UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

DOUTORADO EM ENGENHARIA DE TELEINFORMÁTICA

ANTONIO AUGUSTO TEIXEIRA PEIXOTO

THEORETICAL AND APPLIED CONTRIBUTIONS ON TENSOR LEARNING

FORTALEZA

2022

ANTONIO AUGUSTO TEIXEIRA PEIXOTO

THEORETICAL AND APPLIED CONTRIBUTIONS ON TENSOR LEARNING

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Engenharia de Teleinformática. Área de Concentração: Engenharia de Teleinformática

Orientador: Prof. Dr. Carlos Alexandre Rolim Fernandes

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

P43t Peixoto, Antonio Augusto Teixeira Peixoto.
THEORETICAL AND APPLIED CONTRIBUTIONS ON TENSOR LEARNING / Antonio Augusto Teixeira Peixoto Peixoto. – 2022.
147 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2022.
Orientação: Prof. Dr. Carlos Alexandre Rolim Fernandes.

1. aprendizado tensorial. 2. aprendizado de máquina. 3. classificação de eventos sísmicos. 4. redução de dimensionalidade. 5. amostragem multilinear. I. Título.

CDD 621.38

ANTONIO AUGUSTO TEIXEIRA PEIXOTO

THEORETICAL AND APPLIED CONTRIBUTIONS ON TENSOR LEARNING

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Engenharia de Teleinformática. Área de Concentração: Engenharia de Teleinformática

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Carlos Alexandre Rolim Fernandes (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. André Lima Férrer de Almeida
Universidade Federal do Ceará (UFC)

Prof. Dr. Walter da Cruz Freitas Júnior
Universidade Federal do Ceará (UFC)

Prof. Dr. Auzuir Ripardo de Alexandria
Instituto Federal do Ceará (IFCE)

Prof. Dr. João Paulo Javidi da Costa
Hochschule Hamm-Lippstadt

To my father Wagner, and to my mother Rosa, for always believing and supporting me and my education. To my girlfriend and best partner ever, Alice. I also dedicate this thesis to Professor Alexandre Moreira de Morais, who left us in 2014, and was an example of teacher and person.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Prof. Dr. Carlos Alexandre, for the dedication, patience and friendship he had with me during the 5 long years of my doctorate.

My girlfriend, Alice, I really want to thank you for your support. In the very difficult times, when I was about to give up, or when I was down, you always supported and cheered me up. That I will never forget.

My friends João Paulo, Allex, David, Gabriel, Batata and Paulo Henrique, you guys helped me, even without knowing it, because in each moment of fraternization, I gained strength to continue.

I also thank my co-workers at IFCE, Vitor, Navar, Isaac, Luis, Rodrigo and Jardel. Each week at work became less tiring with your company and friendship.

My long time friend Bruno, in which we shared, years ago, dreams about becoming doctoral students. Your help and support in these many years was very important, because you followed my doctorate from the beginning until now, just as I followed yours.

Thanks to my graduate and undergraduate professors, Fatima Sombra, Michela, Rafael Lima, Alexandre Moreira de Morais, George Thé, Sérgio Antenor, Yuri Carvalho, Jarbas Nunes, Alexandre Coelho, João Batista, Alda Karine and others. Without your teachings and help, I would not have come this far.

I am grateful for the support of my family, especially my father, Wagner, and my mother, Rosa.

Thanks to Pablo Lara and Adolfo Inza, who helped me with their knowledge in Volcanology. Also, thanks to IGP for providing the seismic database used in this thesis.

Thanks to professor Glendo, of IFCE, and to the Fotônica Laboratory personal, for providing the photonic database used.

I thank everyone who was by my side, even when it became very difficult to support me.

“What you leave behind is not what is engraved in stone monuments, but what is woven into the lives of others.”

(Pericles)

ABSTRACT

Tensor learning refers to a series of tools and techniques for modeling and understanding complex multidimensional datasets, in order to build efficient learning models. It is a fairly new area in machine learning, which blends, especially, with statistical learning and tensor models. Indeed, tensor learning has proven to be an alternative to conventional vector-based learning techniques that are well-known in the literature, such as the Support Tensor Machines (STM), which is a multilinear extension of the Support Vector Machines (SVM). This work proposes theoretical and applied contributions to tensor learning in the following way. First, a fully tensorial framework for seismic event classification is proposed, which performs feature extraction, dimensionality reduction and classification, with all these steps using tensor-based techniques. Next, a new dimensionality reduction technique called Low-Correlation Multilinear Dimensionality Reduction (LC-MDR), based on a new tensor decomposition, the Even-Order Nested PARAFAC Decomposition (EONPD), is proposed and validated, successfully reducing the correlation and dimensionality of seismic data, improving classification rates. And last, a multilinear sampling approach for tensor learning and data structuring is proposed, which is used in the modification of the SVM technique, joining the concept of multilinear sampling and tensor decompositions, more specificaly, the PARAFAC and Tucker, then, the proposed modifications, called Support Vector Machines with Multilinear PARAFAC Sampling (SVM-MPS) and Support Vector Machines with Multilinear Tucker Sampling (SVM-MTS), are tested for photonic data classification. These three new proposed contributions to tensor learning showed better performance, in terms of accuracy, when compared to other techniques of the literature.

Keywords: tensor learning. machine learning. seismic event classification. feature extraction. dimensionality reduction. tensor decompositions. multilinear sampling. photonic classification.

RESUMO

O aprendizado tensorial refere-se a uma série de ferramentas e técnicas para modelagem e compreensão de conjuntos de dados multidimensionais complexos, a fim de construir modelos de aprendizado eficientes. É uma área relativamente nova em aprendizado de máquina, que se mistura, principalmente, com aprendizado estatístico e modelos tensoriais. De fato, o aprendizado tensorial provou ser uma alternativa às técnicas convencionais de aprendizado baseado em vetores que são bem conhecidas na literatura, como as Support Tensor Machines (STM), que são uma extensão multilinear das Support Vector Machines (SVM). Este trabalho propõe contribuições teóricas e aplicadas em aprendizado tensorial, da seguinte forma. Primeiramente, é proposto um arcabouço totalmente tensorial para classificação de eventos sísmicos, que realiza extração de características, redução de dimensionalidade e classificação, sendo todas estas etapas realizadas com técnicas baseadas em tensores. Em seguida, uma nova técnica de redução de dimensionalidade chamada Low-Correlation Multilinear Dimensionality Reduction (LC-MDR), baseada em uma nova decomposição tensorial, a Even-Order Nested PARAFAC Decomposition (EONPD), é proposta e validada, reduzindo com sucesso a correlação e dimensionalidade de dados sísmicos, melhorando as taxas de classificação. E por último, é proposta uma abordagem de amostragem multilinear para aprendizado tensorial e estruturação de dados, que é utilizada na modificação da técnica SVM, juntando o conceito de amostragem multilinear e decomposição tensorial, mais especificamente, as decomposições PARAFAC e Tucker, em seguida, as modificações propostas, denominadas Support Vector Machines with Multilinear PARAFAC Sampling (SVM-MPS) e Support Vector Machines com Multilinear Tucker Sampling (SVM-MTS), são testados para classificação de dados fotônicos. Essas três novas contribuições propostas na área de aprendizado tensorial apresentaram melhor desempenho, em termos de acurácia, quando comparadas a outras técnicas da literatura.

Palavras-chave: aprendizado tensorial, aprendizado de máquina, classificação de eventos sísmicos. extração de características. redução de dimensionalidade. decomposições tensoriais. amostragem multilinear. classificação de dados fotônicos.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ALS	Alternating Least Squares
ANN	Artificial Neural Network
CANDECOMP	Canonical Decomposition
CCA	Canonical Correlation Analysis
CNN	Convolutional Neural Network
CP	Canonical Polyadic
DATER	Discriminant Analysis with Tensor Representation
DFT	Discrete Fourier Transform
EMD	Empirical Mode Decomposition
EONPD	Even-Order Nested PARAFAC Decomposition
EX	Explosion
GNPD	Generalized Nested PARAFAC Decomposition
HB	Hybrid
HONPD	High-Order Nested PARAFAC Decomposition
HONTD	High-Order Nested Tucker Decomposition
HOSVD	High-Order Singular Value Decomposition
ICA	Independent Component Analysis
IGP	Geophysical Institute of Peru
KNN	*k*-Nearest Neighbors
LC-MDR	Low-Correlation Multilinear Dimensionality Reduction
LDA	Linear Discriminant Analysis
LP	Long Period
MDA	Multilinear Discriminant Analysis
MDFT	Multidimensional Discrete Fourier Transform
ML	Machine Learning
MPCA	Multilinear Principal Component Analysis
MPCA-DR	Multilinear Principal Component Analysis - Dimensionality Reduction
MPCA-FP	Multilinear Principal Component Analysis - Full Projection
MZI	Mach-Zehnder Interferometer
NLP	Natural Language Processing
NMF	Non-negative Matrix Factorization

