A}$ occurred, is defined (and denoted) by [4]*

$$\mathcal{P}(A \mid B) \coloneqq \frac{\mathcal{P}(A \cap B)}{\mathcal{P}(B)} \quad \text{if } \mathcal{P}(B) > 0.$$

Definition 2.3 *Let $(\Omega, \mathcal{A}, \mathcal{P})$ be a probability space and let $B_1, B_2, \ldots, B_n \in \mathcal{A}$ be a sequence of* incompatible *and* exhaustive *events; i.e.,*

$$(\forall i, j = 1, \ldots, n) : B_i \cap B_j = \emptyset \quad \text{if } i \neq j \quad \text{and} \quad \bigcup_{i=1}^{n} B_i = \Omega.$$

We say that the events B_i constitute a finite partition *of the sample space Ω [4].*

Theorem 2.1 *(The Law of Total Probability [4]) Let $(\Omega, \mathcal{A}, \mathcal{P})$ be a probability space, and let events $B_1, B_2, \ldots, B_n \in \mathcal{A}$ constitute a finite partition of the sample space Ω. Then, for all events $A \in \mathcal{A}$,*

$$\mathcal{P}(A) = \sum_{i=1}^{n} \mathcal{P}(A \mid B_i) \cdot \mathcal{P}(B_i),$$

if the conditional probabilities are well defined, i.e., $(\forall i = 1, \ldots, n) : \mathcal{P}(B_i) > 0$.

Proof

$$\mathcal{P}(A) = \sum_{i=1}^{n} \mathcal{P}(A \cap B_i) = \sum_{i=1}^{n} \mathcal{P}(A \mid B_i) \cdot \mathcal{P}(B_i).$$

■

In the following, we present the definition of stochastic vectors and matrices that is intended to facilitate the notation employed throughout the remainder of this work.

Definition 2.4 *A vector $\mathbf{v} \in \mathbb{R}^n$ is called* stochastic *if and only if all entries are nonnegative and sum up to one, i.e.,*

$$(\forall i = 1, \ldots, n) : \quad \mathbf{v}_i \geq 0 \quad \wedge \quad \sum_{i=1}^{n} \mathbf{v}_i = 1.$$

Afterwards, let

$$\mathbb{P}^n := \left\{ \mathbf{v} \in \mathbb{R}^n \;\middle|\; \mathbf{v}_i \geq 0 \quad \wedge \quad \sum_{i=1}^{n} \mathbf{v}_i = 1 \right\}$$

be the set of all n-dimensional stochastic vectors. A matrix $\mathsf{A} \in \mathbb{R}^{m,n}$ is said to be left stochastic *if and only if $(\forall j = 1, \ldots, n) : \mathsf{A}_{:,j} \in \mathbb{P}^m$. Analogously, we define the set of all left stochastic matrices as follows*

$$\mathbb{P}_L^{m,n} := \left\{ \mathsf{A} \in \mathbb{R}^{m,n} \;\middle|\; (\forall j = 1, \ldots, n) : \mathsf{A}_{:,j} \in \mathbb{P}^m \right\}.$$

Lemma 2.2 *Let $\mathsf{A} \in \mathbb{P}_L^{m,n}$ and $\mathsf{B} \in \mathbb{P}_L^{n,p}$ be left stochastic matrices. Then, their product AB is also a left stochastic matrix.*

Proof Firstly, given that all elements in matrices A and B are by definition nonnegative, it is evident that each element in the product AB will also be nonnegative. Secondly, we demonstrate that every column in AB sums up to one. $(\forall j = 1, \ldots, p)$:

$$\sum_{i=1}^{m} (\mathsf{AB})_{i,j} = \sum_{i=1}^{m} \sum_{k=1}^{n} \mathsf{A}_{i,k} \mathsf{B}_{k,j} = \sum_{k=1}^{n} \left(\sum_{i=1}^{m} \mathsf{A}_{i,k} \right) \mathsf{B}_{k,j} = \sum_{k=1}^{n} \mathsf{B}_{k,j} = 1$$

We have proven that every column of AB is a stochastic vector. Thus, AB is indeed a left stochastic matrix. ■

Given that we will be dealing with stochastic processes in the upcoming section, we will provide a brief overview of the fundamentals. A *stochastic process* is a system that evolves in time while

undergoing chance fluctuations. We can describe such a system by defining a family of random variables X_t, where X_t measures, at time t, the aspect of the system of interest. For example, X_t might be the number of customers in a queue at time t. As time passes, customers will arrive and leave, and so the value of X_t will change [5]. A more rigorous definition can be formulated as follows.

Definition 2.5 *A* stochastic process *(or random process) is a set* $\{X_t,\, t \in \mathcal{T}\}$ *of random variables* X_t*, where* $\mathcal{T}$ *is a subset of* $\mathbb{R}$ *[4].*

Remark 2.1 The set $\mathcal{T}$ is often referred to as the *index set*. The set of all possible values of the random variables X_t is called the *state space* of the stochastic process and is denoted as S. The deterministic variable t is often interpreted as time.

- If $\mathcal{T}$ is a countable set, then $\{X_t,\, t \in \mathcal{T}\}$ is said to be a *discrete-time* stochastic process.
- Similarly, if $\mathcal{T} \subseteq \mathbb{R}$ is an interval, then $\{X_t,\, t \in \mathcal{T}\}$ is said to be a *continuous-time* stochastic process. Usually $\mathcal{T} = [0, +\infty)$.

Definition 2.6 *If the possible values taken by the various random variables* X_t *are assumed to be at most countably infinite, so that* X_t *is a discrete random variable for any fixed value of the variable* t*, then we say that* $\{X_t,\, t \in \mathcal{T}\}$ *is a* discrete-state stochastic process *[4].*

A very important class of stochastic processes for practical applications is the class of those known as Markov processes. A *Markov process* can be interpreted as a stochastic process in which the future is independent of the past, given the present [6]. A formal definition of a Markov process follows.

Definition 2.7 *Let* $\{X_t,\, t \in \mathcal{T}\}$ *be stochastic process. If we can write that*

$$\mathcal{P}\left(X_{t_n} \leq x_n \;\middle|\; X_t, \forall t \leq t_{n-1}\right) = \mathcal{P}\left(X_{t_n} \leq x_n \;\middle|\; X_{t_{n-1}}\right), \tag{2.1}$$

where $t_{n-1} < t_n$*, we say that the stochastic process* $\{X_t,\, t \in \mathcal{T}\}$ *is a* Markov process *(or Markovian process) [4].*

The preceding Eq. (2.1), known as *Markov property* (or *memoryless property*), means that the future of the process depends only on its present state. In the case of a discrete-state stochastic process, we refer to Markov processes as *Markov chains*.

Definition 2.8 *If* $\{X_n,\, n = 0, 1, \dots\}$ *is a discrete-state stochastic process such that*

$$\mathcal{P}\left(X_{n+1} = x_{n+1} \;\middle|\; \bigcap_{i=0}^{n} X_i = x_i\right) = \mathcal{P}\left(X_{n+1} = x_{n+1} \;\middle|\; X_n = x_n\right),$$

for all states $x_0, \ldots, x_n, x_{n+1} \in \{0, 1, \ldots\}$ *and for* $n = 0, 1, \ldots$, *then* $\{X_n,\, n = 0, 1, \ldots\}$ *is called a (discrete-time) Markov chain [4].*

We could also define *continuous-time* Markov chains[1], however, in this work we suffice with discrete-time Markov chains.

2.2 Bayesian Causal Inference Modeling

Assume that we have two discrete-state stochastic processes $\mathcal{X} \coloneqq \{X_t,\, t \in \mathcal{T}\}$ and $\mathcal{Y} \coloneqq \{Y_t,\, t \in \mathcal{T}\}$, where $\mathcal{T} \subset \mathbb{N}$. Our focus is on the causal relationship between these two processes; specifically, $\mathcal{X} \to \mathcal{Y}$. The question at hand is: If we have prior information about the state of $\mathcal{X}$, can we infer the state of $\mathcal{Y}$ based on that information? In a broader sense, we are interested in investigating the interdependence of stochastic processes and whether we can infer the behavior of one process from the behavior of another.

We now propose a Bayesian linear model that establishes a causal relationship between the above stochastic processes. Let $\mathbf{s}^X = \left\{s_1^X, \ldots, s_K^X\right\}$ and $\mathbf{s}^Y = \left\{s_1^Y, \ldots, s_M^Y\right\}$ be the sets of states of $\mathcal{X}$ and $\mathcal{Y}$, respectively. Then $(\forall t \in \mathcal{T})$:

$$\boldsymbol{\pi}_t = \Lambda \boldsymbol{\gamma}_t, \tag{2.2}$$

where

$$\boldsymbol{\pi}_t = \begin{bmatrix} \mathcal{P}(Y_t = s_1^Y) \\ \vdots \\ \mathcal{P}(Y_t = s_M^Y) \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{\gamma}_t = \begin{bmatrix} \mathcal{P}(X_t = s_1^X) \\ \vdots \\ \mathcal{P}(X_t = s_K^X) \end{bmatrix}$$

are stochastic vectors with dimensions $M \in \mathbb{N}$ and $K \in \mathbb{N}$, respectively, and where

$$\Lambda = \begin{bmatrix} \mathcal{P}(Y_t = s_1^Y \mid X_t = s_1^X) & \cdots & \mathcal{P}(Y_t = s_1^Y \mid X_t = s_K^X) \\ \vdots & \ddots & \vdots \\ \mathcal{P}(Y_t = s_M^Y \mid X_t = s_1^X) & \cdots & \mathcal{P}(Y_t = s_M^Y \mid X_t = s_K^X) \end{bmatrix}$$

is a left stochastic matrix consisting of conditional probabilities. Henceforth, we will assume that these conditional probabilities are stationary (independent of the data index t) and depend only on the discrete states. Eq. (2.2) is justified by the Law of Total Probability (Theorem 2.1).

2.3 Solution of the Lambda Problem

In this section, we elaborate on the Bayesian linear model (2.2). We continue to use the notation from the previous section. Let $x_t \in \mathbf{s}^X$ be the observed categorical data (input) and let $y_t \in \mathbf{s}^Y$ be the categorical output data. A method will be presented to determine the matrix Λ from the

[1]The definition can be found, for example, in [4].

observed data pairs $\{x_t, y_t\}$. We formulate the task as a regression-like optimization problem, where we fit the model to the given data and derive the optimal parameters of the model. Therefore, it is convenient to explain the concept of a minimizer.

Definition 2.9 *We say that point* $\mathbf{x}^*$ *is a minimizer of function* $f : \Omega \to \mathbb{R}$ *on a given feasible set* Ω*, written as* $\mathbf{x}^* = \arg\min_{\mathbf{x}} \min_{\mathbf{x}\in\Omega} f(\mathbf{x})$ *if and only if all points in the feasible set have a larger or equal function value than* $f(\mathbf{x}^*)$*, i.e.,* $(\forall \mathbf{x} \in \Omega) : f(\mathbf{x}^*) \leq f(\mathbf{x})$ *[7].*

In our case, let us consider the following constrained optimization problem

$$\Lambda^* = \arg\min_{\Lambda} \min_{\Lambda\in\Omega_\Lambda} \sum_{t=1}^{T} \operatorname{dist}(\boldsymbol{\pi}_t, \Lambda\boldsymbol{\gamma}_t), \tag{2.3}$$

where dist is a distance function[2], Ω_Λ is a set of all feasible left stochastic matrices such that $\Omega_\Lambda = \mathbb{P}_L^{M,K}$, and Λ^* is a minimizer of the matrix Λ. By minimizing the distance between $\boldsymbol{\pi}$ and $\Lambda\boldsymbol{\gamma}$ we try to achieve the best possible approximation of (2.2).

In our case, the Kullback-Leibler divergence (KLD), introduced by Kullback and Leibler in [8], is chosen as the distance function. Generally, the KLD is a measure of dissimilarity between two probability distributions defined over the same sample set. Given two *discrete* probability distributions P and Q, the KLD measures how much information is lost when Q is used to approximate P. Formally, let X be a discrete random variable with values in the set $\mathcal{X} = \{1, 2, \ldots, n\}$. The KLD between two probability mass functions $p(x)$ and $q(x)$ of the random variable X is defined as follows [9]

$$D(p \parallel q) = \begin{cases} \sum_{x\in\mathcal{X}} p(x) \cdot \ln\left(\dfrac{p(x)}{q(x)}\right) & \text{if } (\forall x \in \mathcal{X}) : [q(x) = 0] \Rightarrow [p(x) = 0], \\ +\infty & \text{otherwise,} \end{cases} \tag{KLD}$$

where we use the convention of $0 \cdot \ln 0 = 0$ and $0 \cdot \ln \frac{0}{0} = 0$. The KLD is always non-negative and is zero if and only if $p = q$. The KLD does not qualify as a metric because it does not satisfy symmetry (i.e., $D(p \parallel q) \neq D(q \parallel p)$ in general) nor the triangle inequality [9, 10]. In addition, the

[2]The distance function dist is not necessarily a metric. In the following text, a statistical distance (KLD) is used.

KLD can be expressed in an equivalent form using logarithmic identities[3]:

$$\begin{aligned} D(p \parallel q) &= \sum_{x \in \mathcal{X}} p(x) \cdot \ln\left(\frac{p(x)}{q(x)}\right) \\ &= \sum_{x \in \mathcal{X}} p(x) \cdot \big(\ln p(x) - \ln q(x)\big) \\ &= -\sum_{x \in \mathcal{X}} p(x) \cdot \big(\ln q(x) - \ln p(x)\big) \\ &= -\sum_{x \in \mathcal{X}} p(x) \cdot \ln\left(\frac{q(x)}{p(x)}\right). \end{aligned}$$

Moreover, KLD can be formulated for stochastic vectors $\mathbf{u} \in \mathbb{P}^n$, $\mathbf{v} \in \mathbb{P}^n$ as follows:

$$D(\mathbf{u} \parallel \mathbf{v}) = -\sum_{i=1}^{n} \mathbf{u}_i \cdot \ln\left(\frac{\mathbf{v}_i}{\mathbf{u}_i}\right). \tag{2.4}$$

In the case of our problem, the KLD will have the following form

$$D(\boldsymbol{\pi}_t \parallel \Lambda\boldsymbol{\gamma}_t) = -\sum_{m=1}^{M} \boldsymbol{\pi}_{m,t} \cdot \ln \frac{\sum\limits_{k=1}^{K} \Lambda_{m,k} \cdot \boldsymbol{\gamma}_{k,t}}{\boldsymbol{\pi}_{m,t}},$$

where $\boldsymbol{\pi}_{m,t}$ denotes the m-th component of the vector $\boldsymbol{\pi}_t$, and correspondingly, the k-th element of $\boldsymbol{\gamma}_t$ is represented by $\boldsymbol{\gamma}_{k,t}$. The KLD transforms problem (2.3) into

$$\Lambda^* = \arg\min_{\Lambda \in \Omega_\Lambda} \underbrace{-\sum_{t=1}^{T}\sum_{m=1}^{M} \boldsymbol{\pi}_{m,t} \cdot \ln \frac{\sum\limits_{k=1}^{K} \Lambda_{m,k} \cdot \boldsymbol{\gamma}_{k,t}}{\boldsymbol{\pi}_{m,t}}}_{=:\,\varphi(\Lambda),\,\varphi\colon \mathbb{R}^{M,K} \to \mathbb{R}}. \tag{2.5}$$

Using the well-known logarithmic identity[3] we obtain the following

$$\varphi(\Lambda) = \underbrace{-\sum_{t=1}^{T}\sum_{m=1}^{M} \boldsymbol{\pi}_{m,t} \cdot \ln\left(\sum_{k=1}^{K} \Lambda_{m,k} \cdot \boldsymbol{\gamma}_{k,t}\right)}_{=:\,\psi(\Lambda),\,\psi\colon \mathbb{R}^{M,K} \to \mathbb{R}} + \underbrace{\sum_{t=1}^{T}\sum_{m=1}^{M} \boldsymbol{\pi}_{m,t} \cdot \ln \boldsymbol{\pi}_{m,t}}_{=:\,\xi \in \mathbb{R}}. \tag{2.6}$$

In the following lemma, we clarify that the additive constant term ξ can be omitted in the optimization process.