NPD	Nested PARAFAC Decomposition
NTD	Nested Tucker Decomposition
PARAFAC	Parallel Factor Analysis
PCA	Principal Component Analysis
PCA-DR	Principal Component Analysis - Dimensionality Reduction
PCA-FP	Principal Component Analysis - Full Projection
PSD	Power Spectral Density
QP	Quadratic Programming
R1-SPM	Rank-1 Support PARAFAC Machines
SPM	Support PARAFAC Machines
STL	Supervised Tensor Learning
STM	Support Tensor Machines
STTM	Support Tensor-Train Machines
STuM	Support Tucker Machines
SVD	Singular Value Decomposition
SVM	Support Vector Machines
SVM-MPS	Support Vector Machines with Multilinear PARAFAC Sampling
SVM-MTS	Support Vector Machines with Multilinear Tucker Sampling
SVR	Support Vector Regression
TL	Tensor Learning
TR	Tremors
TSL	Tensor Subspace Learning
TT	Tensor-Train
VT	Volcano-tectonic

LIST OF SYMBOLS

j	Square root of -1
$\mathbb{R}$	Field of real numbers
$\mathbb{C}$	Field of complex numbers
$(\cdot)^T$	Transpose operator
$(\cdot)^+$	Pseudo-inverse operator
$(\cdot)^*$	Complex conjugate
$Tr(\cdot)$	Trace operator
$vec(\cdot)$	Vectorization operator
$\cdot \underset{k}{*} \cdot$	Contraction operator over index k
$\cdot \circ \cdot$	Outer product
$\langle \cdot, \cdot \rangle$	Inner product
$\|\| \cdot \|\|_F$	Frobenius norm
$\cdot \otimes \cdot$	Kronecker product
$\cdot \odot \cdot$	Khatri-Rao product
$\cdot \times_n \cdot$	Mode-n product
$\mathbf{A}_{\otimes}$	Short notation for the Kronecker product of N factor matrices
$\mathbf{A}_{\otimes}^{(n)}$	Short notation for the Kronecker product of $N-1$ factor matrices
$\mathbf{A}_{\odot}$	Short notation for the Khatri-Rao product of N factor matrices
$\mathbf{A}_{\odot}^{(n)}$	Short notation for the Khatri-Rao product of $N-1$ factor matrices

CONTENTS

1 INTRODUCTION

This thesis presents theoretical and applied contributions on tensor learning, aiming to promote and disseminate this important branch of machine learning. The original contributions of the present thesis can be divided into three different branches, all of them focusing in proposing tensor-based approaches with advantages over existing vector and tensor methods.

The rest of this chapter is divided as follows. First, the topics and state of the art are presented alongside the motivations for this work. After that, the contributions of this thesis are described. Then, the applications in which the proposed techniques were tested are outlined, and, in the sequel, the scientific outputs of this work are presented, with the organization of this work presented in the following.

1.1 Motivation and Justification

In this section, the motivation and justification of this thesis contributions are presented, covering the topics of machine learning, tensor learning, tensor-based classifiers, feature extraction and dimensionality reduction.

1.1.1 Introduction to Machine Learning

The goal of building systems that can adapt to their environments and learn from their experience has attracted researchers from many fields, including computer science, engineering, mathematics, physics, neuroscience, and cognitive science. Out of this research topic has come a wide variety of learning techniques that, to this day, are transforming many industrial and scientific fields. These techniques can be summarized in a common topic: Machine Learning (ML).

ML encompasses a broad range of algorithms and computational modeling tools, that are used for a wide variety of data processing applications (JORDAN; MITCHELL, 2015). Such field of research is one of today's most rapidly growing technical ones, lying at the intersection of computer science and data science, and at the core of artificial intelligence (CARLEO *et al.*, 2019). Many applications now employ ML techniques, such as image and speech recognition, language processing, physics, biomedical engineering, telecommunications, health care, manufacturing, education, financial modeling, policing and etc (JORDAN; MITCHELL, 2015; CARLEO *et al.*, 2019).

1.1.2 Introduction to Tensor Learning

Usually, in ML applications, vectors are used to represent the input data. When these inputs are represented as matrices or higher-order arrays (tensors), a vectorization of these arrays is commonly done. However, the process of vectorization breaks the structure of the data, which may lead to performance issues (MA *et al.*, 2017), and also leads to input vectors with very large dimensions. Dealing with high-dimensional data have some drawbacks. Firstly, the computational complexity of the techniques become higher, as there is a high number of features to be processed. Another problem is the so-called curse of dimensionality (BELARBI *et al.*, 2017), which refers to several phenomena that arise when dealing with with high-dimensional data. Indeed, when the number of features is bigger or closer to the sample size, the ML techniques tend to perform poorly.

From this perspective, multidimensional arrays, known as tensors, can be used instead of vectors to eliminate the necessity of vectorizing the data. Tensors have their use already demonstrated in several areas, as seen in (KOLDA; BADER, 2009) and in (COMON, 2014). Moreover, many multidimensional data applications exploit tensor learning approaches, such as recognition of: face (YAN *et al.*, 2006), gait (TAO *et al.*, 2007a), fingerprint (WANG *et al.*, 2009), images (GUO *et al.*, 2014), etc. An advantage of using tensors in comparison to matrices is due to the fact that tensors allows us the use of multidimensional data, which contrasts with two dimensions data represented by matrices, thus allowing a better understanding and processing for a multidimensional perspective.

While matrix methods form the cornerstone of ML and data analysis, tensor methods have been gaining space. Also, tensor representation of data signals open us possibilities to exploit tensor decompositions, which has gained attention in some signal processing applications, as described in (ALMEIDA *et al.*, 2016) and in ML (SIDIROPOULOS *et al.*, 2017; JI *et al.*, 2019). Moreover, different tensor decompositions, such as the Parallel Factor Analysis (PARAFAC) and Tucker decompositions (KOLDA; BADER, 2009; HARSHMAN, 1970; TUCKER, 1966) can be used during the data analysis and processing steps of a machine learning framework, providing more flexibility when dealing with tensor data.

An alternative to conventional linear and vectorial learning approaches for ML applications are the Tensor Learning (TL) methods, where many conventional learning techniques can be generalized to take n-order tensors as inputs (TAO *et al.*, 2007b). This avoids data vectorization and the consequent destruction of the data structure (HE *et al.*, 2017). Surely, in

tensor learning methods, the multidimensional structural information of the data is preserved, providing a better multidimensional modeling (HE *et al.*, 2017; TAO *et al.*, 2007b).

As a result, the use of multilinear algebra, such as tensor decompositions, combined with the concepts of tensor learning, over the last years, provided the development of tensor-based versions of classical vector-based ML methods (SIDIROPOULOS *et al.*, 2017). Tensor-based ML techniques also alleviate the curse of dimensionality and reduce the number of unknown parameters. Also, multilinear representations may help reduce the overfitting in vector-based learning applications and exploit the discriminating nonlinear relationships of tensor data, improving the performance of learning tasks (HE *et al.*, 2017; ZHOU *et al.*, 2013). As a consequence, TL-based approaches appears as an interesting alternative to conventional vectorial methods when multidimensional inputs are used (TAO *et al.*, 2007b).

1.1.3 Introduction to Support Tensor Machines (STM)

The Support Tensor Machines (STM) deserves special attention. The STM technique is a tensor-based version of the Support Vector Machines (SVM), which is one of the most well-known and popular machine learning techniques, widely used on various classification and regression problems (BURGES, 1998; MATHUR; FOODY, 2008; FOODY; MATHUR, 2004). The STM extends the SVM to tensor patterns by constructing multilinear models to the weight tensor (TAO *et al.*, 2007b; CAI *et al.*, 2006).

Several STM algorithms have been proposed, assuming different models for the weight tensor, generally providing significant performance gains with respect to the SVM. In (CAI *et al.*, 2006), a STM algorithm is proposed using the PARAFAC, also known as Canonical Decomposition (CANDECOMP) or Canonical Polyadic (CP), by assuming a rank-one structure for the weight tensor, with application to text categorization. In (KOTSIA *et al.*, 2012), this technique is generalized for a PARAFAC weight tensor of higher rank, called Support PARAFAC Machines (SPM). Generalizations of these STM models have been proposed, as the Support Tucker Machines (STuM) (KOTSIA; PATRAS, 2011) and Support Tensor-Train Machines (STTM) (CHEN *et al.*, 2019). Moreover, in (HE *et al.*, 2017), a linear kernelized STM model was proposed.

1.1.4 Feature Extraction and Dimensionality Reduction

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It generally yields better results than applying machine learning directly to the raw data (ZHOU *et al.*, 2012).

Generally, features can be categorized as: relevant, irrelevant, or redundant. In feature selection process a subset from available features data are selected for the process of learning algorithm. The best subset is the one with least number of dimensions that most contribute to learning accuracy (KHALID *et al.*, 2014). Therefore, a properly optimized feature extraction is the key to effective model construction.

The most popular and widely used feature extraction approach is the Principal Component Analysis (PCA), which is a simple non-parametric method used to extract the most relevant information from a set of redundant or noisy data (KHALID *et al.*, 2014). Other example of vector-based feature extraction technique is the Independent Component Analysis (ICA) (CHOI *et al.*, 2005).

In contrast to vector-based feature extraction techniques, multidimensional feature extraction can extend feature extraction approaches to multilinear structures. A good example is the Multidimensional Discrete Fourier Transform (MDFT), the extension of the Discrete Fourier Transform (DFT) to multilinear arrays (TOLIMIERI *et al.*, 2012), which can be used to perform Fourier analysis on multidimensional arrays, yielding features that can be used later in classification (KARMAKAR *et al.*, 2021).

On the other hand, recently, many applications of ML are increasing the amount of vectorial and multidimensional data being generated. This leads to an enormous demand for learning algorithms to extract useful information from these massive data (LU *et al.*, 2011). These massive multidimensional data are, sometimes, highdimensional, with a large amount of redundancy, and only occupying a subspace of the input space (LU *et al.*, 2011). Thus, for feature extraction, dimensionality reduction is frequently employed to map highdimensional data to a low-dimensional space while retaining as much information as possible (SHAKHNAROVICH; MOGHADDAM, 2005). However, this is a challenging problem due to the large variability and complex pattern distribution of the input data, and, sometimes, the limited number of samples available for training in practice (LU *et al.*, 2011; LU *et al.*, 2006).

Dimensionality reduction techniques can be applied to avoid these issues by trans-

forming the data from a high-dimensional space into a low-dimensional space without losing significant information of the original data set. When the input samples are correlated, they may be confined into a subspace, where an adequate low-dimensional space representation is possible. Within this context, linear techniques such as the Non-negative Matrix Factorization (NMF) (PAUCA *et al.*, 2006), Linear Discriminant Analysis (LDA) (THARWAT *et al.*, 2017), Canonical Correlation Analysis (CCA) (HARDOON *et al.*, 2004) and PCA (WOLD *et al.*, 1987) are popular solutions for dimensionality reduction. In particular, the PCA is the most commonly used dimensionality reduction technique. Indeed, this method is also powerful tool that transforms a set of correlated data into uncorrelated data using an orthonormal basis (RODARMEL; SHAN, 2002). However, said linear techniques may not be the best solution for tensor-based data, as already said earlier, the vectorization of the data would break the multilinear structure.

Linear subspace learning algorithms are traditional dimensionality reduction techniques that represent input data as vectors and solve for an optimal linear mapping to a lower-dimensional space. Unfortunately, they often become inadequate when dealing with multidimensional data. They result in very high-dimensional vectors, lead to the estimation of a large number of parameters, and also break the natural structure and correlation in the original data (LU *et al.*, 2008; TAO *et al.*, 2007b). Based on this motivations, Tensor Subspace Learning (TSL) approaches (HE *et al.*, 2005) may be a solution to said problems, which can take advantages of tensor analysis into subspace learning applications.

Therefore, tensor dimensionality reduction is a hot research topic in machine learning, which learns data representations by preserving the original data structure while avoiding convert samples into vectors and solving the problem of high dimensionality of tensor data (NIU; MA, 2021). In this scope, the Multilinear Principal Component Analysis (MPCA) (LU *et al.*, 2008), an extension of the PCA for tensor patterns, is a multilinear dimensionality reduction technique that can be exploited by tensor learning applications. As well as the PCA, the MPCA also reduces the correlation among the variables, making the MPCA very suitable for classification problems with multidimensional data sets, generating low-dimensional matrix or tensor patterns that can be fed into classifiers (PORGES; FAVIER, 2011).

Another tensor-based dimensionality reduction technique is the multilinear extension of the LDA proposed in (YAN *et al.*, 2005), called Discriminant Analysis with Tensor Representation (DATER). In (YAN *et al.*, 2006), another extension of the LDA was proposed, denoted by Multilinear Discriminant Analysis (MDA), where multiple lower-dimensional discriminative

subspaces are derived for feature selection.

In the next sections, we describe the contributions of this thesis and the applications in which these tensor learning contributions were used.

1.2 Thesis Contributions

As already mentioned, the present thesis proposes contributions in applied and theoretical tensor learning. As said earlier, the original contributions of this thesis can be divided into three different branches, all of them focusing in tensor-based approaches with advantages over existing vector and tensor methods, by proposing the following.

First, a fully tensorial framework, which performs feature extraction, dimensionality and classification is proposed. The framework was employed in seismic event classification. In addition, no previous work has used tensor learning for classifying seismic events. Second, a new dimensionality reduction technique called Low-Correlation Multilinear Dimensionality Reduction (LC-MDR) is proposed. The LC-MDR was also employed in seismic event classification. And third, a multilinear sampling approach for tensor learning and data structuring is proposed, which is employed within the SVM technique, joining the concept of multilinear sampling and tensor decompositions, generating two classifiers, denoted by Support Vector Machines with Multilinear PARAFAC Sampling (SVM-MPS) and Support Vector Machines with Multilinear Tucker Sampling (SVM-MTS), which were used in photonic output classification.

Therefore, we present the following original contributions:

- A tensor-based learning framework for classifying volcanic-seismic events.
- A scalable tensorial methodology to use data from multiple seismology stations and multiple sensors, to measure the events of the volcano;
- The joint use of multilinear techniques for feature extraction (MDFT), dimensionality reduction (MPCA) and classification (SPM, STuM).
- The use of a complete database obtained with 2 stations and 3 channel sensors, during a period of great activity of the Ubinas Volcano.

Furthermore, in the context of the proposed dimensionality reduction technique LC-MDR, we also highlight the following contributions:

- Formulation of two high-order extensions of the Nested PARAFAC Decomposition (NPD), namely Even-Order Nested PARAFAC Decomposition (EONPD) and High-Order Nested PARAFAC Decomposition (HONPD);

- Development of a tensor modeling for the problem of multilinear projection with dimensionality reduction using the EONPD;
- A multilinear dimensionality reduction technique based on the EONPD;
- Application of the LC-MDR in a volcano-seismic database using a full tensor classification framework.

Then, the original contributions regarding the proposed multilinear sampling approach and both SVM-MPS and SVM-MTS techniques, can be highlighted as such:

- The approach of multilinear sampling;
- PARAFAC-based and Tucker-based modifications of the SVM algorithm;
- A Quadratic Programming (QP) formulation of the proposed problems;
- Application of the proposed technique for the classification of logic levels at the output of a Mach-Zehnder Interferometer (MZI).

In addition to the contributions of Chapter 5, in Appendix A, a multilinear sampling formulation for the Support Vector Regression (SVR) technique is proposed. However, due to time issues, this formulation could not be tested.

In the following sections, the applications in which these tensor learning contributions were used are presented.

1.3 Applications for the Proposed Tensor Learning Approaches

In order to to validate the techniques proposed in this thesis, it is important to choose applications that fit the characteristics of the presented approaches. As the techniques are all tensor-based, the datasets to be chosen must be arrangeable in the considered multidimensional structure. Therefore, the proposed tensor learning approaches of this thesis were tested with the following applications:

- Seismic event classification;
- Photonic output classification.

The proposed tensorial framework uses seismic event data as inputs, and its classification results are used as validation. Additionally, the proposed LC-MDR will is tested with the same seismic event data, where dimensionality reduction and feature transformation will be performed. Later on, photonic data classification will be used to validate the proposed SVM-MPS, SVM-MTS methods that exploits the multilinear sampling.

1.3.1 Seismic Event Classification

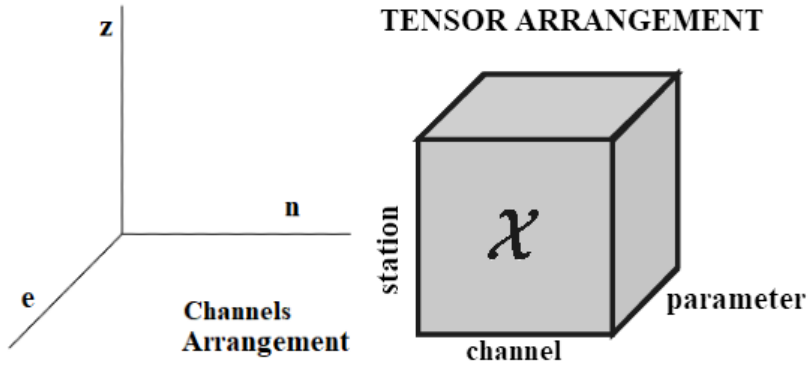
There is significant importance in detecting and classifying seismic events and the reason is simple: volcano-seismic events can be considered a threat to humans and cities located in nearby volcano areas (RAHMAN *et al.*, 2016). For instance, volcanic eruptions such as the ones of the Volcan de Fuego (Guatemala, June 2018), the Stromboli volcano (Italy, July 2019) and more recently, the Semeru volcano (Indonesia, November 2019 and December 2021), showed the catastrophic effects that volcanic eruptions can make. Combined, these eruptions destroyed a large amount of infrastructure and made hundreds of victims.