[3] $(\forall x, y \in \mathbb{R}^+)\colon \ln \dfrac{x}{y} = \ln x - \ln y$

Lemma 2.3 *Let $f\colon \Omega \to \mathbb{R}$ be a function on a given feasible set Ω. Then, $(\forall \alpha \in \mathbb{R})$:*

$$\arg\min_{\mathbf{x}\ \mathbf{x}\in\Omega} f(\mathbf{x}) = \arg\min_{\mathbf{x}\ \mathbf{x}\in\Omega} \left(f(\mathbf{x}) + \alpha\right) \tag{2.7}$$

whenever one side of the equality (2.7) *is meaningful.*

Proof Recall Definition 2.9. The problem $\arg\min_{\mathbf{x}\ \mathbf{x}\in\Omega} f(\mathbf{x})$ on the left hand side of (2.7) is equivalent to the problem find $\mathbf{x}^* \in \Omega$ such that $(\forall \mathbf{x} \in \Omega) : f(\mathbf{x}^*) \leq f(\mathbf{x})$. Without loss of generality, we can add a constant term $\alpha \in \mathbb{R}$ to both sides of the inequality to obtain $f(\mathbf{x}^*) + \alpha \leq f(\mathbf{x}) + \alpha$, which is the definition of a minimizer for the problem $\arg\min_{\mathbf{x}\ \mathbf{x}\in\Omega} \left(f(\mathbf{x}) + \alpha\right)$ on the right hand side of (2.7). Thus, equality (2.7) holds. ∎

Since the function $-\ln(x)$ is convex on its entire domain $(\forall x \in \mathbb{R}^+)$, the problem can be further estimated using the Jensen's inequality proposed by Jensen in 1906 [11]. The Jensen's inequality can be understood as a generalization of the definition of a convex function; see Figure 2.1 for a demonstration. In the following, we remind this definition.

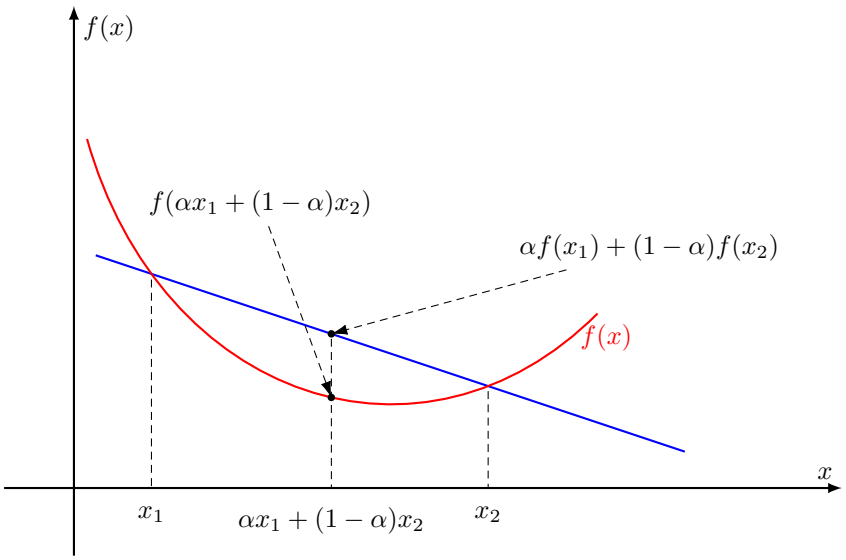

Figure 2.1: Convex function.

Definition 2.10 *A real function f is said to be* convex *if and only if*

$$(\forall x_1, x_2 \in D_f)\,(\forall \alpha \in [0,1]) : f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2),$$

where D_f is the domain of the function f.

The Jensen's inequality generalizes the statement that the secant line of a convex function lies above the graph of the function, which is the Jensen's inequality for two points [12].

Theorem 2.4 *(Jensen's inequality.) Let f be a real convex function defined in $D_f \subset \mathbb{R}$ and let $(\forall i = 1, \ldots, n) : x_i \in D_f$, $\alpha_i \in [0, 1]$ and $\sum_{i=1}^{n} \alpha_i = 1$ $(n \in \mathbb{N})$. Then*

$$f\left(\sum_{i=1}^{n} \alpha_i x_i\right) \leq \sum_{i=1}^{n} \alpha_i f(x_i). \tag{2.8}$$

Proof (By induction.) For the trivial case $n = 1$ the statement holds. The case for $n = 2$ is true by the definition of a convex function. Assume that the statement holds for a certain $k \in \{2, \ldots, n\}$. It follows that

$$\begin{aligned}
f\left(\sum_{i=1}^{k+1} \alpha_i x_i\right) &= f\left(\alpha_{k+1} x_{k+1} + \sum_{i=1}^{k} \alpha_i x_i\right) = f\left(\alpha_{k+1} x_{k+1} + (1 - \alpha_{k+1}) \frac{1}{1 - \alpha_{k+1}} \sum_{i=1}^{k} \alpha_i x_i\right) \\
&\leq \alpha_{k+1} f(x_{k+1}) + (1 - \alpha_{k+1}) f\left(\frac{1}{1 - \alpha_{k+1}} \sum_{i=1}^{k} \alpha_i x_i\right) \\
&= \alpha_{k+1} f(x_{k+1}) + (1 - \alpha_{k+1}) f\left(\sum_{i=1}^{k} \frac{\alpha_i}{1 - \alpha_{k+1}} x_i\right) \\
&\leq \alpha_{k+1} f(x_{k+1}) + (1 - \alpha_{k+1}) \sum_{i=1}^{k} \frac{\alpha_i}{1 - \alpha_{k+1}} f(x_i) \\
&= \alpha_{k+1} f(x_{k+1}) + \sum_{i=1}^{k} \alpha_i f(x_i) \\
&= \sum_{i=1}^{k+1} \alpha_i f(x_i).
\end{aligned}$$

We have demonstrated that the assumption implies that the proposition is valid for $k + 1$. Thus, by the principle of Mathematical Induction the inequality is true for all natural numbers [13]. ∎

In our case, the Jensen's inequality can be applied to Eq. (2.6) to obtain

$$\psi(\Lambda) = -\sum_{t=1}^{T} \sum_{m=1}^{M} \boldsymbol{\pi}_{m,t} \cdot \ln\left(\sum_{k=1}^{K} \Lambda_{m,k} \boldsymbol{\gamma}_{k,t}\right) \leq \underbrace{-\sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{k=1}^{K} \boldsymbol{\pi}_{m,t} \boldsymbol{\gamma}_{k,t} \ln \Lambda_{m,k}}_{=: \hat{\psi}(\Lambda),\ \hat{\psi} : \mathbb{R}^{M,K} \to \mathbb{R}}.$$

The goal is to minimize the estimation $\hat{\psi}$. We will show that such an estimation results in a better problem in terms of the solvability. In this case, the problem has an analytical solution for Λ. In

the following, we provide the derivation. Since the given optimization problem is constrained, we briefly review the theory of Lagrange function and the corresponding optimality conditions.

Lemma 2.5 *(About Lagrange Function and KKT Conditions [7, 14].)*
Let us consider an optimization problem

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \min_{\mathbf{x}\in\Omega} f(\mathbf{x}),$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function and the feasible set $\Omega \subset \mathbb{R}^n$ is convex (i.e., $(\forall \mathbf{x}_1, \mathbf{x}_2 \in \Omega)(\forall \alpha \in [0,1]) : \alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 \in \Omega$), described by equality and inequality constraints

$$\Omega := \left\{ \mathbf{x} \in \mathbb{R}^n : \begin{matrix} h_{Ei}(\mathbf{x}) = 0 & i=1,\ldots,m_E \\ h_{Ij}(\mathbf{x}) \leq 0 & j=1,\ldots,m_I \end{matrix} \right\} \neq \emptyset,$$

where

- *$h_{Ei} : \mathbb{R}^n \to \mathbb{R}$ are linear functions describing equality constraints,*
- *$h_{Ij} : \mathbb{R}^n \to \mathbb{R}$ are convex functions describing inequality constraints.*

Suppose $\mathbf{x}^*$ solves this problem. Then, there exist vectors $\boldsymbol{\lambda}_E \in \mathbb{R}^{m_E}$ and $\boldsymbol{\lambda}_I \in \mathbb{R}^{m_I}$, $\boldsymbol{\lambda}_I \geq 0$, such that $\mathbf{x}^*$ solves the Lagrangian problem

$$\mathbf{x}^* := \arg\min_{\mathbf{x}} \mathcal{L}\left(\mathbf{x}, \boldsymbol{\lambda}_E, \boldsymbol{\lambda}_I\right),$$

where $\mathcal{L} : \mathbb{R}^{m_E} \times \mathbb{R}^{m_I} \to \mathbb{R}$ is Lagrangian function defined by

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}_E, \boldsymbol{\lambda}_I) := f(\mathbf{x}) + \sum_{i=1}^{m_E} \boldsymbol{\lambda}_{Ei} h_{Ei}(\mathbf{x}) + \sum_{j=1}^{m_I} \boldsymbol{\lambda}_{Ij} h_{Ij}(\mathbf{x}).$$

The appropriate optimality conditions of this problem, so-called Karush-Kuhn-Tucker (KKT) conditions, are given by

$$\begin{aligned}
\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}_E, \boldsymbol{\lambda}_I) &= \nabla f(\mathbf{x}) + \sum_{i=1}^{m_E} \boldsymbol{\lambda}_{Ei} \nabla h_{Ei}(\mathbf{x}) + \sum_{j=1}^{m_I} \boldsymbol{\lambda}_{Ij} \nabla h_{Ij}(\mathbf{x}) && = 0 \\
\nabla_{\boldsymbol{\lambda}_E}\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}_E, \boldsymbol{\lambda}_I) &= [h_{E1}(\mathbf{x}), \ldots, h_{Em_E}(\mathbf{x})]^T && = 0 \\
\nabla_{\boldsymbol{\lambda}_I}\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}_E, \boldsymbol{\lambda}_I) &= [h_{I1}(\mathbf{x}), \ldots, h_{Im_I}(\mathbf{x})]^T && \leq 0 \\
& \boldsymbol{\lambda}_I && \geq 0 \\
& \boldsymbol{\lambda}_{Ij} h_{Ij}(\mathbf{x}) && = 0, \quad j = 1, \ldots, m_I.
\end{aligned}$$

Proof See Luenberger[15], Bertsekas[16], Nocedal and Wright[17] or Boyd and Vandenberghe[7]. ∎

In the case of our problem, we will show that the inequality constraints on the feasible set Ω_{Λ} in the optimization problem (2.3) can be ignored, i.e., we do not have to consider the constraint that each element of Λ must be in the interval $[0, 1]$, since the equality constraint that each column of Λ sums up to one will be sufficient. Let us define a Lagrangian function $\mathcal{L}(\Lambda, \boldsymbol{\lambda})$ to minimize Λ such that

$$\mathcal{L}(\Lambda, \boldsymbol{\lambda}) = \hat{\psi}(\Lambda) + \sum_{k=1}^{K} \boldsymbol{\lambda}_k h_k(\Lambda_{:,k}), \tag{2.9}$$

where $(\forall k = 1, \ldots, K) : h_k : \mathbb{R}^M \to \mathbb{R}$ are the linear equality constraint functions that represent that each column of Λ sums up to one, that is, $\sum_{m=1}^{M} \Lambda_{m,k} = 1$. It follows that $h_k(\Lambda_{:,k}) = \sum_{m=1}^{M} \Lambda_{m,k} - 1$. By substitution of functions h_k and $\hat{\psi}$ to (2.9) we obtain

$$\mathcal{L}(\Lambda, \boldsymbol{\lambda}) = -\sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{k=1}^{K} \boldsymbol{\pi}_{m,t} \boldsymbol{\gamma}_{k,t} \ln \Lambda_{m,k} + \sum_{k=1}^{K} \boldsymbol{\lambda}_k \left(\sum_{m=1}^{M} \Lambda_{m,k} - 1 \right).$$

We can find the stationary points of $\mathcal{L}$ using KKT conditions (Lemma 2.5):

$$(\forall \hat{m}, \hat{k}) : \frac{\partial \mathcal{L}}{\partial \Lambda_{\hat{m},\hat{k}}} = -\sum_{t=1}^{T} \boldsymbol{\pi}_{\hat{m},t} \boldsymbol{\gamma}_{\hat{k},t} \frac{1}{\Lambda_{\hat{m},\hat{k}}} + \boldsymbol{\lambda}_{\hat{k}} = 0 \tag{2.10}$$

$$(\forall \hat{k}) : \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}_{\hat{k}}} = \sum_{m=1}^{M} \Lambda_{m,\hat{k}} - 1 = 0 \tag{2.11}$$

from (2.10),

$$\begin{aligned} \boldsymbol{\lambda}_{\hat{k}} &= \frac{1}{\Lambda_{\hat{m},\hat{k}}} \sum_{t=1}^{T} \boldsymbol{\pi}_{\hat{m},t} \boldsymbol{\gamma}_{\hat{k},t} \\ \Lambda_{\hat{m},\hat{k}} &= \frac{1}{\boldsymbol{\lambda}_{\hat{k}}} \sum_{t=1}^{T} \boldsymbol{\pi}_{\hat{m},t} \boldsymbol{\gamma}_{\hat{k},t} \end{aligned} \tag{2.12}$$

from (2.11),

$$\begin{aligned} \frac{1}{\boldsymbol{\lambda}_{\hat{k}}} \sum_{t=1}^{T} \underbrace{\left(\sum_{m=1}^{M} \boldsymbol{\pi}_{m,t} \right)}_{=1 \ (\forall t)} \boldsymbol{\gamma}_{\hat{k},t} - 1 &= 0 \\ \sum_{t=1}^{T} \boldsymbol{\gamma}_{\hat{k},t} &= \boldsymbol{\lambda}_{\hat{k}}. \end{aligned} \tag{2.13}$$

Substituting (2.13) into (2.12) we obtain the following

$$\Lambda_{\hat{m},\hat{k}} = \frac{\sum_{t=1}^{T} \boldsymbol{\pi}_{\hat{m},t} \boldsymbol{\gamma}_{\hat{k},t}}{\sum_{t=1}^{T} \boldsymbol{\gamma}_{\hat{k},t}} \in [0,1]\,. \tag{2.14}$$

We have shown that every element of Λ is indeed in the interval $[0,1]$ because the inequality

$$\sum_{t=1}^{T} \boldsymbol{\pi}_{\hat{m},t} \boldsymbol{\gamma}_{\hat{k},t} \leq \sum_{t=1}^{T} \boldsymbol{\gamma}_{\hat{k},t}$$

holds. Thus, the assumption of ignoring the inequality constraints of the feasible set Ω_{Λ} is justified. If the denominator of (2.14) is equal to zero, then there is no data point that can provide information to compute the conditional probabilities between the $\tilde{k}$-th state of input x_t and the state of output y_t. The corresponding state is not present in the given data. From the minimization point of view, the terms corresponding to these zero entries $\boldsymbol{\gamma}_{\tilde{k},t}$, $\forall t = 1,\ldots,T$, are not present in the objective function (that is, all are equal to zero); therefore, any choice of $\Lambda_{:,\tilde{k}}$ does not influence the value of the objective function. Consequently, any choice of these values is the solution. If this situation occurs, then we use the uniform distribution,

$$(\forall m = 1,\ldots,M):\ \Lambda_{m,\tilde{k}} = \frac{1}{M},$$

i.e., the conditional probabilities are equally distributed between states. This choice reflects the lack of information on the causality between this state and $\boldsymbol{\pi}_t$.