Due to advances in volcano monitoring, a large amount of seismic data is observed worldwide and the analysis of these time series can be used to predict or detect the eruptive state of volcanoes. Unfortunately, seismic data are still classified manually in many places, which can lead to catastrophic errors and delays in the detection process, reinforcing the importance of developing automatic classification systems (LARA *et al.*, 2020).

Many methods for classification of seismic patterns were proposed over the past several years, as cited in (PEIXOTO *et al.*, 2021) and in (MALFANTE *et al.*, 2018). Surely, supervised learning has impacted the area of seismology in a positive way. For instance, (MALFANTE *et al.*, 2018) focuses on integrated and operational tools dedicated to the automatic analysis of volcano-seismic signals using ML techniques. Moreover, the work of (MALFANTE *et al.*, 2018) outlines many ML techniques and approaches for seismic event classification, which are all vector-based. In (SHIMSHONI; INTRATOR, 1998; SCARPETTA *et al.*, 2005), ML techniques were applied for pattern recognition of volcanic waveforms, in conjunction with an Artificial Neural Network (ANN). In (LARA *et al.*, 2020), an automatic classification system for volcano-seismic events is proposed using the Empirical Mode Decomposition (EMD). Recently, in (CURILEM *et al.*, 2018), deep learning, by means of a Convolutional Neural Network (CNN), was used to classify spectrograms of seismic events from a South American volcano.

Some works have used multilinear algebraic techniques applied to seismic signals. In (VRABIE *et al.*, 2006), a tensor modeling is used for a three-mode system that uses polarization, distance and temporal modes, in seismic event waves separation, by using the HOSVD and unimodal ICA to split the recorded three-mode data into two orthogonal subspaces: the signal and noise subspaces. This decomposition allows the separation and identification of polarized waves. However, as said earlier, there is no work on the literature that is similar to the one proposed in this thesis, which employs a complete tensor-based framework.

Figure 1 – Seismic data tensor arrangements.



Source: Author.

Both the proposed tensorial framework and the LC-MDR technique were tested and validated using seismic data obtained from the Ubinas volcano, in Peru, during a period of great activity in 2009. The reason of such choice is because the seismic data was capable of being organized as a tensor, which could be exploited by tensor learning techniques. Moreover, there is no similar tensor-based approach for seismic event classification, making the proposed framework an alternative to the conventional vector-based approaches of the literature, such as the works of (REYNEN; AUDET, 2017), (KORTSTRÖM *et al.*, 2016), (CURILEM *et al.*, 2009), etc.

As for the used database itself, it was collected from two stations of the Ubinas volcano, located 70 km northeast of the city of Arequipa, in Peru, during a period of great activity in 2009. The data catalog was constructed by experts of the Volcanological Observatory of the Geophysical Institute of Peru (IGP). The data tensors are constructed by exploiting the use of multichannel triaxial sensors, whereas the standard approach of the literature is using only a single channel sensor. Moreover, the database used in this thesis was recorded with sensors that have 3 channels: vertical, east and north, which shown to offer a better representation of the seismic signals in comparison to single-channel sensors (LARA *et al.*, 2020).

In addition, more than one seismic station is used to build the tensor patterns. Hence, the three dimensional feature arrays are constructed as follows *stations* $\times$ *channels* $\times$ *features*. The tensor representation of the data is preserved in the proposed approach, avoiding the earlier mentioned drawbacks of the vectorization process. Figure 1 illustrates the arrangement performed of the seismic data into tensor format.

1.3.2 Photonic Output Classification

All-optical processing is especially essential in systems and networks that want to avoid optoelectronic conversions and therefore need high-speed data reception and transmission. The idea of designing logic gates based on optic devices, such as optical couplers, resonators and interferometers is intendend to solve the problems of optoeletronic conversion.

One of the devices that has been exploited in photonic and optic applications is the fiber-optic MZI (ZETIE *et al.*, 2000). Basically, numerical studies have used the solution of the nonlinear Schrodinger equation to design MZIs capable of obtaining several logic functions (KUMAR *et al.*, 2014; ARAÚJO *et al.*, 2015; CORREIA *et al.*, 2017). Furthermore, optical devices such as the MZI have been used in many computer and engineering applications, such as optical sensors, optical modulators and others (SOUZA *et al.*, 2018; GAYEN *et al.*, 2012).

On the other hand, optical logic gates are a very important part in the development of all-optical communication and optical signal processing networks (ALIPOUR-BANAEI *et al.*, 2017). Indeed, they represent the basic building block of optical devices and networks, where all-optical processing is, in general, essential in systems and networks that want to avoid optoelectronic conversions and exploit high-speed data reception and transmission. Several researches have been done for designing logic gates based on optic devices, such as optical couplers, resonators, interferometers etc (KUMAR *et al.*, 2014; ARAÚJO *et al.*, 2015).

Due to the characteristics and efficiency of the ML techniques, they can be considered as adequate tools to the problems involved in the design of all-optical devices or materials. Indeed, in recent works, multiple ML-based approaches have been applied to the design of photonic devices or structures (MA *et al.*, 2021; LIU *et al.*, 2021).

When ML methods are used to model dynamic systems, such as photonic devices, it is usual to construct a training database in which a scan of the input variables is carried out, i.e. the training database is constructed by varying each input in a certain range at each time. This scan of input of variables usually follows a multilinear sampling structure. This multilinear sampling approach means that the samples were originally obtained obeying a multidimensional structure with multiple dimensions.

For instance, this multidimensional structure for the samples is common in the optics and photonics fields. In (XIE *et al.*, 2020), an arbitrary ratio optical power splitter is designed with data structured in a similar way as described. Moreover, in (KUMAR *et al.*, 2014) and in (ARAÚJO *et al.*, 2015; CORREIA *et al.*, 2017; SOUSA *et al.*, 2014), logic gates are implemented

through MZIs, with sets of parameters and inputs that could exploit the multlinear sampling structures presented. In addition, the works of (MENGU *et al.*, 2022) and (TAHERSIMA *et al.*, 2019) that exploit inverse logic design in ML, perform the input and parameters scan, from MZIs and other optic devices, similar as the multidimensional sampling approach.

Therefore, photonic devices output classification poses as a great application in which the multilinear sampling approach can be used and validated.

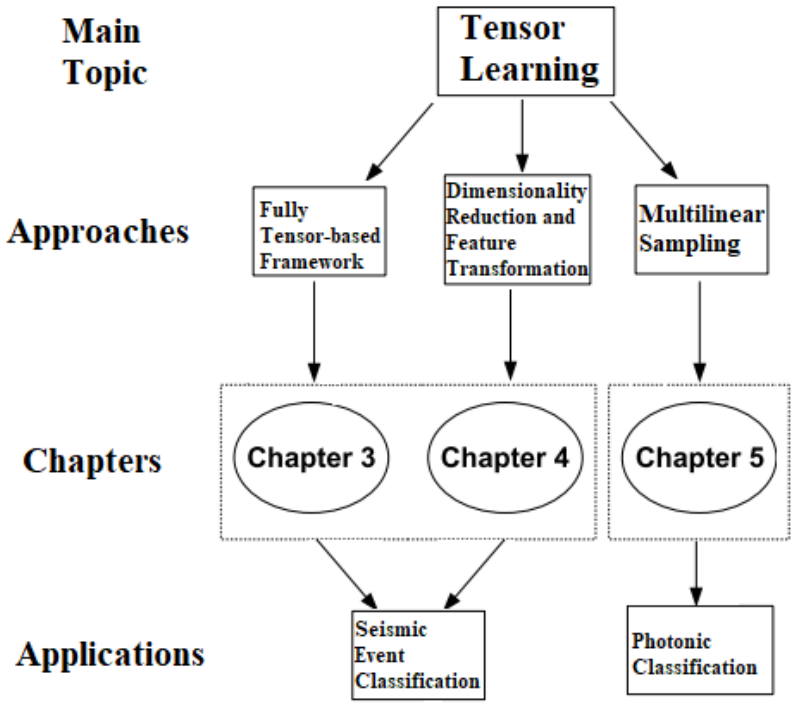
In the present work, a MZI is used to generate photonic data. Inherent to the MZI structure, both input values and both phase deviation values are captured into a 4-th order tensor. One of the outputs of the interferometer is chosen to represent the class tags, either a logic 1 or logic 0, then, the proposed algorithms are used to perform classification of the output of the photonic device.