Remark 2.2 Instead of the Bayesian model (2.2) we may employ the following discrete-time Markov chain

$$(\forall t \in T):\ \boldsymbol{\pi}_{t+1} = \Lambda \boldsymbol{\pi}_t, \tag{2.15}$$

where $T \subset \mathbb{N} \cup \{0\}$, $\boldsymbol{\pi}_t \in \mathbb{P}^K$ is a stochastic vector, and $\Lambda \in \mathbb{P}_L^{K,K}$ is a left stochastic matrix that encompasses conditional probabilities of transition between states. Following the identical steps shown in this section, one can straightforwardly derive a solution for the Markov chain (2.15) that is analogous to the solution of the Bayesian model (2.2). The solution is given below:

$$(\forall k_1, k_2 = 1,\ldots,K):\ \Lambda_{k_1,k_2} = \frac{\sum_{t=0}^{T-1} \boldsymbol{\pi}_{k_1,t+1} \boldsymbol{\pi}_{k_2,t}}{\sum_{t=0}^{T-1} \boldsymbol{\pi}_{k_2,t}}.$$

Chapter 3

Discretization

The Bayesian model presented in the previous section represents the exact relationship between two categorical processes under the assumption of a stationary transition matrix Λ. However, in practical applications, we usually deal with continuous stochastic processes. To apply the (discrete) Bayesian model, we first need to discretize the given processes, and we accomplish this by utilizing the provided data.

One of the most commonly used approaches for discretization is introducing fixed *boxes* (also known as data *binning* or data *bucketing*), where the range of values for a continuous variable is divided into a grid (e.g., equidistant) and categorized based on the corresponding intervals. A typical example of this is a histogram. Although this approach is straightforward, it suffers from an exponential growth in the number of boxes as the dimensionality of the problem increases.

In our methodology, we utilize *adaptive discretization*, where the number of boxes is independent of the dimension of the given data. Specifically, we adopt K-means.

3.1 *K*-means

K-means is widely recognized as an important *clustering technique* that belongs to the field of *unsupervised learning*. The fundamental objective of clustering is to assign labels to data points by leveraging an unlabeled dataset [18]. A loose definition of clustering could be "the process of organizing objects into *groups* whose members are similar in some way." Each cluster is then characterized by the common features of the entities it contains; the vector of these common features is called a *centroid*. Usually, the objects in a cluster are "more similar" to each other and "less similar" to the objects of the other clusters [19].

The basic idea is the following. First, one has to choose $K \in \mathbb{N}$ – the number of clusters. Then an initial clustering is required. This can be achieved by randomly assigning K feature vectors to be the initial centroids or, for example, by means of the K-means++ initialization [20]. Afterwards, we compute the distance from each data point to each centroid using a metric, such

as the Euclidean distance. Consequently, we assign each data point to the closest centroid (like if we labeled each data point with a centroid id as the label). For each centroid, we calculate the average feature vector of the data points labeled with it. These average feature vectors become the new locations of the centroids. We recompute the distance from each data point to each centroid, modify the assignment, and repeat the procedure until the assignments do not change after the centroid locations were recomputed [18], or other convergence criteria (e.g., a predefined number of iterations[1]) are satisfied. The above procedure with random centroid initialization is sometimes referred to as Lloyd's K-means algorithm [20].

3.1.1 Prerequisite for *K*-means

In this section, we derive an analytical solution for the trivial case of K-means, where $K = 1$. Let $\mathsf{X} \in \mathbb{R}^{D,T}$ represents the given data from a finite-dimensional data space. We denote $\mathcal{X} \coloneqq \{\mathbf{x}_t \mid (\forall t = 1, \ldots, T) : \mathbf{x}_t \coloneqq \mathsf{X}_{:,t}\}$ as the set of feature vectors (or data points). The goal is to find the optimal centroid $\mathbf{c}^* \in \mathbb{R}^D$ for the set $\mathcal{X}$, that is, a centroid, such that the sum of distances from the centroid to all feature vectors of $\mathcal{X}$ is minimal. The formulation of the corresponding optimization problem is is given by

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \sum_{t=1}^{T} \|\mathbf{x}_t - \mathbf{c}\|_2^2 , \tag{3.1}$$

where $\|\cdot\|_2^2$ denotes the Euclidean distance squared. The squared Euclidean distance is chosen because of its more favorable properties. In this case, the optimization problem (3.1) has an analytical solution, and we now provide its derivation. We define the objective function L as follows

$$L(\mathbf{c}) \coloneqq \sum_{t=1}^{T} \|\mathbf{x}_t - \mathbf{c}\|_2^2 = \sum_{t=1}^{T} \langle \mathbf{x}_t - \mathbf{c}, \mathbf{x}_t - \mathbf{c} \rangle = \sum_{t=1}^{T} \left(\mathbf{x}_t^T \mathbf{x}_t - 2\mathbf{x}_t^T \mathbf{c} + \mathbf{c}^T \mathsf{I} \mathbf{c} \right),$$

where $\mathsf{I} \in \mathbb{R}^{D,D}$ is an identity matrix such that

$$(\forall i, j = 1, \ldots, D \mid i \neq j) : \mathsf{I}_{i,j} = 0 \quad \text{and} \quad (\forall i, j = 1, \ldots, D \mid i = j) : \mathsf{I}_{i,j} = 1.$$

[1] It is advisable to use the maximum number of iterations as a safeguard, since the convergence of the algorithm cannot be guaranteed.

From *necessary condition for local minimum*, we can conclude that

$$\begin{aligned}\nabla_{\mathbf{c}} L(\mathbf{c}) = \sum_{t=1}^{T} (-2\mathbf{x}_t + 2\mathbf{c}) &= 0 \\ \sum_{t=1}^{T} 2\mathbf{x}_t &= \sum_{t=1}^{T} 2\mathbf{c} \\ T\mathbf{c} &= \sum_{t=1}^{T} \mathbf{x}_t \\ \mathbf{c} &= \frac{1}{T} \sum_{t=1}^{T} \mathbf{x}_t. \end{aligned} \tag{3.2}$$

Additionally, the corresponding Hessian matrix of function L is given by

$$\mathsf{H} = 2\sum_{t=1}^{T} \mathsf{I} = 2T\mathsf{I} = 2 \begin{bmatrix} T & 0 & \cdots & 0 \\ 0 & T & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & T \end{bmatrix},$$

where $\mathsf{I} \in \mathbb{R}^{D,D}$ is an identity matrix. Since the Hessian matrix H is *symmetric positive definite*, the *sufficient condition for local minimum* and the *quadratic* nature of function L with respect to the variable $\mathbf{c}$ imply that (3.2) represents the *global minimum* of L. Thus, (3.2) is the solution to the optimization problem (3.1). Unsurprisingly, the result (3.2) is the mean value of all data points in the set $\mathcal{X}$.

3.1.2 *K*-means as Optimization Problem

The aim of this section is to derive the Lloyd's K-means algorithm. We use the same notation as in preceding Section 3.1.1. The goal is to find the optimal centroids $\mathbf{c}_k^* \in \mathbb{R}^D$, $k = 1, \ldots, K$, for the set $\mathcal{X}$, that is, centroids such that the sum of distances from the centroid to all points affiliated with the given centroid is minimal. The data point $\mathbf{x}_t$ is said to be affiliated with the centroid $\mathbf{c}_k^*$, if and only if $\mathbf{c}_k^*$ minimizes the distance from $\mathbf{x}_t$ to all centroids $\mathbf{c}_1, \ldots, \mathbf{c}_K$. We need to find a way to determine whether a point is affiliated with a centroid. Motivated by the set indicator function (characteristic function)

$$\Omega \subset \mathbb{R}^n, \ \chi(\mathbf{x}) \coloneqq \begin{cases} 1, \text{ if } \mathbf{x} \in \Omega, \\ 0, \text{ if } \mathbf{x} \notin \Omega, \end{cases}$$

we define *cluster indicator function* $\gamma\colon \{1,\dots,K\}\times\{1,\dots,T\}\to\{0,1\}$ as follows

$$\gamma(k,t) \coloneqq \begin{cases} 1, \text{ if } \mathbf{x}_t \text{ is affiliated to cluster } \mathbf{c}_k^*, \\ 0, \text{ elsewhere.} \end{cases}$$

Moreover, we construct the matrix $\mathsf{\Gamma} \in \mathbb{P}_L^{K,T}$ so that $(\forall k = 1,\dots,K)(\forall t = 1,\dots,T)$:

$$\mathsf{\Gamma}_{k,t} \coloneqq \gamma(k,t).$$

One data point $\mathbf{x}_t$ can be associated with exactly one centroid $\mathbf{c}_k$ at a time. In other words, each data point in $\mathcal{X}$ is exactly in one cluster. It follows that $\mathsf{\Gamma}$ is a left stochastic matrix. Finally, we can describe K-means as an optimization problem given by

$$[\mathsf{C}^*, \mathsf{\Gamma}^*] = \underset{\mathsf{C},\mathsf{\Gamma}}{\arg}\ \underset{\mathsf{\Gamma}\in\Omega_{\mathsf{\Gamma}}}{\min} L_{kmeans}, \tag{3.3}$$

where $\mathsf{C} \coloneqq \begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_K \end{bmatrix} \in \mathbb{R}^{D,K}$, $\Omega_{\mathsf{\Gamma}} = \mathbb{P}_L^{K,T}$ is the set of all feasible left stochastic matrices $\mathsf{\Gamma}$, and

$$L_{kmeans}(\mathsf{C},\mathsf{\Gamma}) \coloneqq \sum_{t=1}^{T}\sum_{k=1}^{K} \mathsf{\Gamma}_{k,t} \left\|\mathbf{x}_t - \mathbf{c}_k\right\|_2^2. \tag{3.4}$$

The derivation of the analytical solution for the centroids C^* is analogous to Section 3.1.1. Since the problem (3.3) is separable in variable k, we can obtain the optimal centroid for each k as follows

$$(\forall k = 1\dots,K):\ \mathbf{c}_k^* = \underset{\mathbf{c}_k}{\arg}\ \min \sum_{t=1}^{T} \mathsf{\Gamma}_{k,t} \left\|\mathbf{x}_t - \mathbf{c}_k\right\|_2^2$$

$$L(\mathbf{c}_k) \coloneqq \sum_{t=1}^{T} \mathsf{\Gamma}_{k,t} \left\|\mathbf{x}_t - \mathbf{c}_k\right\|_2^2 = \sum_{t=1}^{T} \mathsf{\Gamma}_{k,t} \left\langle \mathbf{x}_t - \mathbf{c}_k, \mathbf{x}_t - \mathbf{c}_k \right\rangle = \sum_{t=1}^{T} \mathsf{\Gamma}_{k,t} \left(\mathbf{x}_t^T\mathbf{x}_t - 2\mathbf{x}_t^T\mathbf{c}_k + \mathbf{c}_k^T\mathsf{I}\mathbf{c}_k \right)$$

$$\begin{aligned} \nabla_{\mathbf{c}_k} L(\mathbf{c}_k) = \sum_{t=1}^{T} \mathsf{\Gamma}_{k,t} \left(-2\mathbf{x}_t + 2\mathbf{c}_k\right) = 0 \qquad & \textit{(Necessary condition for local minimum.)} \\ -\sum_{t=1}^{T} \mathsf{\Gamma}_{k,t}\mathbf{x}_t + \mathbf{c}_k \sum_{t=1}^{T} \mathsf{\Gamma}_{k,t} = 0 & \\ \mathbf{c}_k \sum_{t=1}^{T} \mathsf{\Gamma}_{k,t} = \sum_{t=1}^{T} \mathsf{\Gamma}_{k,t}\mathbf{x}_t & \qquad (3.5) \\ \mathbf{c}_k^* = \frac{\sum_{t=1}^{T} \mathsf{\Gamma}_{k,t}\mathbf{x}_t}{\sum_{t=1}^{T} \mathsf{\Gamma}_{k,t}} & \qquad (3.6) \end{aligned}$$

Notice that the denominator in (3.6), $\sum_{t=1}^{T} \Gamma_{k,t}$, is equivalent to the number of points affiliated to the k-th centroid $\mathbf{c}_k$, for brevity, we denote this number as N_k. The corresponding Hessian associated with the function L is given by

$$\mathsf{H} = 2N_k \mathsf{I} = 2 \begin{bmatrix} N_k & 0 & \cdots & 0 \\ 0 & N_k & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & N_k \end{bmatrix},$$

where $\mathsf{I} \in \mathbb{R}^{D,D}$ is an identity matrix. The Hessian H is *symmetric positive definite* if $N_k > 0$. Additionally, the function L is quadratic in variable $\mathbf{c}_k$, and the equation (3.6) is well defined for any non-empty cluster. Thus, (3.6) is a minimum of the objective function L. However, if there is an empty cluster, then (3.5) has an indefinite number of solutions. This is because the objective function does not incorporate the corresponding variable $\mathbf{c}_k$, and, as a result, the problem is solved for any value. This situation arises when the centroid of the empty cluster cannot be determined unambiguously. Under the assumption that there is no empty cluster, the result (3.6) represents the mean value of all points affiliated with the cluster $\mathbf{c}_k$.

The solution to the Γ subproblem is straightforward. For each data point $\mathbf{x}_t$ we find the distances to all centroids and choose the centroid that is closest to the given data point. Mathematically, we can write that

$$(\forall t = 1, \ldots, T) : \Gamma^*_{:,t} = \underset{\Gamma_{:,t}}{\arg} \min_{\Gamma_{:,t} \in \mathbb{P}^K} \sum_{k=1}^{K} \Gamma_{k,t} \left\| \mathbf{x}_t - \mathbf{c}_k \right\|_2^2, \tag{3.7}$$

$$(\forall k = 1, \ldots, K) : \Gamma^*_{k,t} = \begin{cases} 1, \text{ if } k^* = \underset{k}{\arg} \min \left\| \mathbf{x}_t - \mathbf{c}_k \right\|_2^2, \\ 0, \text{ elsewhere.} \end{cases} \tag{3.8}$$

The K-means algorithm can be solved as a sequence of split optimization problems. We alternately update the matrix Γ^* (with fixed C) and find minimizers for the centroids C^* (with fixed Γ).

Algorithm 1: K-means

Set initial approximation of centroids C_0.

$i = 0$

while *"convergence criteria"* **do**

$\Gamma_{i+1} = \underset{\Gamma}{\arg} \min_{\Gamma \in \Omega_\Gamma} L(\mathsf{C}_i, \Gamma)$ $\longrightarrow$ Reassign each data point to its nearest centroid (3.8).

$\mathsf{C}_{i+1} = \underset{\mathsf{C}}{\arg} \min L(\mathsf{C}, \Gamma_{i+1})$ $\longrightarrow$ Compute centroid of each cluster (3.6).

$i = i + 1$

end

In each step of Algorithm 1 the objective function improves (or remains the same when the algorithm

converges), since the solution found in the previous step is in the search space of the current step. However, since we fix some of the variables in each step, this is a local search procedure that does not guarantee optimality [21].

A simple MATLAB implementation of Algorithm 1 (Listing A.1) and a practical example of K-means++ initialization (Fig. A.1), as well as the resulting clustering (Fig. A.2) and a widely used heuristic for determining the ideal number of clusters (Fig. A.3), can be found in Appendix A.

3.2 Scalable Probabilistic Approximation

In this chapter, we introduce the *Scalable Probabilistic Approximation* (SPA) method [22]. Later in this work, we will demonstrate that SPA and K-means are closely interconnected, not only because both can accomplish data discretization into clusters.

Let $\mathsf{X} \in \mathbb{R}^{D,T}$ represents the given data, and let $\Omega_{\mathsf{\Gamma}} = \mathbb{P}_L^{K,T}$ be the feasible set of all left stochastic matrices $\mathsf{\Gamma}$, where $\mathsf{\Gamma}$ is composed of stochastic column vectors $\boldsymbol{\gamma}_t \in \mathbb{P}^K$. Finally, let the matrix $\mathsf{S} \in \mathbb{R}^{D,K}$, defined as $\mathsf{S} \coloneqq \begin{bmatrix}\mathbf{s}_1 & \mathbf{s}_2 & \dots & \mathbf{s}_K\end{bmatrix}$, encompasses all possible discrete states (clusters). The number of discrete states is denoted as $K > 1$, $T \in \mathbb{N}$ represents the number of data points (feature vectors) $\mathbf{x}_t \coloneqq \mathsf{X}_{:,t}$, and $D \in \mathbb{N}$ stands for the dimension (number of features) of each data point. The general formulation of the SPA optimization problem in *Euclidean space* is given by

$$[\mathsf{S}^*, \mathsf{\Gamma}^*] \coloneqq \underset{\mathsf{S},\mathsf{\Gamma}}{\arg\min} \underset{\mathsf{\Gamma}\in\Omega_{\mathsf{\Gamma}}}{} L(\mathsf{S}, \mathsf{\Gamma}), \tag{3.9}$$

where L is an objective function defined as

$$L(\mathsf{S}, \mathsf{\Gamma}) \coloneqq \sum_{t=1}^{T} \left\| \mathbf{x}_t - \sum_{k=1}^{K} \mathsf{\Gamma}_{k,t}\mathbf{s}_k \right\|_2^2 + \varepsilon_{\mathsf{S}}^2 \Phi_{\mathsf{S}}(\mathsf{S}) + \varepsilon_{\mathsf{\Gamma}}^2 \Phi_{\mathsf{\Gamma}}(\mathsf{\Gamma}). \tag{3.10}$$

In the case of SPA, $\mathsf{\Gamma}$ encompasses unknown cluster affiliation stochastic vectors and S contains unknown mapping parameters between the data X and the probability representation $\mathsf{\Gamma}$. The possibility of Tikhonov-based regularization [23] of the original ill-posed problem[2] using regularization functions $\Phi_{\mathsf{S}}(\mathsf{S})$, $\Phi_{\mathsf{\Gamma}}(\mathsf{\Gamma})$ with the corresponding regularization parameters $\varepsilon_{\mathsf{S}}^2, \varepsilon_{\mathsf{\Gamma}}^2 > 0$ is included. We can consider a linear relationship between a data point $\mathbf{x}_t$ and its corresponding probabilistic representation $\boldsymbol{\gamma}_t$ such that

$$\mathbf{x}_t = \sum_{k=1}^{K} \mathsf{\Gamma}_{k,t}\mathbf{s}_k = \mathsf{S}\boldsymbol{\gamma}_t, \tag{3.11}$$

where $\mathsf{S} \in \mathbb{R}^{D,K}$ is a matrix of parameters of this mapping [22]. Equation (3.11) can be interpreted as the *expected value* of the data representation vectors $\mathbf{s}_k$ with the probability density vector $\boldsymbol{\gamma}_t$.

[2]The solution to the problem (3.9) without the regularization is not unique [22]. Therefore, it is an ill-posed problem. Tikhonov-based regularization is introduced to transform the original ill-posed problem into a well-posed problem.

The problem (3.9) can be solved iteratively as a sequence of split optimization problems. Assume that a feasible initial approximation S_0 is given. Each iteration i involves solving the subproblem for $\mathsf{\Gamma}_{i+1}$ with fixed S_i, followed by the subproblem for S_{i+1} with fixed $\mathsf{\Gamma}_{i+1}$. This procedure is summarized in the following pseudocode.

Algorithm 2: SPA

Set initial approximation S_0.

$i = 0$

while *"convergence criteria"* **do**

$\mathsf{\Gamma}_{i+1} = \underset{\mathsf{\Gamma}}{\arg} \min_{\mathsf{\Gamma} \in \Omega_{\mathsf{\Gamma}}} L(\mathsf{S}_i, \mathsf{\Gamma}),$ where S_i is fixed.

$\mathsf{S}_{i+1} = \underset{\mathsf{S}}{\arg} \min L(\mathsf{S}, \mathsf{\Gamma}_{i+1}),$ where $\mathsf{\Gamma}_{i+1}$ is fixed.

$i = i + 1$

end

The attentive reader may discern the similarity between Algorithms 1 and 2. In the upcoming chapter, we clarify that there exist certain instances in which SPA provides the same solution as K-means. However, SPA has the potential to attain an even better level of discretization. Additional features of SPA, such as solutions to the S and $\mathsf{\Gamma}$ subproblems, optimality conditions, and generalization to arbitrary spaces, can be found in the original text [22].

Chapter 4

Multiobjective Optimization With SPA

Multiobjective optimization (also known as multicriteria or vector optimization) is a field of mathematical optimization that deals with the simultaneous optimization of multiple objective functions. In the case of real-world applications, the objectives are often in conflict with one another. For example, in financial portfolio management, one may want to maximize the return on investment while minimizing risk. Multiobjective optimization aims to find the best trade-offs among these objectives, rather than optimizing them individually. This chapter provides an introduction to the fundamentals of multiobjective optimization, presents a step-by-step example, and ultimately proposes a method that integrates SPA with a Bayesian linear model.

4.1 Multiobjective Optimization Problems

In this section, we introduce multiobjective optimization problems (MOPs) in the following form.

$$\begin{aligned} &\text{Minimize} && \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\} \\ &\text{subject to} && \mathbf{x} \in \Omega, \end{aligned} \tag{4.1}$$

where $m \geq 2$, $(\forall i = 1, \ldots, m) : f_i : \mathbb{R}^n \to \mathbb{R}$ are *objective functions* and $\Omega \subset \mathbb{R}^n$ is a (nonempty) *feasible set* [24]. If an objective function f_i is to be maximized, it is equivalent to minimize the function $-f_i$. If there is no conflict between the objective functions, then a solution can be found where every objective function attains its optimum. In this trivial case, no special methods are required. Henceforth, we assume that there does not exist a single solution that is optimal with respect to every objective function, i.e., the objective functions are at least partly conflicting. Therefore, due to the contradiction and possible incommensurability of the objective functions, it is not possible to find a single solution that would be optimal for all the objectives simultaneously. MOPs are, in a sense, ill-posed [24].

Multiobjective optimization is a vast area of research and we will only scratch the surface here[1]. In the following, we introduce the *weighting method* for solving MOPs [25].

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{m} \alpha_i f_i(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \Omega, \end{aligned}$$

where $\alpha_i \geq 0$ are the *weighting coefficients* representing the relative importance of the objectives. The α_i are usually normalized so that $\sum_{i=1}^{m} \alpha_i = 1$, this variant of the weighting method is often referred to as the *weighted sum method* (WSM). The idea of the WSM is the conversion of the MOP into a single objective optimization problem using a *convex combination* of objectives [27]. The weighting coefficients α_i do not reflect the relative importance of the objectives in the proportional sense because the objectives may be incommensurable.

We demonstrate the application of multiobjective optimization on a typical financial problem of *portfolio selection*, which involves two conflicting objectives. Specifically, the goal is to maximize financial gains while minimizing financial risks.

Example 4.1
Let $x \in \mathbb{R}$, $f : \mathbb{R} \to \mathbb{R}$ represents negative financial gains, e.g. $f(x) = x^2 - 3$, and let $g : \mathbb{R} \to \mathbb{R}$ represents financial risks, e.g. $g(x) = (x-3)^2$. We use the WSM with a weighting coefficient $\alpha \in [0, 1]$. The unconstrained optimization problem with a composite objective function can be written as follows

$$x^* = \arg\min_x \alpha \cdot f(x) + (1-\alpha) \cdot g(x). \tag{4.2}$$

Table 4.1: Optimization with different values of the coefficient α.

α	x^*	Meaning
1	$\arg\min_x f(x)$	Profit maximization is paramount, risks are ignored.
0.5	$\arg\min_x 0.5 \cdot f(x) + 0.5 \cdot g(x)$	The meaning of the weight α is not known a priori.
0	$\arg\min_x g(x)$	Risk minimization is the most important, gains are ignored.

The meaning behind different choices of α is explained within Table 4.1.

[1]For more information on multiobjective optimization, see [24, 25, 26] or [27].

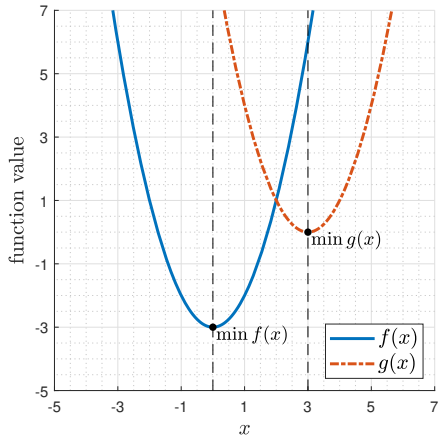

Figure 4.1: Functions f and g.

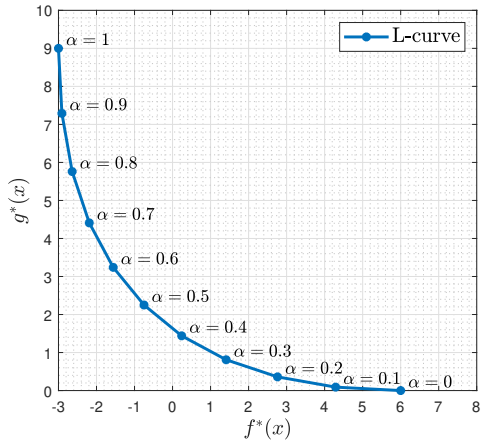

Figure 4.2: L-curve.

Fig. 4.1 displays the graphs of functions f and g with their respective minima marked with black dots, while Fig. 4.2 shows the corresponding L-curve [28]. The L-curve is a curve constructed from the solutions of optimization problems for different choices of weighting coefficient α. The curve demonstrates that for higher values of α, the optimization process prioritizes minimizing the first objective function f, resulting in a small value of f in the minimizers. In contrast, the function g is neglected, leading to a higher value in the corresponding minimizers. The opposite situation occurs when the value of α is small.

The L-curve can be used to determine the optimal value of α according to preferences. However, the two objective functions are generally conflicting, i.e., they do not share a common minimizer. Hence, the right choice of the appropriate value of the coefficient α that balances both objectives is ultimately a matter of trade-off between the two objectives. ▲

Thus far, we have sufficed with the intuitive meaning of multiobjective optimization. However, there exists extensible underlying theory. In the following, we give an overview of one of the fundamental principles, *Pareto optimality*.

A point $\mathbf{x}^* \in \Omega \subset \mathbb{R}^n$ is called *Pareto optimal* or a *Pareto minimizer* for the problem (4.1), if there does not exist $\mathbf{x} \in \Omega$ such that $(\forall i = 1, \ldots, m) : f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ and $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ for at least one index $j \in \{1, \ldots, m\}$. The set of all the Pareto optimal solutions is denoted by $\mathcal{S}$. The set $\mathcal{F} := \{f(\mathbf{x}^*) \mid \mathbf{x}^* \in \mathcal{S}\}$, which contains the image of all Pareto optimal solutions, is called the *Pareto front* [26]. In the case of Fig. 4.2, each point on the L-curve corresponds to a Pareto minimizer of the problem (4.2). It follows that in this simple example the L-curve is equivalent to the Pareto front.

4.2 MOP With SPA

In this section, we demonstrate that the MOP methodology can be applied to SPA (3.9) and a slight modification of the Bayesian linear model (2.2). In the following, we formulate a Bayesian linear model for any two stochastic processes $\mathcal{X} := \{X_t,\, t \in \mathcal{T}\}$ and $\mathcal{Y} := \{Y_t,\, t \in \mathcal{T}\}$; please notice that these stochastic processes need not be discrete-state. In order to apply the (discrete) Bayesian model, a discretization of the processes into K discrete states (clusters) is introduced. Specifically, $\mathsf{S}^X \in \mathbb{R}^{D,K}$ and $\mathsf{S}^Y \in \mathbb{R}^{M,K}$, where, in general, $K \ll T$. Let $x_t \in \mathcal{X}$ be the observed input data and let $y_t \in \mathcal{Y}$ be the output data. Finally, we can formulate the Bayesian linear model as follows

$$\mathsf{\Pi} = \mathsf{\Lambda}\mathsf{\Gamma}, \tag{4.3}$$

where $(\forall m = 1, \ldots, M)(\forall t = 1, \ldots, T)(\forall k = 1, \ldots, K)$:

$$\begin{aligned} \mathsf{\Pi}_{m,t} &= \mathcal{P}(y_t \text{ is in state } \mathsf{S}^Y_m), \\ \mathsf{\Gamma}_{k,t} &= \mathcal{P}(x_t \text{ is in state } \mathsf{S}^X_k), \end{aligned}$$

are probabilistic representations of the above discretization and where $\mathsf{\Lambda}$ is a matrix of conditional probabilities such that

$$\mathsf{\Lambda}_{m,k} = \mathcal{P}(y_t \text{ is in state } \mathsf{S}^Y_m \text{ if } x_t \text{ is in state } \mathsf{S}^X_k).$$

It follows that $\mathsf{\Pi}, \mathsf{\Lambda}$, and $\mathsf{\Gamma}$ are left stochastic matrices. Henceforth, we assume that the conditional probabilities in the matrix $\mathsf{\Lambda}$ are stationary (independent of the data index t) and depend only on the discrete state indices m and k. Moreover, when it is apparent from the context, we will omit the superscript of discrete states (S^X and S^Y). Eq. (4.3) can be justified by the the Law of Total Probability (Theorem 2.1).

Similarly to Section 2.3, one can derive a solution of the $\mathsf{\Lambda}$ problem for (4.3) by minimizing the KLD between $\mathsf{\Pi}_{:,t}$ and $\mathsf{\Lambda}\mathsf{\Gamma}_{:,t}$, $(\forall t = 1, \ldots, T)$. The form of the KLD is as follows

$$D(\mathsf{\Pi}_{:,t} \parallel \mathsf{\Lambda}\mathsf{\Gamma}_{:,t}) = -\sum_{m=1}^{M} \mathsf{\Pi}_{m,t} \cdot \ln \frac{\mathsf{\Lambda}_{m,:}\mathsf{\Gamma}_{:,t}}{\mathsf{\Pi}_{m,t}}.$$

The solution of the $\mathsf{\Lambda}$ problem in (4.3) is analogous to (2.14) and is given by

$$\mathsf{\Lambda}_{m,k} = \frac{\sum\limits_{t=1}^{T} \mathsf{\Pi}_{m,t}\mathsf{\Gamma}_{k,t}}{\sum\limits_{t=1}^{T} \mathsf{\Gamma}_{k,t}} = \frac{\langle \mathsf{\Pi}_{m,:}\,,\, \mathsf{\Gamma}_{k,:} \rangle}{\sum\limits_{t=1}^{T} \mathsf{\Gamma}_{k,t}} \in [0, 1]\,. \tag{4.4}$$

In the case of our problem, we can assume that the matrix S in SPA provides the required discretiza-

tion of the given data $\mathsf{X} \in \mathbb{R}^{D,T}$ and that $\mathsf{\Gamma} \in \mathbb{P}_L^{K,T}$ in (4.3) contains probabilistic representations of the data points $\mathsf{X}_{:,t}$. The corresponding constrained MOP is stated as follows.

$$\begin{aligned} &\text{Minimize} && \{\, L_1(\mathsf{S}, \mathsf{\Gamma}),\, L_2(\mathsf{\Lambda}, \mathsf{\Gamma})\} \\ &\text{subject to} && \mathsf{\Gamma} \in \Omega_{\mathsf{\Gamma}} \quad \text{and} \quad \mathsf{\Lambda} \in \Omega_{\mathsf{\Lambda}}, \end{aligned} \tag{4.5}$$

where $\Omega_{\mathsf{\Gamma}} = \mathbb{P}_L^{K,T}$, $\Omega_{\mathsf{\Lambda}} = \mathbb{P}_L^{M,K}$, and

$$\begin{aligned} L_1(\mathsf{S}, \mathsf{\Gamma}) &= \sum_{t=1}^{T} \left\| \mathsf{X}_{:,t} - \sum_{k=1}^{K} \mathsf{\Gamma}_{k,t} \mathsf{S}_{:,k} \right\|_2^2, \\ L_2(\mathsf{\Lambda}, \mathsf{\Gamma}) &= -\sum_{t=1}^{T} \sum_{m=1}^{M} \mathsf{\Pi}_{m,t} \cdot \ln \frac{\mathsf{\Lambda}_{m,:} \mathsf{\Gamma}_{:,t}}{\mathsf{\Pi}_{m,t}}. \end{aligned}$$

To clarify, L_1 is the SPA objective and L_2 is the KLD objective. Using the WSM with weight $\alpha \in [0, 1]$, we can formulate the corresponding composite optimization problem as follows

$$[\mathsf{S}^*, \mathsf{\Gamma}^*, \mathsf{\Lambda}^*] = \underset{\substack{\mathsf{S}, \mathsf{\Gamma}, \mathsf{\Lambda} \\ \mathsf{\Lambda} \in \Omega_{\mathsf{\Lambda}}}}{\arg\min} \; \alpha \sum_{t=1}^{T} \left\| \mathsf{X}_{:,t} - \sum_{k=1}^{K} \mathsf{\Gamma}_{k,t} \mathsf{S}_{:,k} \right\|_2^2 - (1-\alpha) \sum_{t=1}^{T} \sum_{m=1}^{M} \mathsf{\Pi}_{m,t} \cdot \ln \frac{\mathsf{\Lambda}_{m,:} \mathsf{\Gamma}_{:,t}}{\mathsf{\Pi}_{m,t}}, \tag{4.6}$$

where $\mathsf{S} \in \mathbb{R}^{D,K}$ are unknown mapping parameters between the data points $\mathsf{X}_{:,t}$ and the corresponding probability representations $\mathsf{\Gamma}_{:,t}$. The problem described in (4.6) is separable from t, which means that we could potentially solve T simpler problems rather than solving (4.6) directly. However, we have managed to find a solution that solves all T problems in a single step. The algorithm used to solve (4.6) can be expressed as a sequence of split optimization problems, similar to the SPA formulation in Section 3.2. Nonetheless, the subproblems for $\mathsf{\Gamma}$ and $\mathsf{\Lambda}$ require a numerical solution. Specifically, we utilize the *Spectral Projected Gradient* (SPG) *method* [29, 30] to solve them.