1.4 Scientific Outputs

Three papers were produced from the content of this thesis (1 published, 2 submitted) and one is in the process of conclusion:

- Peixoto, A.A.T., Fernandes, C.A.R., Lara, P.E.E., Inza, A., Mars, J.I., Metaxian, J.P., Dalla Mura, M. and Malfante, M., 2021. Tensor-based learning framework for automatic multichannel volcano-seismic classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14, pp.4517-4529. This paper refers to the contents of Chapter 3.
- Peixoto, A.A.T., Fernandes, C.A.R., Lara, P.E.E., Inza, A., 2022. Low-Correlation Multilinear Dimensionality Reduction Applied to Volcano-Seismic Classification. Submitted to Multidimensional Systems and Signal Processing, in May 2022. Currently under review. This paper refers to the contents of Chapter 4.
- Peixoto, A.A.T., Fernandes, C.A.R., 2022. Multilinear Sampling in Support Vector Machines for Photonic Data Classification. In review before submission to specialized IEEE Journal. This paper refers to the contents of Chapter 5.
- Peixoto, A.A.T., Fernandes, C.A.R., 2022. Packet Classification using Support Tensor Machines. Submitted to XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais. This paper is a new application of STMs and its content is not included in this thesis. It is attached in Appendix B.

Figure 2 – Links between the main topic, approaches, chapters and applications.



Source: Author.

1.5 Organization

This thesis is organized as follows:

– Chapter 2 is devoted to the presentation of essential theoretical background concepts of this thesis, where tensor algebra, tensor decompositions and other multilinear techniques (High-Order Singular Value Decomposition (HOSVD) and MDFT) are outlined. Later, machine learning concepts, PCA and SVM formulations, tensor learning concepts and tensor-based learning techniques are detailed;

– Chapter 3 presents the proposed supervised tensor-based learning framework for classifying volcano-seismic events. First, seismic event classification is motivated, then, each step of the framework is detailed, and after, the semisc dataset is described. The results presented in this chapter showed the performance of the proposed framework in comparison to other techniques of the literature.

– Chapter 4 presents the proposed multilinear dimensionality reduction technique called LC-MDR. The tensor decompositions EONPD and HONPD are proposed and the LC-MDR is formulated using the EONPD. Results showing the performance of the proposed technique against other dimensionality reduction techniques of the literature are presented at the end

of the chapter;

- Chapter 5 proposes a multilinear sampling approach and two modifications of the SVM classifier (SVM-MPS and SVM-MTS), where two tensorial decompositions are considered for the slack variables, the PARAFAC and Tucker. Then, the modified SVM algorithm is used to classify the photonic outputs obtained from a Mach-Zehnder interferometer. Results presented in the chapter showed better results for the proposed techniques in comparison to other classifiers of the literature, also emphasizing the multilinear structure adopted;
- Chapter 6 summarizes our conclusions and lists some research perspectives in this thesis subject;
- A multilinear sampling formulation for the SVR technique is attached in Appendix A;
- The submitted paper mentioned in Section 1.4 is attached in Appendix B.

This thesis links the topic of tensor learning, proposing theoretical and applied contributions with the adopted approaches, as Figure 2 illustrates.

2 THEORETICAL BACKGROUND

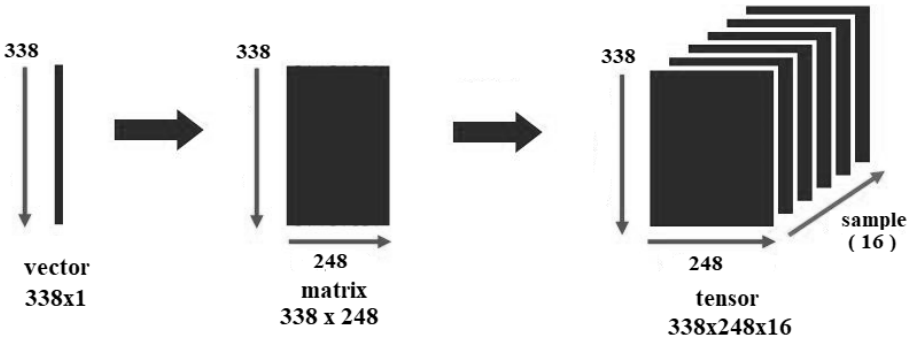
This chapter is devoted to the presentation of essential background concepts of this thesis, where tensor algebra, tensor decompositions, machine learning and tensor learning are detailed. First, it is introduced a background of tensor algebra. Afterwards, tensor representations and basic operations are shown. Then, we outline some tensor decompositions, where we first present the PARAFAC and the nested version of the PARAFAC, the Nested-PARAFAC. Next, the Tucker decomposition is described and, further on, the HOSVD and the MDFT are shown. In the sequence, machine learning concepts are introduced and detailed, where classical techniques, such as the SVM and PCA are described. Finally, tensor learning basics and two important techniques, the MPCA and the STMs, are detailed.

2.1 Tensor Algebra

Multilinear algebra is the algebra of arrays with order higher than two. These high order arrays are called tensors. The theory of tensors is nowadays also known as tensor algebra. The word "tensor" was first introduced in the XIX century (ALMEIDA, 2007) but its use as we know nowadays was only introduced between the 60s and 70s by Kruskal (KRUSKAL, 1977), Richard A. Harshman (HARSHMAN, 1970) and L. R. Tucker (TUCKER, 1966), who were the pioneers on the development of tensor decompositions, analysis and factorizations for third order tensors.

A N-th order tensor is a multilinear mapping. If the space basis associated to the mapping are fixed, then a tensor can be represented by a finite array, or table, of N coordinates. Hence, a N-th order tensor is interpreted by an array whose elements can be accessed by N indices. A tensor can be also called multidimensional array or multi-way array. The notation used in this thesis is presented now: scalars are denoted by lower-case letters (x,y,...), vectors by lower-case boldface letters ($\mathbf{x}$,$\mathbf{y}$,...), matrices by upper-case boldface letters ($\mathbf{X}$,$\mathbf{Y}$,...) and tensors by calligraphic letters ($\mathscr{X}$,$\mathscr{Y}$,...). To retrieve the i-th element of vector $\mathbf{x}$, we use x_i, the element (i,j) of matrix $\mathbf{X}$ is denoted by $[\mathbf{X}]_{i,j}$ or $x_{i,j}$, and, the element $(i_1,...,i_N)$ of the N-th order tensor $\mathscr{X}$ is denoted by $[\mathscr{X}]_{i_1,...,i_N}$ or $x_{i_1,...,i_N}$. The transpose of the matrix $\mathbf{X}$ is denoted by $\mathbf{X}^T$, its Moore-Penrose inverse (pseudoinverse) is denoted by $\mathbf{X}^+$, its complex conjugate is denoted by $\mathbf{X}^*$, the trace operation, which is defined to be the sum of elements on the main diagonal (from the upper left to the lower right), is given by $Tr[\mathbf{X}]$ and $\mathbf{X}_{:,i}$ represents the i-th column of

Figure 3 – A Vector, a matrix and a tensor



Source: Author.

the matrix **X**.

In Fig. 3 we can see a representation of a vector, a matrix and a third order tensor, for illustrative purposes. As a tensor is a multilinear form and has its own associated linear vector space, common linear operations that are valid for matrices can be extended and used for tensors. For instance, we have:

Definition 2.1.1 (Inner product)**.** Let $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N}$ and $\mathscr{Y} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N}$ be N-th order tensors, the inner product between $\mathscr{X}$ and $\mathscr{Y}$ is given by:

$$\langle \mathscr{X}, \mathscr{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \ldots \sum_{i_N=1}^{I_N} x_{i_1,i_2,\ldots,i_N} y_{i_1,i_2,\ldots,i_N}, \tag{2.1}$$

where $\mathscr{X}$ and $\mathscr{Y}$ are said to be orthogonal if $\langle \mathscr{X}, \mathscr{Y} \rangle = 0$.

Definition 2.1.2 (Outer product)**.** Let $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N}$ and $\mathscr{Y} \in \mathbb{R}^{J_1 \times J_2 \times J_3 \times \ldots \times J_M}$ be N-th and M-th order tensors, the outer product between $\mathscr{X}$ and $\mathscr{Y}$ is described as follows:

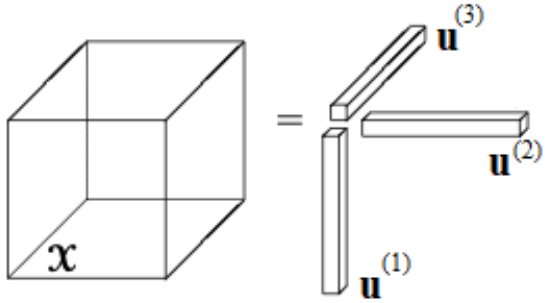
$$[\mathscr{X} \circ \mathscr{Y}]_{i_1,i_2,\ldots,i_N,j_1,j_2,\ldots,j_M} = x_{i_1,i_2,\ldots,i_N} y_{j_1,j_2,\ldots,j_M}, \tag{2.2}$$

where "$\circ$" denotes the outer product. The result of $[\mathscr{X} \circ \mathscr{Y}]$ is a tensor with order equal to the sum of the orders of $\mathscr{X}$ and $\mathscr{Y}$ (a $(N+M)$-th order tensor).