Algorithm 3: SPA + KLD

Set feasible initial approximation S_0 and $\mathsf{\Lambda}_0$.

i = 0

while *"convergence criteria"* **do**

$\mathsf{\Gamma}_{i+1} = \underset{\mathsf{\Gamma}}{\arg\min}\, \underset{\mathsf{\Gamma} \in \Omega_{\mathsf{\Gamma}}}{\min} L(\mathsf{S}_i, \mathsf{\Gamma}, \mathsf{\Lambda}_i)$ $\longrightarrow$ Numerical solver (SPG).

$\mathsf{S}_{i+1} = \underset{\mathsf{S}}{\arg\min}\, L(\mathsf{S}, \mathsf{\Gamma}_{i+1}, \mathsf{\Lambda}_i)$ $\longrightarrow$ Solution of S subproblem in SPA [22].

$\mathsf{\Lambda}_{i+1} = \underset{\mathsf{\Lambda}}{\arg\min}\, \underset{\mathsf{\Lambda} \in \Omega_{\mathsf{\Lambda}}}{\min} L(S_{i+1}, \mathsf{\Gamma}_{i+1}, \mathsf{\Lambda})$ $\longrightarrow$ Numerical solver (SPG).

$i = i + 1$

end

The performance of Algorithm 3 is limited in terms of the high computational cost of the $\mathsf{\Gamma}$ subproblem. However, there exists an alternative; the L_1 function (SPA) can be estimated by means

of the Jensen's inequality described in Theorem 2.4:

$$L_1(\mathsf{S},\mathsf{\Gamma}) = \sum_{t=1}^{T} \left\| \mathsf{X}_{:,t} - \sum_{k=1}^{K} \mathsf{\Gamma}_{k,t}\mathsf{S}_{:,k} \right\|_2^2 \leq \sum_{t=1}^{T}\sum_{k=1}^{K} \mathsf{\Gamma}_{k,t} \cdot \|\mathsf{X}_{:,t} - \mathsf{S}_{:,k}\|_2^2 = L_{kmeans}(\mathsf{S},\mathsf{\Gamma}), \tag{4.7}$$

where L_{kmeans} is the objective function of K-means (3.3), when we choose S to be equivalent to C. The motivation behind the different notation is to emphasize that the clustering provided by SPA is in general distinct from K-means, and the computation is also quite different.

Eq. (4.7) demonstrates a vital relationship between SPA and K-means, specifically that K-means produces only an upper estimate of the SPA optimization problem (3.9), when the squared Euclidean distance is employed as the distance function [22]. In other words, measured in terms of the squared Euclidean distance, the discretization provided by K-means is always suboptimal to the discretization obtained with SPA. However, if the matrix $\mathsf{\Gamma}$ is binary, the inequality (4.7) becomes an equality, thereby making K-means equivalent to SPA in Euclidean space [22]. In the upcoming section, we discuss a similar MOP involving K-means.

4.3 MOP With *K*-means

This section presents a multiobjective formulation of K-means (3.3) and the Bayesian linear model (4.3). It is worth noting that this approach bears a strong resemblance to the one introduced in Section 4.2, the main difference being that the discretization is now performed by K-means rather than SPA. The notation used in previous sections is retained. The estimate of the constrained MOP (4.5) is given as follows.

$$\begin{aligned} &\text{Minimize} && \{\, L_1(\mathsf{C},\mathsf{\Gamma}),\, L_2(\mathsf{\Lambda},\mathsf{\Gamma})\} \\ &\text{subject to} && \mathsf{\Gamma} \in \Omega_{\mathsf{\Gamma}} \quad \text{and} \quad \mathsf{\Lambda} \in \Omega_{\mathsf{\Lambda}}, \end{aligned}$$

where

$$L_1(S,\mathsf{\Gamma}) = \sum_{t=1}^{T}\sum_{k=1}^{K} \mathsf{\Gamma}_{k,t} \cdot \|\mathsf{X}_{:,t} - \mathsf{C}_{:,k}\|_2^2\,, \tag{4.8}$$

$$L_2(\mathsf{\Lambda},\mathsf{\Gamma}) = -\sum_{t=1}^{T}\sum_{m=1}^{M} \mathsf{\Pi}_{m,t} \cdot \ln \frac{\mathsf{\Lambda}_{m,:}\mathsf{\Gamma}_{:,t}}{\mathsf{\Pi}_{m,t}}. \tag{4.9}$$

Note that L_1 is now the K-means objective and L_2 remains the KLD objective. Again, the WSM with weight $\alpha \in [0,1]$ is used. The corresponding composite optimization problem can be written

as follows

$$[\mathsf{C}^*, \mathsf{\Gamma}^*, \mathsf{\Lambda}^*] = \underset{\mathsf{C},\mathsf{\Gamma},\mathsf{\Lambda}}{\arg} \min_{\substack{\mathsf{\Gamma}\in\Omega_{\mathsf{\Gamma}} \\ \mathsf{\Lambda}\in\Omega_{\mathsf{\Lambda}}}} \alpha \sum_{t=1}^{T}\sum_{k=1}^{K} \mathsf{\Gamma}_{k,t} \cdot \|\mathsf{X}_{:,t} - \mathsf{C}_{:,k}\|_2^2 - (1-\alpha) \sum_{t=1}^{T}\sum_{m=1}^{M} \mathsf{\Pi}_{m,t} \cdot \ln \frac{\mathsf{\Lambda}_{m,:}\mathsf{\Gamma}_{:,t}}{\mathsf{\Pi}_{m,t}}. \tag{4.10}$$

The problem (4.10) can be solved as a sequence of split optimization problems as shown in Algorithm 4.

Algorithm 4: K-means + KLD

Set feasible initial approximation C_0 and $\mathsf{\Lambda}_0$.

i = 0

while *"convergence criteria"* **do**

- $\mathsf{\Gamma}_{i+1} = \underset{\mathsf{\Gamma}}{\arg} \min_{\mathsf{\Gamma}\in\Omega_{\mathsf{\Gamma}}} L(\mathsf{C}_i, \mathsf{\Gamma}, \mathsf{\Lambda}_i)$ $\longrightarrow$ Numerical solver (SPG).
- $\mathsf{C}_{i+1} = \underset{\mathsf{C}}{\arg} \min L(\mathsf{C}, \mathsf{\Gamma}_{i+1}, \mathsf{\Lambda}_i)$ $\longrightarrow$ Solution of K-means in variable C (3.6).
- $\mathsf{\Lambda}_{i+1} = \underset{\mathsf{\Lambda}}{\arg} \min_{\mathsf{\Lambda}\in\Omega_{\mathsf{\Lambda}}} L(\mathsf{C}_{i+1}, \mathsf{\Gamma}_{i+1}, \mathsf{\Lambda})$ $\longrightarrow$ Numerical solver (SPG).
- $i = i + 1$

end

Although the subproblems for $\mathsf{\Gamma}$ and $\mathsf{\Lambda}$ still require a numerical solution, from a numerical standpoint this problem is substantially easier to solve than the problem with SPA. However, Algorithm 4 still has some limitations in terms of performance, particularly when dealing with large datasets where $T > 10,000$.

4.4 MOP With Jensen Estimation

Because the preceding algorithm, Algorithm 4, does not meet the desired level of performance, it is necessary to either find a more effective algorithm or to further estimate the objective function. To address this issue, we demonstrate in this section that it is possible to further estimate the second objective function, $L2$ (4.9), by using logarithmic identities and then applying the Jensen's inequality (Theorem 2.4):

$$L_2(\mathsf{\Lambda}, \mathsf{\Gamma}) = -\sum_{t=1}^{T}\sum_{m=1}^{M} \mathsf{\Pi}_{m,t} \cdot \ln \frac{\mathsf{\Lambda}_{m,:}\mathsf{\Gamma}_{:,t}}{\mathsf{\Pi}_{m,t}} = -\sum_{t=1}^{T}\sum_{m=1}^{M} \mathsf{\Pi}_{m,t} \cdot \ln \left(\sum_{k=1}^{K} \mathsf{\Lambda}_{m,k}\mathsf{\Gamma}_{k,t} \right) + \underbrace{\sum_{t=1}^{T}\sum_{m=1}^{M} \mathsf{\Pi}_{m,t} \cdot \ln \mathsf{\Pi}_{m,t}}_{=:\,\xi\,\in\,\mathbb{R}},$$

where the additive constant ξ can be omitted in the context of minimization[2]. Since the function $-\ln(\cdot)$ is convex, it follows that

$$L_2(\Lambda, \Gamma) \leq -\sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{k=1}^{K} \Pi_{m,t} \cdot \Gamma_{k,t} \cdot \ln\left(\Lambda_{m,k}\right). \tag{4.11}$$

Analogously to the previous sections, we combine the K-means objective L_1 (4.8) and the estimated KLD objective $L2$ (4.11) into a single optimization problem given by

$$[\mathsf{C}^*, \Gamma^*, \Lambda^*] = \underset{\mathsf{C},\Gamma,\Lambda}{\arg\min} \underset{\substack{\Gamma\in\Omega_\Gamma \\ \Lambda\in\Omega_\Lambda}}{\min} \alpha \sum_{t=1}^{T}\sum_{k=1}^{K} \Gamma_{k,t} \cdot \|\mathsf{X}_{:,t} - \mathsf{C}_{:,k}\|_2^2 - (1-\alpha)\sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{k=1}^{K} \Pi_{m,t} \cdot \Gamma_{k,t} \cdot \ln\left(\Lambda_{m,k}\right). \tag{4.12}$$

In the case of our problem, Γ can be factored out from (4.12) to obtain the following

$$[\mathsf{C}^*, \Gamma^*, \Lambda^*] = \underset{\mathsf{C},\Gamma,\Lambda}{\arg\min} \underset{\substack{\Gamma\in\Omega_\Gamma \\ \Lambda\in\Omega_\Lambda}}{\min} \sum_{t=1}^{T}\sum_{k=1}^{K} \Gamma_{k,t} \left(\alpha \cdot \|\mathsf{X}_{:,t} - \mathsf{C}_{:,k}\|_2^2 - (1-\alpha)\sum_{m=1}^{M} \Pi_{m,t} \cdot \ln \Lambda_{m,k} \right). \tag{4.13}$$

The constrained optimization problem (4.13) can be solved using a modified K-means algorithm, where the main difference is that the solution of the Γ subproblem is now as follows

$$(\forall t = 1, \ldots, T) : \Gamma^*_{:,t} = \underset{\Gamma_{:,t}}{\arg\min} \underset{\Gamma_{:,t}\in\mathbb{P}^K}{\min} \sum_{k=1}^{K} \Gamma_{k,t} \left(\alpha \cdot \|\mathsf{X}_{:,t} - \mathsf{C}_{:,k}\|_2^2 - (1-\alpha)\sum_{m=1}^{M} \Pi_{m,t} \cdot \ln \Lambda_{m,k} \right),$$

$$\Gamma^*_{k,t} = \begin{cases} 1, \text{ if } k^* = \underset{k}{\arg\min} \left\{ \alpha \|\mathsf{X}_{:,t} - \mathsf{C}_{:,k}\|_2^2 - (1-\alpha) \sum_{m=1}^{M} \Pi_{m,t} \cdot \ln \Lambda_{m,k} \right\}, \\ 0, \text{ elsewhere.} \end{cases} \tag{4.14}$$

Algorithm 5: K-means + KLD + Jensen

Set feasible initial approximation C_0 and Λ_0.

i = 0

while *"convergence criteria"* **do**

- $\Gamma_{i+1} = \underset{\Gamma}{\arg\min} \underset{\Gamma\in\Omega_\Gamma}{\min} L(\mathsf{C}_i, \Gamma, \Lambda_i)$ $\longrightarrow$ Solution of the Γ subproblem (4.14).
- $\mathsf{C}_{i+1} = \underset{\mathsf{C}}{\arg\min}\, L(\mathsf{C}, \Gamma_{i+1}, \Lambda_i)$ $\longrightarrow$ Solution of K-means in variable C (3.6).
- $\Lambda_{i+1} = \underset{\Lambda}{\arg\min} \underset{\Lambda\in\Omega_\Lambda}{\min} L(\mathsf{C}_{i+1}, \Gamma_{i+1}, \Lambda)$ $\longrightarrow$ Solution of the Bayesian linear model (4.4).
- $i = i + 1$

end

[2]Please recall Lemma 2.3.

Algorithm 5 summarizes the solution of (4.13). We solve three split optimization problems for Γ, C, and Λ, similar to the approach taken in previous sections. However, with the proposed estimation, the solution of each subproblem can now be expressed as a closed-form analytic formula.

This algorithm has demonstrated a respectable ability to process large datasets, making it a useful tool when dealing with a significant amount of data. In fact, its efficiency makes it suitable for datasets with sizes exceeding 10,000 data points, where some conventional algorithms may struggle to perform effectively.

Chapter 5

Applications

In this chapter, we explore the fundamental concepts and techniques behind *supervised machine learning* (SML) and the application of Algorithms 3, 4, and 5 to classification tasks. Additionally, we delve into the important topics of feature engineering, model evaluation, and hyperparameter tuning, which play a crucial role in achieving high accuracy and robustness in classification models. All applications presented in this chapter have been implemented in MATLAB and are freely available on GitHub [3].

In supervised learning, the dataset is the collection of labeled samples $(X, Y) = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$. Each element $\mathbf{x}_t$ among T is called a feature vector. A feature vector is a vector in which each dimension $d = 1, \ldots, D$ contains a value (feature) that describes the sample in some way. The label y_t can be either an element belonging to a finite set of classes $\{1, 2, \ldots, M\}$ or a real number [18]. For instance, if our samples are images of corrosion, then we can have two classes $\{\texttt{corrosion}, \texttt{not\_corrosion}\}$.

The goal of an SML algorithm is to use the dataset to produce a model that takes a feature vector $\mathbf{x}_t$ as input and outputs information that allows deducing the label for this feature vector [18].

5.1 Feature Engineering

The problem of transforming raw data into a dataset is called *feature engineering*. Everything measurable can be used as a feature. The goal is to create *informative features*: these would allow the learning algorithm to build a model that predicts the correct labels of the data used for training. Highly informative features are also called features with high predictive power [18]. We demonstrate the process of feature engineering in the following example.