The rank of a tensor is a concept inherited from matrix algebra. An intuitive way to describe the rank of a tensor is as follows:

Definition 2.1.3 (Rank-1 tensor (KOLDA; BADER, 2009))**.** Let $\mathscr{X} \in \mathbb{C}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N}$ be a N-th order tensor. $\mathscr{X}$ is a rank-1 tensor if it can be represented as the outer product of N vectors $\mathbf{u}^{(1)} \in \mathbb{C}^{I_1}$, $\mathbf{u}^{(2)} \in \mathbb{C}^{I_2}$,..., $\mathbf{u}^{(N)} \in \mathbb{C}^{I_N}$, as follows:

Figure 4 – A rank-1 third order tensor



Source: (KOLDA; BADER, 2009).

$$x_{i_1,i_2,\ldots,i_N} = \mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ \ldots \circ \mathbf{u}^{(N)}. \tag{2.3}$$

The vectors $\mathbf{u}^{(N)}$ are so called the components of $\mathscr{X}$. As an example, a rank-1 matrix is given by the outer product of two vectors. Figure 4 illustrates a rank-1 third order tensor.

Definition 2.1.4 (Rank of a tensor (KOLDA; BADER, 2009))**.** The rank of a tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N}$, denoted by R, is defined as the minimum number of rank-1 components that gives $\mathscr{X}$ as a linear combination.

Definition 2.1.5 (Frobenius norm)**.** The Frobenius norm of an N-th order tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N}$ is defined as:

$$||\mathscr{X}||_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} |x_{i_1,i_2,\ldots,i_N}|^2}. \tag{2.4}$$

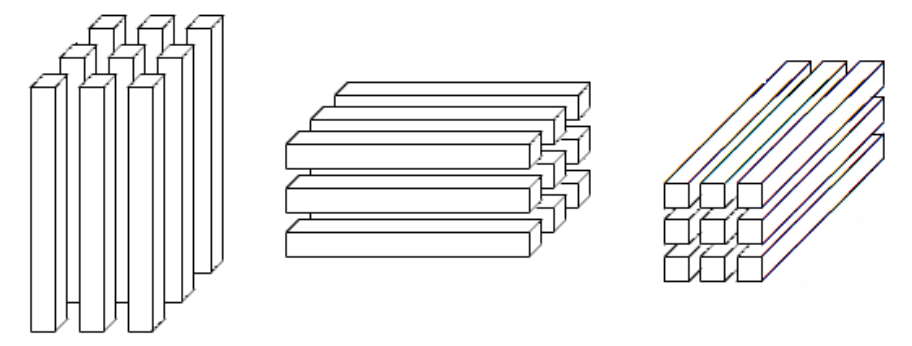
The Frobenius norm can be also expressed in terms of the inner product $||\mathscr{X}||_F = \sqrt{\langle \mathscr{X}, \mathscr{X} \rangle}$.

Definition 2.1.6 (Tensor fiber (KOLDA; BADER, 2009))**.** The mode-n tensor fiber of a N-th order tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N}$ is defined as the vector formed by fixing every index but the i_n-th.

Furthermore, considering a third order tensor $\mathscr{Y} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, its mode-1, mode-2, and mode-3 fibers are given, respectively, by $\mathbf{y}_{\cdot i_2 i_3} \in \mathbb{R}^{I_1}$, $\mathbf{y}_{i_1 \cdot i_3} \in \mathbb{R}^{I_2}$ and $\mathbf{y}_{i_1 i_2 \cdot} \in \mathbb{R}^{I_3}$, where "$\cdot$" denotes the varying index. In Fig. 5 we can see an illustration of tensor fibers in different modes of a third order tensor.

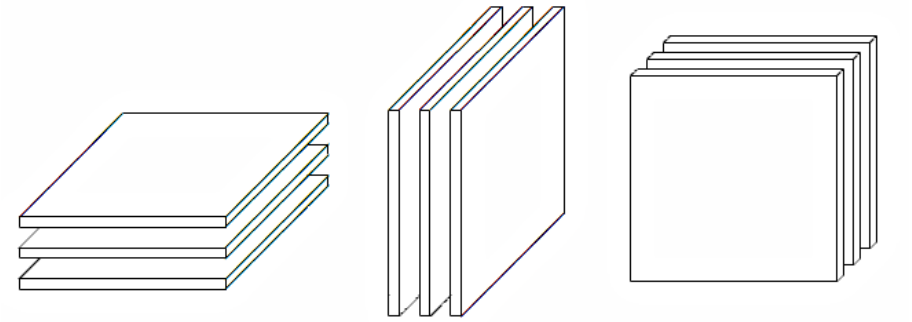
We may also define the tensor slices, which are two-dimensional sections of a tensor, defined by fixing all but two indices (KOLDA; BADER, 2009). As Figure 6 shows, from left to

Figure 5 – View of mode-1, mode-2 and mode-3 fibers of a third order tensor.



Source: (KOLDA; BADER, 2009).

Figure 6 – View of mode-1, mode-2 and mode-3 slices of a third order tensor.



Source: (KOLDA; BADER, 2009).

right we have the mode-1 (or first-mode) slices, mode-2 (or second-mode) slices and mode-3 (or third-mode) slices, denoted respectively by $\mathbf{Y}_{i_1..}$, $\mathbf{Y}_{.i_2.}$ and $\mathbf{Y}_{..i_3}$.

Definition 2.1.7 (Tensor unfolding (KOLDA; BADER, 2009))**.** The mode-n unfolding, also known as matricization, of a N-th order tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N}$ yields a matrix $\mathbf{X}^{[n]} \in \mathbb{R}^{I_n \times I_1 I_2 \ldots I_{n-1} I_{n+1} \ldots I_N}$ whose elements are obtained from the tensor $\mathscr{X}$ in the following way:

$$[\mathbf{X}^{[n]}]_{i_n, j} = [\mathscr{X}]_{i_1, \ldots, i_N}, \quad j = 1 + \sum_{\substack{u=1 \\ u \neq n}}^{N} (i_u - 1) \prod_{\substack{v=1 \\ v \neq n}}^{u-1} I_v.$$

Hence, as an example, the unfoldings of an arbitrary third order tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ are given by $\mathbf{X}^{[1]} \in \mathbb{R}^{I_1 \times I_2 I_3}$, $\mathbf{X}^{[2]} \in \mathbb{R}^{I_2 \times I_1 I_3}$ and $\mathbf{X}^{[3]} \in \mathbb{R}^{I_3 \times I_1 I_2}$. We may also note that the mode-n matrix unfolding can be seen as the concatenation of the mode-n fibers along the matrix columns. The mode-n unfolding matrices of a tensor can also be obtained by stacking the tensor slices.

Definition 2.1.8 (Vectorization)**.** The operation $\text{vec}(\,) : \mathbb{R}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N} \rightarrow \mathbb{R}^{I_1 I_2 I_3 \ldots I_N}$ denote the vectorization operator, which transforms a tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N}$ into a vector $\text{vec}(\mathscr{X}) \in$

$\mathbb{R}^{I_1 I_2 I_3 \ldots I_N}$ with components defined as:

$$[\text{vec}(\mathscr{X})]_j = [\mathscr{X}]_{i_1,i_2,\ldots,i_N}, \quad j = i_1 + \sum_{n=2}^{N}(i_n - 1)\prod_{v=1}^{n-1} I_v. \tag{2.5}$$

The inverse process (turning a vector into a tensor) is called "tensorization".

Definition 2.1.9 (Mode-*n* product (KOLDA; BADER, 2009))**.** The mode-*n* product between a *N*-th order tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N}$ and a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ is defined as:

$$[\mathscr{X} \times_n \mathbf{U}]_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} x_{i_1,i_2,\ldots,i_N} u_{j,i_n}, \quad j \in 1,\ldots,J, \tag{2.6}$$

with "$\times_n$" being the mode-*n* product operator. The mode-*n* product is a good way for representing linear transformations involving tensors.

Definition 2.1.10 (Contraction)**.** The contraction operation between two tensors $\mathscr{X}$ and $\mathscr{Y}$, that share a common dimension ($I_p = J_q = K$, with $1 \leq p \leq N$ and $1 \leq q \leq M$), is denoted by $\mathscr{Z} = \mathscr{X} \underset{k}{*} \mathscr{Y}$ and is defined as the following sum over the common mode ($i_p = j_q = k$):

$$z_{i_1,\ldots,i_{p-1},\ldots,j_1,\ldots,j_{q-1},\ldots,j_M,\ldots,i_N} = \sum_{k=1}^{K} x_{i_1,\ldots,i_{p-1},k,\ldots,i_N} y_{j_1,\ldots,j_{q-1},k,\ldots,j_M}. \tag{2.7}$$

Next, some matrix operations that will be used further on in this thesis are presented, beginning with an alternative equation for the inner product, in matrix notation, using the vectorization definition:

$$\langle \mathscr{X}, \mathscr{X} \rangle = Tr[\mathbf{X}^{(n)}(\mathbf{X}^{(n)})^T] = vec(\mathbf{X}^{(n)})^T vec(\mathbf{X}^{(n)}). \tag{2.8}$$

Similarly:

$$\langle \mathscr{X}, \mathscr{Y} \rangle = Tr[\mathbf{X}^{(n)}(\mathbf{Y}^{(n)})^T] = vec(\mathbf{X}^{(n)})^T vec(\mathbf{Y}^{(n)}). \tag{2.9}$$

Now, two matrix products are presented, the Kronecker and the Khatri-Rao matrix product. Both products are important in the next sections and chapters.

Definition 2.1.11 (Kronecker product)**.** The Kronecker product between matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$ is given by (SIDIROPOULOS *et al.*, 2017):

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \ldots & a_{1,J}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{I,1}\mathbf{B} & \ldots & a_{I,J}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{IK \times JL}. \tag{2.10}$$

Definition 2.1.12 (Khatri-Rao product)**.** The Khatri-Rao (column-wise Kronecker) product between matrices with the same number of columns $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$ is given by (SIDIROPOULOS *et al.*, 2017):

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{A}_{:,1} \otimes \mathbf{B}_{:,1} & \dots & \mathbf{A}_{:,K} \otimes \mathbf{B}_{:,K} \end{bmatrix} \in \mathbb{R}^{IJ \times K}. \tag{2.11}$$

Finally, given N matrices $\mathbf{A}^{(1)}, ..., \mathbf{A}^{(N)}$, short notations for the Khatri-Rao and Kronecker products are given by:

$$\mathbf{A}_{\odot} = \mathbf{A}^{(N)} \odot \ldots \odot \mathbf{A}^{(2)} \odot \mathbf{A}^{(1)} \tag{2.12}$$

and

$$\mathbf{A}_{\otimes} = \mathbf{A}^{(N)} \otimes \ldots \otimes \mathbf{A}^{(2)} \otimes \mathbf{A}^{(1)}. \tag{2.13}$$

Additionally, for the Khatri-Rao and Kronecker products between N-1 of these matrices (all but the n-th matrix), the short notations are respectively given by:

$$\mathbf{A}^{(n)}_{\odot} = \mathbf{A}^{(N)} \odot \ldots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \ldots \odot \mathbf{A}^{(1)} \tag{2.14}$$

and

$$\mathbf{A}^{(n)}_{\otimes} = \mathbf{A}^{(N)} \otimes \ldots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \ldots \otimes \mathbf{A}^{(1)}. \tag{2.15}$$

2.2 Tensor Decompositions and other Multidimensional Techniques

In the last section, it was presented an introduction to multilinear algebra. Based on these concepts, we present now the tensor decompositions that will be used in the rest of this thesis. These decompositions, also known as multi-way factor analysis, can be viewed, depending on the approach and point of view, as generalizations of PCA or Singular Value Decomposition (SVD) to higher order arrays. In general, the decompositions of high order arrays can be viewed as generalizations of matrix decompositions.

A multidimensional variable can be interpreted as a tensor, so, the analysis of a tensor in terms of its decomposed factors is useful in problems where a multilinear junction of different factors must be identified or separated from the measured data. In this context, a tensor decomposition of an observed variable, can separate the data originated from different sources, allowing the development of powerful signal processing and learning tools.

In the following, the PARAFAC decomposition of third order and N-th order tensors (or three-way and N-way arrays) is presented and, after, the Nested-PARAFAC is shown, since both are important to the understanding of the applications and methods encountered in this thesis. Furthermore, the Tucker decomposition and the HOSVD are described.

2.2.1 Parallel Factor Analysis (PARAFAC)

The PARAFAC decomposition was first presented by (HITCHCOCK, 1927) and further developed by Harshman in (HARSHMAN, 1970) and Carroll & Chang in (CARROLL; CHANG, 1970), in different works in 1970. It was referred in Carroll & Chang's work as Canonical Decomposition, abbreviated to CANDECOMP. The analysis that Harshamn showed in (HARSHMAN, 1970) and in (HARSHMAN; LUNDY, 1984) is called PARAFAC decomposition, which has been extensively studied in the literature and applied on several areas (KOLDA; BADER, 2009). The PARAFAC decomposition can also be referred by the acronyms CANDECOMP and CP.

The use of tensor models and decompositions, more precisely the PARAFAC, was found to be useful in ICA applications (LATHAUWER *et al.*, 2000a; CHOI *et al.*, 2005). ICA is a special case of blind source separation and is defined as a computational method used to separate a multivariate signal into additive independent subcomponents, contrasting to the fact that the PARAFAC decomposition can describe the basic structure of high order cumulants of multivariate data (BRO, 1997), (HARSHMAN; LUNDY, 1984), thus, showing that tensor decompositions can be an interesting way to deal with multidimensional data (LATTIN *et al.*, 2003).

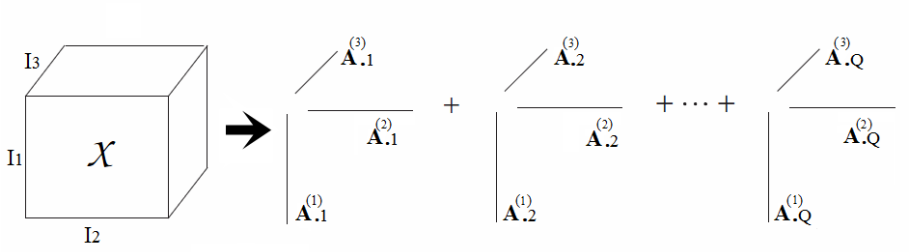
The formulation of the PARAFAC decomposition of an arbitrary third order tensor $\mathscr{X} \in \mathbb{C}^{I_1 \times I_2 \times I_3}$ can be expressed, in scalar notation, as:

$$x_{i_1,i_2,i_3} = \sum_{q=1}^{Q} a^{(1)}_{i_1,q} a^{(2)}_{i_2,q} a^{(3)}_{i_3,q}, \tag{2.16}$$

where Q is the rank of the PARAFAC decomposition and $a^{(1)}_{i_1,q}$, $a^{(2)}_{i_2,q}$ and $a^{(3)}_{i_3,q}$ are the elements of the three factor matrices $\mathbf{A}^{(1)} \in \mathbb{R}^{I_1 \times Q}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{I_2 \times Q}$ and $\mathbf{A}^{(3)} \in \mathbb{R}^{I_3 \times Q}$.

The tensor $\mathscr{X}$ can also be expressed in the form of outer products between the factor

Figure 7 – PARAFAC decomposition of a third-order tensor.



Source: (PEIXOTO, 2017).

matrices, as such:

$$\mathscr{X} = \sum_{q=1}^{Q} \mathbf{A}^{(1)}_{.q} \circ \mathbf{A}^{(2)}_{.q} \circ \mathbf{A}^{(3)}_{.q}, \tag{2.17}$$

It can be viewed from (2.17) that $\mathscr{X}$ is a sum of outer products known as "trilinear model" or "trilinear decomposition". Fig. 7 illustrates $\mathscr{X}$ as sum of Q outer products.

The mode-n unfolding of a third order PARAFAC decomposition can be expressed as:

$$\mathbf{X}^{[n]} = \mathbf{A}^{(n)}[\mathbf{A}^{(n)}_{\odot}]^T \quad \in \mathbb{C}^{I_n \times I_1 I_2 I_3 / I_n}, \tag{2.18}$$

for n= 1,2,3, where $\mathbf{A}^{(n)}_{\odot} \in \mathbb{C}^{I_1 I_2 I_3 / I_n \times Q}$ is given by (2.14). For instance, $\mathbf{X}^{[1]} = \mathbf{A}^{(1)}[\mathbf{A}^{(3)} \odot \mathbf{A}^{(2)}]^T \in \mathbb{C}^{I_1 \times I_3 I_2}$.

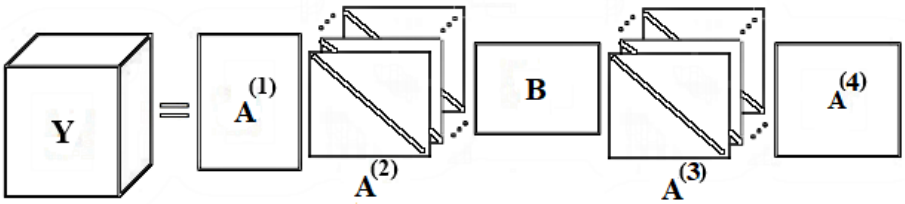
The generalization of the PARAFAC decomposition for N-th order tensors, such as $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times \ldots \times I_N}$, is given by:

$$x_{i_1,i_2,i_3,\ldots,i_n} = \sum_{q=1}^{Q} a^{(1)}_{i_1,q} a^{(2)}_{i_2,q} \ldots a^{(N)}_{i_N,q}, \tag{2.19}$$

where $a^{(n)}_{i_n,q}$ is an element of the factor matrix $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times Q}$, with n = 1,2,...,N and i_n = 1,2,...,I_N.