Example 5.1

Suppose that we have a dataset that includes corrosion images. The aim is to identify pixels that show signs of corrosion. Consider an image, a square image with a resolution of 256×256 pixels, for simplicity. The image can be divided into tiles, hereafter called *patches*. Assume that the patches

have dimensions 16×16 pixels; the total would be $16^2 = 256$ patches. The objective is to extract informative features from the patches that would allow the learning algorithm to build a reliable model.

$$\texttt{Image} = \left[\begin{array}{cccc} & & & \\ & & & \\ & & & \\ & & & \end{array}\right]$$

For instance, we can investigate the *roughness* of individual patches because corrosion generally has a rough texture. This can be accomplished by computing the *gray-level co-occurrence matrix* (GLCM) and its corresponding features *contrast*, *correlation*, *energy (uniformity)*, and *homogeneity* [31, 32]. These features can be used to create a patch descriptor (feature vector) such that

$$\mathbf{x}_t = \begin{bmatrix} \textit{contrast} \text{ in the } t\text{-th patch} \\ \textit{correlation} \text{ in the } t\text{-th patch} \\ \textit{energy} \text{ in the } t\text{-th patch} \\ \textit{homogeneity} \text{ in the } t\text{-th patch} \end{bmatrix},$$

where $t = 1, \ldots, T$ and $T \in \mathbb{N}$ is the number of patches. Subsequently, we assemble the patch descriptors into a matrix $\mathsf{X} \in \mathbb{R}^{D,T}$ as follows

$$\mathsf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_T \end{bmatrix}.$$

▲

Frequently, the values of extracted features are on dissimilar scales. For example, one feature may range between $[0, 1]$, while another may take values from the interval $[-1000, 1000]$. This can lead to undesirable behavior, such as biased outcomes in K-means clustering. In this case, the squared Euclidean distance is computed from each data point to all centroids. When the values of the features are on different scales, one feature may dominate over the others and the features with smaller values may be neglected.

5.1.1 Normalization

A widely used and straightforward solution is to normalize the values. *Normalization* [18] is the process of converting an actual range of values that a numerical feature can acquire, into a standard range of values, typically within the interval $[0, 1]$. The normalization formula is as follows

$$\overline{\mathbf{x}}^{(d)} = \frac{\mathbf{x}^{(d)} - \min^{(d)}}{\max^{(d)} - \min^{(d)}}, \tag{5.1}$$

where $\min^{(d)}$ and $\max^{(d)}$ are, respectively, the minimum and the maximum value of the feature $\mathbf{x}^{(d)}$ in the dataset, and $\overline{\mathbf{x}}^{(d)}$ denotes the normalized feature within the interval $[0, 1]$. Eq. (5.1) is also known as *min-max normalization* or *min-max scaling*. Using the notation from the previous chapters, we can express $\mathbf{x}^{(d)}$ as a vector from the set $\{\mathsf{X}_{1,:}, \mathsf{X}_{2,:}, \ldots, \mathsf{X}_{D,:}\}$, where $\mathsf{X} \in \mathbb{R}^{D,T}$.

5.1.2 Standardization

Another prevalent option for dealing with differences in the scale of individual features is standardization. *Standardization* (or z-score normalization) [18] is a procedure during which the feature values are rescaled so that they exhibit the properties of a standard normal distribution with $\mu = 0$ and $\sigma = 1$, where μ is the mean (the average value of the feature, averaged over all samples in the dataset) and σ is the standard deviation from the mean. Standard scores (or z-scores) of features are calculated as follows:

$$\overline{\mathbf{x}}^{(d)} = \frac{\mathbf{x}^{(d)} - \mu^{(d)}}{\sigma^{(d)}},$$

where $\mu^{(d)}$ and $\sigma^{(d)}$ are, respectively, the mean and standard deviation of the d-th feature $\mathbf{x}^{(d)}$, and $\overline{\mathbf{x}}^{(d)}$ denotes the standardized feature.

A common practice in data analysis is to test both normalization and standardization methods, and then decide which one to use based on the performance of the selected algorithm [18]. Further information on evaluating the performance of machine learning algorithms will be covered in Section 5.3.

5.2 Bayesian Causal Inference

Assuming that an SML model has been trained using one of the developed algorithms, the question that arises is how to employ this model for prediction purposes. This is where *Bayesian inference* comes into play. Recall that in Algorithms 3, 4, and 5, the main goal is to find the optimal matrices C (S), $\mathsf{\Gamma}$, and $\mathsf{\Lambda}$. Essentially, $\mathsf{\Lambda}$ allows us to infer the label from a vector of features. To make a prediction using the Bayesian linear model (4.3), given matrices $\mathsf{\Lambda}$ and C, we need a discretization of the input feature vector $\mathbf{x}_t$ and the corresponding probabilistic representation $\mathsf{\Gamma}_{:,t}$ ($\forall t = 1, \ldots, T$). The discretization can be obtained using one step of the K-means algorithm with the pre-trained matrix of centroids C (or S in the case of SPA). Then, we can make a prediction using the Bayesian linear model (4.3):

$$(\forall t = 1, \ldots, T): \quad \mathsf{\Pi}_{:,t} = \mathsf{\Lambda}\mathsf{\Gamma}_{:,t},$$

where $\mathsf{\Pi}_{:,t}$ contains the predicted encoded label[1]. We use the so-called *one-hot encoding* for labels, where we create a (transposed) binary vector for each class of a finite set of classes $\{1, 2, \ldots, M\}$. For example, suppose that we have three labels and two classes $\{\texttt{corrosion}, \texttt{not\_corrosion}\}$,

[1]The prediction does not have to be binary, instead the columns of $\mathsf{\Pi}$ may contain probabilities.

corresponding one-hot encoding, where the first row represents `corrosion` and the second row represents `not_corrosion`, is illustrated below:

$$\Pi = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{matrix} \texttt{corrosion}, \\ \texttt{not\_corrosion}, \end{matrix}$$

where 1 represents class membership. In this example, the first label is `corrosion`, the second label is `not_corrosion` and the third label is again `corrosion`. Since each column of Π is a stochastic vector, we can write in terms of probability that

$$\Pi_{:,t} = \begin{bmatrix} \mathcal{P}(\mathbf{x}_t \text{ is } \texttt{corrosion}) \\ \mathcal{P}(\mathbf{x}_t \text{ is } \texttt{not\_corrosion}) \end{bmatrix}.$$

5.3 Model Evaluation

A widely used technique for evaluating machine learning models is the *three-way split*, also known as the *train-validation-test split* [18]. This method involves dividing the available data into three subsets:

1. *Training set*, which is used to train the model.

2. *Validation set*, which is used to evaluate the model during training. The model is tested on the validation data to assess how well it generalizes to new data. Moreover, the validation set is used for *hyperparameter tuning* (we address this in following Section 5.4).

3. *Test set*, which is used to measure the final performance of the model. Once the model has been trained and optimized using the training and validation sets, it is tested on the test data to assess its ability to make accurate predictions on new, unseen data.

To summarize, the three-way split is a key step in developing and evaluating machine learning models because it allows us to tune hyperparameters and assess the model's generalization performance, i.e., its ability to perform on previously unseen data.

To evaluate the performance of classification models, a set of widely accepted performance metrics and tools are typically used, including *confusion matrix*, *accuracy*, *precision / recall*, *F-score*, and *Matthew's correlation coefficient* (MCC). The confusion matrix is a table that summarizes how successful the classification model is at predicting samples belonging to various classes. One axis of the confusion matrix is the label *predicted* by the model and the other axis is the *actual* label [18].

Example 5.2
Suppose that we have a binary classification problem and that our model predicts two classes, `corrosion` and `not_corrosion`. An illustration of what a confusion matrix may look like follows.

Table 5.1: Confusion matrix.

		Predicted	
		`corrosion`	`not_corrosion`
Actual	`corrosion`	57 (TP)	7 (FN)
	`not_corrosion`	13 (FP)	42 (TN)

The above confusion matrix (Table 5.1) indicates that out of the samples that actually belonged to the `corrosion` class, 57 were accurately identified (TP – true positive). Similarly, 42 samples that did not belong to the `corrosion` class were correctly classified as `not_corrosion` (TN – true negative). However, the model also made incorrect predictions by classifying 13 samples that were `not_corrosion` as `corrosion` (FP – false positive), and 7 samples that were actually `corrosion` as `not_corrosion` (FN – false negative). ▲

In the following, we provide a concise description of the performance indicators that can be computed from the confusion matrix. *Accuracy* is a useful performance metric when errors in predicting all classes are equally important. Accuracy is given by the number of correctly classified samples divided by the total number of classified samples. In terms of the confusion matrix, it is given by the following formula [18]:

$$\text{Accuracy} := \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

Accuracy is a useful performance metric for balanced datasets, where the number of samples in each class is roughly equal. However, in the case of an imbalanced dataset, accuracy can be misleading, as a high accuracy score can be achieved by merely predicting the majority class. For example, when 10% of the samples are in the positive class and 90% of the samples come from the negative class, the model can achieve 90% accuracy by predicting solely the negative class and ignoring the positive class. This phenomenon is known as the *accuracy paradox*. Nonetheless, there are other metrics that are more suitable for unbalanced datasets, such as precision / recall, F-score, and MCC. *Precision* is the ratio of correct positive predictions to the total number of positive predictions [18]:

$$\text{Precision} := \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Recall is the ratio of correct positive predictions to the overall number of positive samples in the test set [18]:

$$\text{Recall} := \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

The definition of *F-score* or *F-measure* is given as follows [33]:

$$F_\beta := \frac{(\beta^2 + 1) \cdot \text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}},$$

where $0 \leq \beta < +\infty$ is a parameter that controls the balance between precision and recall. When $\beta = 1$, F_1 becomes equivalent to the *harmonic mean of precision and recall*. This performance metric is commonly known as F_1-score [33]:

$$F_1 := \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}.$$

It is worth noting that the F_1-score ignores true negatives, which may be important in some applications. Although accuracy and F_1-score are the prevalent performance metrics, both have their flaws. The authors of [34] recommend the use of Matthew's correlation coefficient (MCC) for binary classification problems over accuracy and F_1-score, particularly when dealing with unbalanced datasets. The formula in terms of confusion matrix is as [34] follows

$$\text{MCC} := \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}.$$

Accuracy, F_1-score, and precision / recall range from 0 to 1, where 1 represents perfect classification and 0 means perfect misclassification. In contrast, the MCC ranges from -1 to 1. However, it can be normalized to the standard interval $[0, 1]$:

$$\text{NormMCC} := \frac{\text{MCC} + 1}{2}.$$

For more information about performance metrics, please refer, for example, to this insightful comparative study by Chicco and Jurman [34].

5.4 Hyperparameter Tuning

Hyperparameters are parameters that are not optimized by the learning algorithm itself, and the best combination must be found experimentally [18]. The optimal choice of hyperparameters is crucial because they control the behavior of the algorithm and affect (often significantly) the final performance. In the case of Algorithms 3, 4, and 5 we have to optimize two hyperparameters, specifically, the weighting coefficient α and the number of clusters K. To address this issue, we adopt *grid search*, which is one of the simplest hyperparameter tuning strategies. The grid search process is straightforward and is best explained by an example.

Example 5.3
Assume that we want to evaluate the performance of the model with four different numbers of clusters $K \in \mathcal{K} = \{25, 30, 35, 40\}$ and four distinct values of the weighting coefficient $\alpha \in \mathcal{W} = \{0.9997, 0.9998, 0.9999, 1\}$. As the name of the technique suggests, we arrange the values into a grid (i.e., we perform the Cartesian product $\mathcal{K} \times \mathcal{W}$ or $\mathcal{W} \times \mathcal{K}$) and calculate the performance of the model for each of the pairs in the grid. An illustration of the results is given in Fig. 5.1.

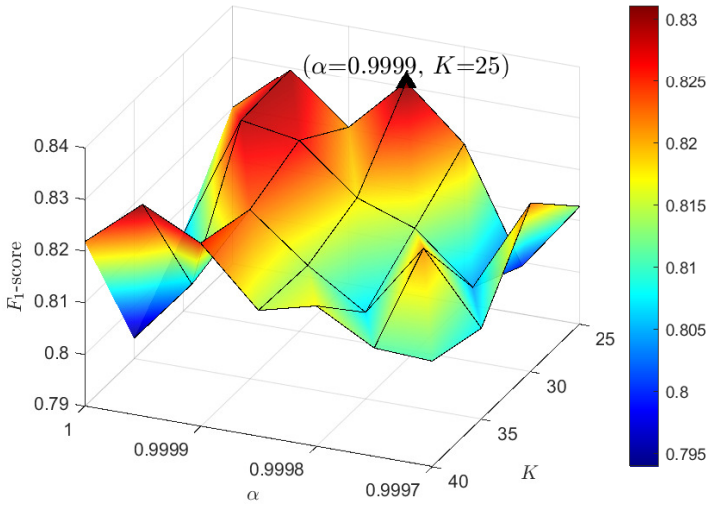

Figure 5.1: Grid search.

Fig. 5.1 demonstrates the grid search for the above problem. Here, we compare the performance by the F_1-score metric, which is plotted against the grid as a surface plot, the highest score being marked by a black triangle. Hyperparameters that result in the best model (measured by F_1-score) are $\alpha = 0.9999$ and $K = 25$. ▲

In summary, grid search is a brute-force approach that systematically tries every combination of hyperparameters defined in the grid. As with most brute-force algorithms, this approach can often be computationally demanding. In the application for corrosion detection, we used *Bayesian optimization* [35], which utilizes a more sophisticated strategy. Specifically, the Bayesian hyperparameter optimization algorithm iteratively builds a probabilistic model to decide which set of hyperparameters should be evaluated next. This approach is usually more efficient and, in the case of our problem, demonstrates better results. Additional information is covered in Section 5.7.

5.5 Synthetic Problem

Before we delve into practical applications, we demonstrate the functionality of Algorithms 3, 4, and 5 on the following synthetic problem. We generated:

- a random matrix of cluster centroids $\mathsf{C} \in \mathbb{R}^{D,K}$ (random integers from interval $[-10, 10]$),
- a random left stochastic matrix $\mathsf{\Gamma} \in \mathbb{P}_L^{K,T}$,
- a random left stochastic matrix $\mathsf{\Lambda} \in \mathbb{P}_L^{M,K}$,
- a matrix $\mathsf{X} \in \mathbb{R}^{D,T}$ such that $\mathsf{X} = \mathsf{C\Gamma}$, and

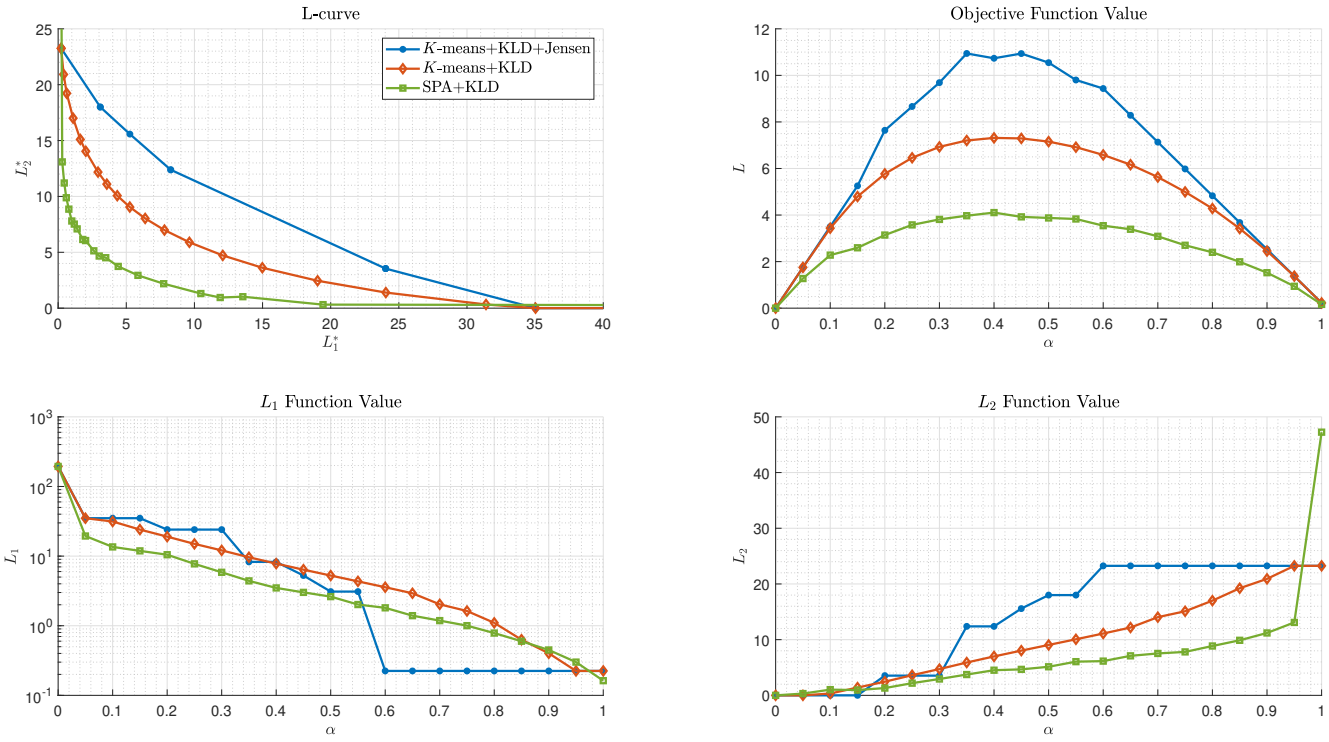

Figure 5.2: Comparison of Algorithms 3, 4, and 5 on synthetic benchmark.

- a matrix of one-hot encoded labels $\Pi \in \mathbb{P}_L^{M,T}$, such that the labels are generated according to the probabilities given by the product $\Lambda\Gamma$. If both Λ and Γ are binary, then it follows from Lemma 2.2 that $\Pi = \Lambda\Gamma$.

Moreover, we added Gaussian noise with standard deviation $\sigma = 0.15$ to matrix X, and introduced artificial labeling errors in matrix Π by deliberately setting 5% of the labels to be incorrect. The problem has a dimension of $T = 1000$, ten features $D = 10$, four true clusters $K = 4$, and three classes $M = 3$. The true Λ and Γ matrices are both binary. To mitigate the possibility of inaccurate initial approximations, three random runs are conducted for each fitting (learning) of the algorithms. The algorithms are fitted to the data X for each value of the weighting coefficient $\alpha \in \{0, 0.05, 0.1, 0.15, \ldots, 0.95, 1\}$. The benchmark results are presented in Fig. 5.2. The figure illustrates the theoretical results of the preceding chapter, demonstrating two important estimations. Firstly, K-means produces only an upper estimate of the SPA method. Secondly, the estimation of KLD with the Jensen's inequality is an upper estimate of KLD. Recall that the objective functions in Algorithms 3, 4, and 5 can be generally expressed as

$$L = \alpha \cdot L_1 + (1 - \alpha) \cdot L_2,$$

where $\alpha \in [0, 1]$. The outcome is similar to Example 4.1. When the coefficient α approaches zero, the value of L_1 is large because it is neglected in the optimization process, while the value of L_2 is minimized because it is prioritized in the optimization. The opposite occurs when the value of α is close to one.

The elapsed time for the benchmark was 4301.32 seconds, which is equivalent to approximately 72 minutes. The majority of the computation time (around 70 minutes) was consumed by the numerical solution of the Γ subproblem in Algorithm 3 (SPA+KLD). The benchmark was performed on a standard laptop computer with an AMD Ryzen 5 4600H 6-core CPU and 16 GB of RAM.

5.6 Wisconsin Breast Cancer Dataset

In this section, we compare the developed algorithms with other standard SML algorithms on the Wisconsin Breast Cancer (Diagnostic) dataset, available on the UCI Machine Learning Repository [36]. Specifically, the algorithms used for the comparison are decision tree [37], k-nearest neighbors (KNN) [38], support vector machine (SVM) with linear kernel [39], and support vector machine with Gaussian kernel [39]. The Wisconsin Breast Cancer dataset comprises 569 samples, each with 30 features extracted from digitized images of fine needle aspirate (FNA) of breast mass [36]. The primary objective is to develop a machine learning model that can accurately predict whether a given sample is *malignant* (positive class) or *benign* (negative class).

The first step is standardization[2], because the features are not on the same scale. Afterwards, for the decision tree and the SVMs, the data is split into a training set (75%) and a test set (25%). Moreover, MATLAB internally optimized the *kernel scale* hyperparameter for the SVMs. For the remaining algorithms, the data was divided into a training set (60%), validation set (20%), and test set (20%). In the case of the KNN algorithm, we tuned the hyperparameter $k \in \{1, 2, \ldots, 30\}$ (the number of neighbors) on the validation set before evaluating its performance using the optimal k on the test set. For Algorithm 4 and Algorithm 5, we performed a grid search on the validation set to select the optimal hyperparameters α and K. After consulting the L-curve to determine the optimal range of the weighting parameter α, we selected the grid as follows

$$\alpha \in \{0, 0.1, 0.2, \ldots, 0.9, 1\},$$
$$K \in \{2, 3, 4, 7, 10, 15\}.$$

The best model was chosen based on the performance on the validation set measured by the MCC. It is worth mentioning that in both of the aforementioned cases, the data is stratified[3] during splitting due to the slight imbalance of the data set. The positive class to negative class ratio is approximately 0.37. The mean results with standard deviation obtained from 10 random runs are

[2]We also tested normalization, however, the model achieved better performance with standardization.

[3]Stratified sampling is a technique in which the data are divided into training, validation (optional), and test sets so that the ratio between the classes is approximately the same.

given in Table 5.2. Algorithm 3 is excluded from the comparison due to its excessive computational cost.

Table 5.2: Comparison on Wisconsin Breast Cancer dataset.

SML algorithm	NormMCC	Accuracy	F_1-score
Gaussian SVM	0.972 ± 0.01	0.965 ± 0.02	0.973 ± 0.01
Linear SVM	0.970 ± 0.01	0.962 ± 0.01	0.971 ± 0.01
K-means + KLD + Jensen	0.958 ± 0.02	0.945 ± 0.03	0.962 ± 0.02
KNN	0.954 ± 0.02	0.936 ± 0.03	0.958 ± 0.02
K-means + KLD	0.948 ± 0.02	0.931 ± 0.03	0.952 ± 0.02
Decision tree	0.908 ± 0.02	0.885 ± 0.03	0.913 ± 0.02

Based on the data presented in Table 5.2, the support vector machines (SVMs) produced the most favorable results. Subsequently, the modified K-means (Algorithm 5), k-nearest neighbors (KNN), K-means + KLD (Algorithm 4), and the decision tree followed in terms of performance.

5.7 Corrosion Detection

In this section, we present a method for detecting corrosion from image data using an annotated dataset [40]. The dataset contains images of corrosion, where each pixel is marked as either `corrosion` or `not_corrosion`. From the original dataset, a subset of 70 images is selected and 21 of them are used for visualization (some of those images are displayed in Fig. 5.4). The remaining 49 images are utilized to train, validate, and test the SML model.

In the feature engineering phase, we first resize each image to match the height of 256 pixels and divide each image into 16×16 patches. If more than half of the pixels in a patch are labeled as `corrosion`, we consider the patch to contain corrosion and label it with the number *one* (positive class). Otherwise, we consider the patch to be free from corrosion and label it with the number *zero* (negative class). The resulting binary vector of patch labels is denoted as $\mathbf{y}$.

Afterwards, we employ a feature extraction technique inspired by Hoang [41] to obtain color features from the patches. We calculate *statistical moments*, *entropy*, and *range* for each patch to extract six features per color channel from both the HSV (hue, saturation, value) and RGB (red, green, blue) color models. In total, we obtain 36 color features.

Subsequently, we construct four *gray level co-occurrence matrices* (GLCMs) [31, 32, 42], one for each direction in the set $\{(7,7),(-7,7),(7,-7),(-7,-7)\}$, for each color channel including the gray-scale. From each matrix, we compute four features: *contrast*, *correlation*, *energy*, and *homogeneity*. Altogether, we obtain $36 + 112 = 148$ features per patch; the set of these features is referred to as a *patch descriptor*.

Eventually, the patch descriptors are assembled into a matrix $\mathsf{X} \in \mathbb{R}^{D,T}$, where $D = 148$ is the number of features and $T = 17872$ represents the number of samples (i.e., the number of patch descriptors). Because the features have varying scales, their values are transformed into the range $[0, 1]$ using *min-max normalization*[4]. Since the feature dimension D is large, we employ *principal component analysis* (PCA) [43] for the dimensionality reduction. The number of dimensions is reduced from 148 to 18 while maintaining 95% of explained variance. Given the size of the dataset, the most suitable algorithm from those developed is Algorithm 5.

To ensure the reliability of the model, we split the data X and the labels $\mathbf{y}$ into training, validation, and test sets with ratios of 60%, 20%, and 20%, respectively, where each set contains a proportional number of positive and negative samples (i.e., we perform stratified sampling). The model is trained by fitting it to the data X_{train} and the labels $\mathbf{y}_{train}$ that are drawn from the training set. Once trained, the model's hyperparameters are tuned using Bayesian optimization [35] on the validation set. We experimentally set the ranges for the hyperparameters α (weighting parameter) and K (number of clusters) as $[0.99, 1 - 10^{-8}]$ and $[2, 150]$, respectively, and chose negative MCC as the objective of Bayesian optimization. Eventually, Bayesian optimization found the optimal hyperparameters to be $\alpha = 0.9901$ and $K = 136$. Finally, we evaluated the model's performance on new unseen data represented by the test set.

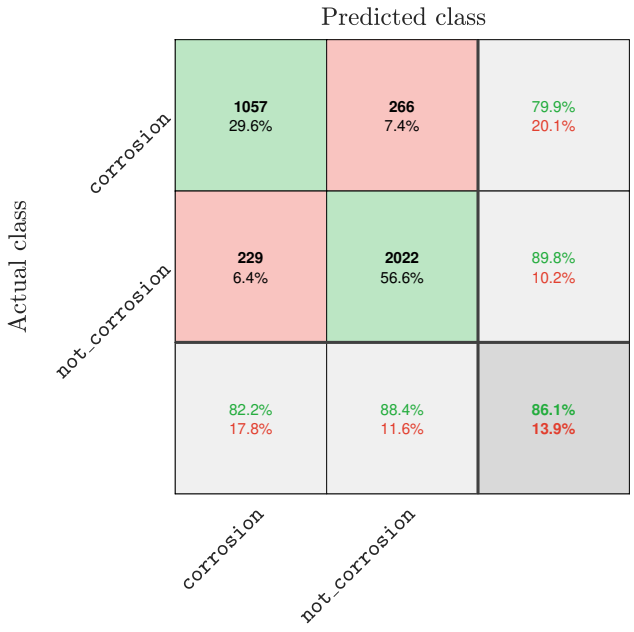

Figure 5.3: Confusion matrix for corrosion detection.

[4]Similarly to Section 5.6, we tested both normalization and standardization. However, the model performed better with normalization.

Once the model was trained and the hyperparameters tuned, we evaluated its performance using various metrics. Fig. 5.3 presents the confusion matrix for the test set. The corresponding performance metrics are as follows:

$$\begin{aligned} \text{NormMCC} &= 0.851, \\ F_1\text{-score} &= 0.810, \\ \text{Accuracy} &= 0.861, \\ \text{Precision} &= 0.822, \\ \text{Recall} &= 0.799. \end{aligned}$$

Fig. 5.4 demonstrates the algorithm's ability to generalize to previously unseen data. The first column shows the original image, while the second column displays the corresponding annotation. The third column represents the conversion of annotations to patches, and the fourth column reveals the prediction on a scale from zero to one, where one (white color) represents absolute certainty that the patch contains corrosion, and zero (black color) indicates the opposite. Finally, the last column presents the overlay of the original image with the prediction.

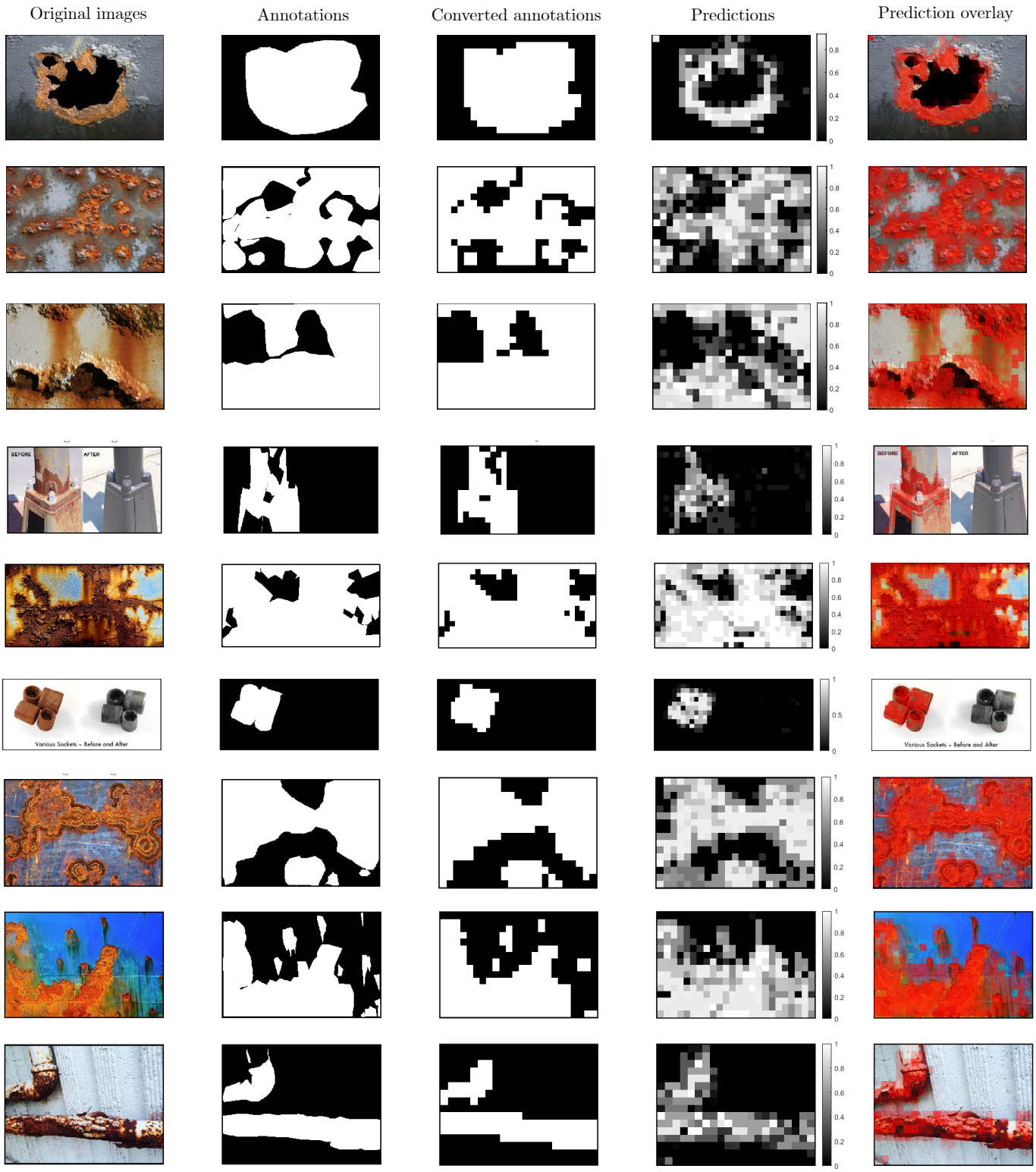

Figure 5.4: Visualization of corrosion detection on the test set.

Chapter 6

Conclusion

This thesis provides an overview of the SPA methodology and its connection to K-means, along with a technique for utilizing SPA to discretize continuous stochastic processes as a preliminary step for Bayesian causal inference modeling. Furthermore, research on the SPA methodology has been extended from the perspective of multiobjective optimization, resulting in a framework with a family of three supervised machine learning algorithms that demonstrate the applicability of the SPA methodology to a variety of classification problems. All proposed algorithms have been implemented in MATLAB and made freely available on GitHub [3]. The thesis presents results on selected applications.

Although the efficiency of the proposed models falls short of that of neural networks, they offer the advantage of being *explainable models*, unlike neural networks that are considered black box heuristics [44]. However, further research is needed to enhance the proposed methodology. This may involve the inclusion of additional image features, exploring possible image augmentations, image preprocessing techniques such as denoising, and alternative feature selection methods. Additionally, further regularization may yield improvements.

In the future, research may focus on using the SPA methodology for *time series analysis* by combining the SPA method and Markovian models. Additionally, as neural networks continue to gain popularity, we may incorporate them into future investigations and comparisons.

Bibliography

1. MAZUMDER, M. A. J. Global Impact of Corrosion: Occurrence, Cost and Mitigation. *Global Journal of Engineering Sciences*. 2020-06, vol. 5, no. 4. Available from DOI: 10.33552/gjes.2020.05.000618.
2. FAYOMI, O. S. I.; AKANDE, I. G.; ODIGIE, S. Economic Impact of Corrosion in Oil Sectors and Prevention: An Overview. *Journal of Physics: Conference Series*. 2019, vol. 1378, no. 2, p. 022037. Available from DOI: 10.1088/1742-6596/1378/2/022037.
3. FRIČ, M. *Scalable Probabilistic Approximation Method in Applications*. 2023-04. Version 1.0.0. Available also from: https://github.com/matejfric/SPAMethodInApplications.
4. LEFEBVRE, M. *Basic Probability Theory with Applications*. Springer New York, 2009. Springer Undergraduate Texts in Mathematics and Technology. ISBN 978-0-387-74995-2.
5. COLEMAN, R. What is a Stochastic Process? In: *Stochastic Processes*. Dordrecht: Springer Netherlands, 1974, pp. 1–5. ISBN 978-94-010-9796-3. Available from DOI: 10.1007/978-94-010-9796-3_1.
6. SIEGRIST, K. *Random: Probability, Mathematical Statistics, Stochastic Processes* [online]. University of Alabama in Huntsville, Department of Mathematical Sciences. [visited on 2022-06-29]. Available from: https://www.randomservices.org/random/.
7. BOYD, S.; VANDENBERGHE, L. *Convex Optimization*. New York: Cambridge University Press, 2004. ISBN 978-0-521-83378-3.
8. KULLBACK, S.; LEIBLER, R. A. On Information and Sufficiency. *The Annals of Mathematical Statistics*. 1951, vol. 22, no. 1, pp. 79–86. Available from DOI: 10.1214/aoms/1177729694.
9. COVER, T. M.; THOMAS, J. A. Elements of information theory. In: John Wiley & Sons, 1991, pp. 18–18. ISBN 0-471-20061-1.
10. POLANI, D. Kullback-Leibler Divergence. In: *Encyclopedia of Systems Biology*. Ed. by DUBITZKY, W.; WOLKENHAUER, O.; CHO, K.-H.; YOKOTA, H. New York, NY: Springer New York, 2013, pp. 1087–1088. ISBN 978-1-4419-9863-7. Available from DOI: 10.1007/978-1-4419-9863-7_1551.

11. JENSEN, J. L. W. V. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*. 1906, vol. 30, no. none, pp. 175–193. Available from DOI: 10.1007/BF02418571.

12. *Jensen's inequality* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [visited on 2023-04-01]. Available from: https://en.wikipedia.org/wiki/Jensen%27s_inequality.

13. STANKOVA, Z. E. *Convex Functions and Jensen's Inequality* [online]. University of California at Berkeley [visited on 2022-06-30]. Available from: https://math.berkeley.edu/~stankova/MathCircle/Joyce/trig/node2.html.

14. POSPÍŠIL, L. *Development of Algorithms for Solving Minimizing Problems with Convex Quadratic Function on Special Convex Sets and Applications*. Ostrava, 2015. Dissertation Thesis. VŠB - Technical University of Ostrava.

15. LUENBERGER, D. G.; YE, Y. *Linear and Nonlinear Programming*. 4th ed. New York: Springer, 2008. ISBN 978-0-387-74502-2.

16. BERTSEKAS, D. P. *Nonlinear Programming*. 2nd ed. Belmont, Massachusetts: Athena Scientific, 2004. ISBN 978-1886529007.

17. NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. 2nd ed. New York: Springer, 2006. ISBN 978-0387-30303-1.

18. BURKOV, A. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019. ISBN 1-9995795-1-8.

19. XU, K. Clustering. In: *Encyclopedia of Systems Biology*. Ed. by DUBITZKY, W.; WOLKENHAUER, O.; CHO, K.-H.; YOKOTA, H. New York, NY: Springer New York, 2013, pp. 422–422. ISBN 978-1-4419-9863-7. Available from DOI: 10.1007/978-1-4419-9863-7_511.

20. ARTHUR, D.; VASSILVITSKII, S. K-Means++: The Advantages of Careful Seeding. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. SODA '07. ISBN 978-0-898716-24-5.

21. BABAKI, B. *Why doesn't k-means give the global minimum?* [online]. [N.d.]. [visited on 2023-04-03]. Available from: https://stats.stackexchange.com/q/261316.

22. GERBER, S.; POSPÍŠIL, L.; NAVANDAR, M.; HORENKO, I. Low-cost scalable discretization, prediction, and feature selection for complex systems. *Science Advances* [online]. 2020, vol. 6, no. 5 [visited on 2023-03-12]. Available from DOI: 10.1126/sciadv.aaw0961.

23. TIKHONOV, A. N.; GONCHARSKY, A. V.; STEPANOV, V. V.; YAGOLA, A. G. *Numerical Methods for the Solution of Ill-Posed Problems*. 1st ed. Springer Dordrecht, 1995. ISBN 978-94-015-8480-7. Available from DOI: 10.1007/978-94-015-8480-7.

24. MIETTINEN, K. Concepts. In: *Nonlinear Multiobjective Optimization*. Boston, MA: Springer US, 1998, pp. 5–11. ISBN 978-1-4615-5563-6. Available from DOI: 10.1007/978-1-4615-5563-6_1.

25. HWANG, C.-L.; MASUD, A. S. M. Methods for Multiple Objective Decision Making. In: *Multiple Objective Decision Making — Methods and Applications: A State-of-the-Art Survey*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1979, pp. 243–247. ISBN 978-3-642-45511-7. Available from DOI: 10.1007/978-3-642-45511-7_3.

26. ZARE, H.; HAJARIAN, M. Determination of regularization parameter via solving a multi-objective optimization problem. *Applied Numerical Mathematics*. 2020, vol. 156, pp. 542–554. ISSN 0168-9274. Available from DOI: https://doi.org/10.1016/j.apnum.2020.05.021.

27. GHANE-KANAFI, A.; KHORRAM, E. A new scalarization method for finding the efficient frontier in non-convex multi-objective problems. *Applied Mathematical Modelling*. 2015, vol. 39, no. 23, pp. 7483–7498. ISSN 0307-904X. Available from DOI: https://doi.org/10.1016/j.apm.2015.03.022.

28. HANSEN, P. C.; O'LEARY, D. P. The Use of the L-Curve in the Regularization of Discrete Ill-Posed Problems. *SIAM Journal on Scientific Computing*. 1993, vol. 14, no. 6, pp. 1487–1503. Available from DOI: 10.1137/0914086.

29. BIRGIN, E. G.; MARTÍNEZ, J. M.; RAYDAN, M. Nonmonotone Spectral Projected Gradient Methods on Convex Sets. *SIAM Journal on Optimization*. 2000, vol. 10, no. 4, pp. 1196–1211. Available from DOI: 10.1137/S1052623497330963.

30. POSPÍŠIL, L.; GAGLIARDINI, P.; SAWYER, W.; HORENKO, I. On a scalable nonparametric denoising of time series signals. *Communications in Applied Mathematics and Computational Science*. 2018, vol. 13, no. 1, pp. 107–138. Available from DOI: 10.2140/camcos.2018.13.107.

31. POSPÍŠIL, L.; FRIČ, M.; ČERMÁK, M. The texture roughness measures for engineering problems. In: *International Conference of Numerical Analysis and Applied Mathematics (ICNAAM), will be published in AIP Conference Proceedings*. 2022.

32. HARALICK, R. M.; SHANMUGAM, K.; DINSTEIN, I. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*. 1973, vol. SMC-3, no. 6, pp. 610–621. Available from DOI: 10.1109/TSMC.1973.4309314.

33. SASAKI, Y. The truth of the F-measure. *Teach Tutor Mater*. 2007-01. Available also from: https://www.researchgate.net/publication/268185911_The_truth_of_the_F-measure.

34. CHICCO, D.; JURMAN, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*. 2020-01, vol. 21, no. 1. Available from DOI: 10.1186/s12864-019-6413-7.

35. WU, J.; CHEN, X.-Y.; ZHANG, H.; XIONG, L.-D.; LEI, H.; DENG, S.-H. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. *Journal of Electronic Science and Technology*. 2019, vol. 17, no. 1, pp. 26–40. ISSN 1674-862X. Available from DOI: https://doi.org/10.11989/JEST.1674-862X.80904120.

36. MANGASARIAN, O. L.; STREET, W. N.; WOLBERG, W. H. *Diagnostic Wisconsin Breast Cancer Database* [UCI Machine Learning Repository]. 1995. Available from DOI: 10.24432/C5DW2B.

37. TIMOFEEV, R. A. *Classification and Regression Trees (CART) Theory and Applications*. Humboldt University, Berlin, 2004.

38. BEYER, K.; GOLDSTEIN, J.; RAMAKRISHNAN, R.; SHAFT, U. When Is "Nearest Neighbor" Meaningful? In: BEERI, C.; BUNEMAN, P. (eds.). *Database Theory — ICDT'99*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 217–235. ISBN 978-3-540-49257-3.

39. HEARST, M.; DUMAIS, S.; OSUNA, E.; PLATT, J.; SCHOLKOPF, B. Support vector machines. *IEEE Intelligent Systems and their Applications*. 1998, vol. 13, no. 4, pp. 18–28. Available from DOI: 10.1109/5254.708428.

40. NGUYEN, T. X. *Corrosion Annotated* [online]. 2021. [visited on 2023-04-21]. Available from: https://www.kaggle.com/datasets/tungxnguyen/corrosionannotated.

41. HOANG, N.-D. Image Processing-Based Pitting Corrosion Detection Using Metaheuristic Optimized Multilevel Image Thresholding and Machine-Learning Approaches. *Mathematical Problems in Engineering*. 2020-05, vol. 2020, pp. 3–5. Available from DOI: 10.1155/2020/6765274.

42. KHAYATAZAD, M.; PUE, L. D.; WAELE, W. D. Detection of corrosion on steel structures using automated image processing. *Developments in the Built Environment*. 2020, vol. 3. Available from DOI: 10.1016/j.dibe.2020.100022.

43. JOLLIFFE, I. T.; CADIMA, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 2016-04, vol. 374, no. 2065. Available from DOI: 10.1098/rsta.2015.0202.

44. RUDIN, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*. 2019-05, vol. 1, no. 5, pp. 206–215. Available from DOI: 10.1038/s42256-019-0048-x.

Appendix A

K-means

```
function [C, Gamma, sse, it] = kmeans(X, K, maxIters, eps)

sse = zeros(1,maxIters+1); sse(1) = Inf; % Sum of squares error
T = size(X,2); % Number of data points in the dataset
[C] = get_kmeans_pp_centroids(X, K); % K-means++ initialization

for it = 2:maxIters+1
        Gamma = zeros(K, T);
        for t = 1:T
            [sse_t,id] = min( sum( (C - X(:,t) ).^2 , 1 ));
            sse(it) = sse(it) + sse_t;
            Gamma(id,t) = 1;
        end
        for k = 1:K
            ids = Gamma(k,:) == 1;
            C(:,k) = mean(X(:,ids),2);
        end
        if norm(sse(it-1) - sse(it)) < eps
            sse = nonzeros(sse(2:end));
            it = it-1;
            break
        end
end
end
```

Listing A.1: Basic implementation of the K-means algorithm in MATLAB.

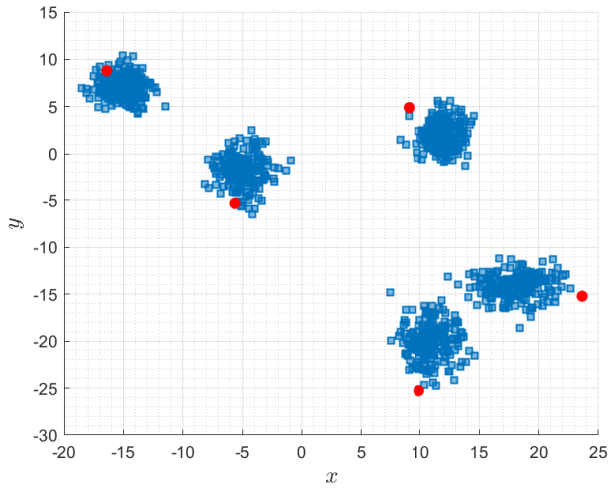

Figure A.1: K-means++ initialization.

Displayed in Fig. A.1 are the red circles representing the initial centroid approximation generated by the K-means++ algorithm on a dataset of 2D points (blue squares). These centroids serve as starting points for the iterative K-means clustering algorithm. The K-means++ algorithm is a modification of the standard K-means algorithm, which utilizes a more intelligent method to select the initial centroids to improve the convergence rate and quality of the resulting clusters [20].

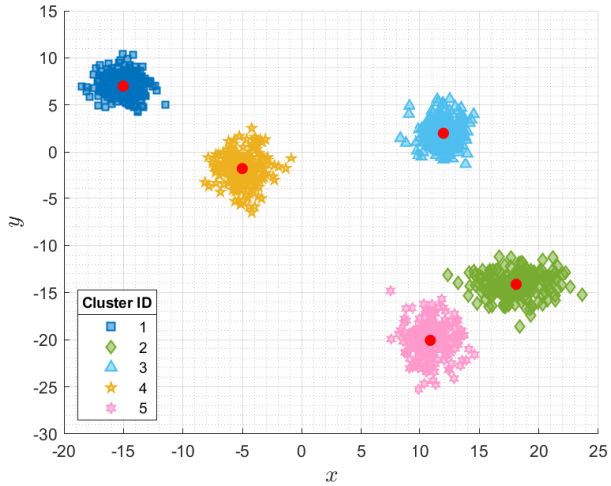

Figure A.2: Clustering results using the K-means++ algorithm.

Fig. A.2 illustrates the resulting clustering of a dataset consisting of 2D points obtained by applying the K-means++ algorithm. In this case, the K-means++ algorithm partitions the data points into five clusters based on their similarity in terms of the Euclidean distance.

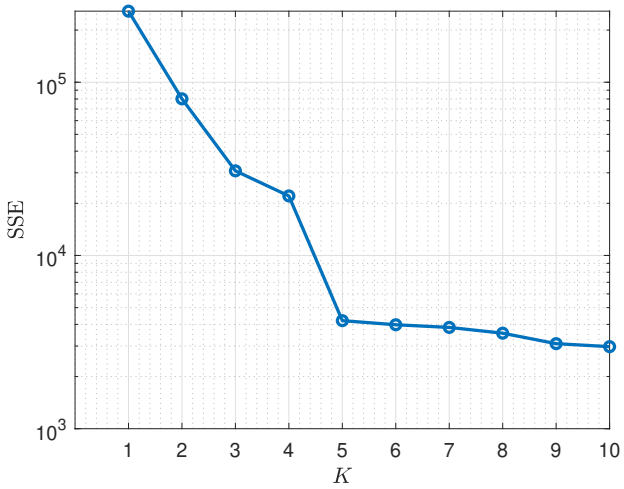

Figure A.3: Sum of squares error (SSE) for different number of clusters K.

Figure A.3 demonstrates the *elbow method heuristic*. The elbow method is a clustering analysis technique that involves plotting the SSE against the number of clusters. The optimal number of clusters is determined by selecting the "elbow" point on the graph, which balances minimizing within-cluster variance and avoiding overfitting. In the case of Figure A.3, based on the elbow method heuristic, $K = 5$ seems to be a sensible choice. Furthermore, the visualization in Fig. A.2 confirms the presence of five clusters in the data.