Alongside the advantages of representing and analyzing multidimensional data, another important feature of the PARAFAC is its uniqueness feature. The PARARAC decomposition of tensors is unique up to permutation and scaling indeterminacy, under certain circumstances (KRUSKAL, 1977; STEGEMAN; SIDIROPOULOS, 2007). An uniqueness proof was made by Kruskal in (KRUSKAL, 1977). Also, a generalization of the uniqueness results of (KRUSKAL, 1977) to tensors of any order was given in (SIDIROPOULOS; BRO, 2000) by N. Sidiropoulos and R. Bro, who also applied tensor models to telecommunications and provided the uniqueness conditions to complex tensor models in (SIDIROPOULOS *et al.*,

Figure 8 – Block Diagram of the NPD.



Source: (PEIXOTO; FERNANDES, 2019).

2000). Furthermore, in (LATHAUWER, 2006), a great overview of the PARAFAC uniqueness properties are outlined.

The PARAFAC decomposition of tensors with rank greater than 1 can be unique up to scaling and permutation of factors, unlike matrix decompositions which are mostly not unique for ranks less than 1.

2.2.2 Nested-PARAFAC decomposition

The Nested PARAFAC Decomposition (NPD) of a 4-th order tensor $\mathscr{Y} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$ is defined as (ALMEIDA; FAVIER, 2013; PEIXOTO; FERNANDES, 2019):

$$y_{i_1,i_2,i_3,i_4} = \sum_{q_1=1}^{Q_1} \sum_{q_2=1}^{Q_2} a^{(1)}_{i_1,q_1} a^{(2)}_{i_2,q_1} b_{q_1,q_2} a^{(3)}_{i_3,q_2} a^{(4)}_{i_4,q_2}, \tag{2.20}$$

where $\mathbf{A}^{(1)} \in \mathbb{R}^{I_1 \times Q_1}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{I_2 \times Q_1}$, $\mathbf{B} \in \mathbb{C}^{Q_1 \times Q_2}$, $\mathbf{A}^{(3)} \in \mathbb{R}^{I_3 \times Q_2}$ and $\mathbf{A}^{(4)} \in \mathbb{R}^{I_4 \times Q_2}$. Fig. 8 shows a block diagram of the NPD. By concatenating the indices i_3 and i_4 in the following way: $y_{i_1,i_2,j_1} = y_{i_1,i_2,i_3,i_4}$, where $j_1 = (i_3 - 1)I_4 + i_4$, for $1 \leq j_1 \leq J_1$, with $J_1 = I_3 I_4$, eq. (2.20) can be rewritten as:

$$y_{i_1,i_2,j_1} = \sum_{q_1=1}^{Q_1} a^{(1)}_{i_1,q_1} a^{(2)}_{i_2,q_1} w^{[1]}_{j_1,q_1}, \tag{2.21}$$

where $w^{[1]}_{j_1,q_1}$ is an element of matrix $\mathbf{W}^{[1]} \in \mathbb{R}^{J_1 \times Q_1}$, which is the mode-1 unfolded matrix of the tensor $\mathscr{W} \in \mathbb{R}^{Q_1 \times I_3 \times I_4}$, defined by:

$$w^{[1]}_{q_1,i_3,i_4} = \sum_{q_2=1}^{Q_2} b_{q_1,q_2} a^{(3)}_{i_3,q_2} a^{(4)}_{i_4,q_2}. \tag{2.22}$$

It can be concluded from (2.21) and (2.22) that $\mathscr{Y} \in \mathbb{R}^{I_1 \times I_2 \times J_1}$ can be viewed as the nesting of two PARAFAC decompositions. Indeed, $\mathscr{Y}$ follows a PARAFAC model with factor matrices $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$ and $\mathbf{W}^{[1]}$, with $\mathbf{W}^{[1]}$ being an unfolded matrix of a PARAFAC tensor with factors $\mathbf{B}$, $\mathbf{A}^{(3)}$ and $\mathbf{A}^{(4)}$. The mode-1 unfolding of $\mathscr{Y}$ can be obtained as follows:

$$\mathbf{Y}^{[1]} = \mathbf{A}^{(1)}[\mathbf{W}^{[1]} \odot \mathbf{A}^{(2)}]^T \in \mathbb{C}^{I_1 \times J_1 I_2}. \tag{2.23}$$

Other unfoldings of $\mathscr{Y}$ can be obtained similarly as in (2.23).

The nesting property can also be expressed differently. In a similar manner, $\mathscr{Y}$ can be expressed as a PARAFAC decomposition with factor matrices $\mathbf{U}^{[2]}$, $\mathbf{A}^{(3)}$ and $\mathbf{A}^{(4)}$, with $\mathbf{U}^{[2]} \in \mathbb{R}^{J_2 \times Q_2}$ being an unfolded matrix of a PARAFAC tensor with factors $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$ and $\mathbf{B}$, where $J_2 = I_1 I_2$.

Finally, the NPD can also be expressed using tensor notation, as a double contraction, as follows:

$$\mathscr{Y} = \mathbf{B} \underset{q_1}{*} \mathscr{R}^{(1)} \underset{q_2}{*} \mathscr{R}^{(2)}, \tag{2.24}$$

where $\mathscr{R}^{(1)} \in \mathbb{C}^{I_1 \times I_2 \times Q_1}$ and $\mathscr{R}^{(2)} \in \mathbb{C}^{I_3 \times I_4 \times Q_2}$ are given by $r^{(1)}_{i_1,i_2,q_1} = a^{(1)}_{i_1,q_1} a^{(2)}_{i_2,q_1}$ and $r^{(2)}_{i_3,i_4,q_2} = a^{(3)}_{i_3,q_2} a^{(4)}_{i_4,q_2}$, respectively. Note that the factor matrix $\mathbf{B}$ interacts with the two PARAFAC tensors $\mathscr{R}^{(1)}$ and $\mathscr{R}^{(2)}$ by means of the contraction operation.

2.2.3 Tucker Decomposition

The Tucker decomposition, proposed by L. Tucker in 1966 (TUCKER, 1966), decomposes a tensor into a set of matrices that interact with a core tensor. It is a more general and flexible model than the PARAFAC. Indeed, the PARAFAC decomposition is a particular case of the Tucker model when the core tensor is superdiagonal, with all dimensions equal to the rank Q.

Then, for a third order tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the Tucker decomposition can be expressed as:

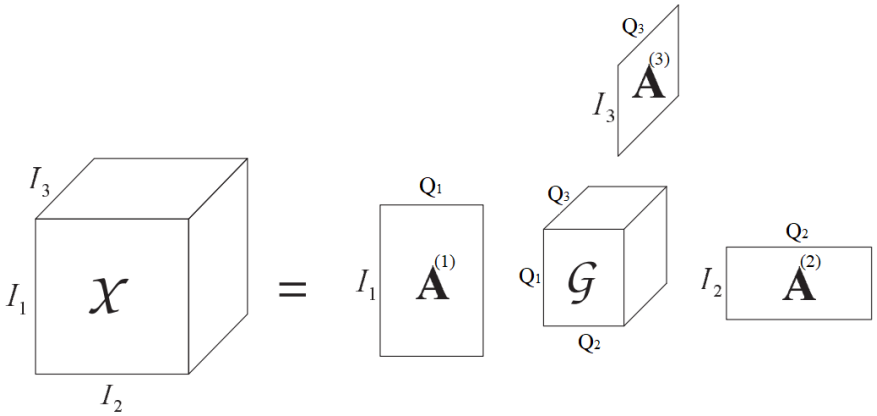
$$\mathscr{X} = \mathscr{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)}, \tag{2.25}$$

where $\mathbf{A}^{(1)} \in \mathbb{R}^{I_1 \times Q_1}$, $\mathbf{A}^{(2)} \in \mathbb{C}^{I_2 \times Q_2}$ and $\mathbf{A}^{(3)} \in \mathbb{R}^{I_3 \times Q_3}$ are the three factor matrices of the decomposition, whereas $\mathscr{G} \in \mathbb{R}^{Q_1 \times Q_2 \times Q_3}$ is the core tensor. Figure 9 illustrates the Tucker decomposition of a third order tensor.

In scalar form, (2.25) is given as such:

$$x_{i_1,i_2,i_3} = \sum_{q_1=1}^{Q_1} \sum_{q_2=1}^{Q_2} \sum_{q_3=1}^{Q_3} g_{q_1,q_2,q_3} a^{(1)}_{i_1,q_1} a^{(2)}_{i_2,q_2} a^{(3)}_{i_3,q_3}, \tag{2.26}$$

Figure 9 – Tucker decomposition of a third-order tensor.



Source: (PEIXOTO; FERNANDES, 2019).

where $a^{(1)}_{i_1,q_1}$, $a^{(2)}_{i_2,q_2}$, and $a^{(3)}_{i_3,q_3}$, are elements of the matrices $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$, with g_{q_1,q_2,q_3} being an element of the core tensor.

The mode-n unfolding of a third-order Tucker decomposition can be expressed as:

$$\mathbf{X}^{[n]} = \mathbf{A}^{(n)}\mathbf{G}^{[n]}[\mathbf{A}^{(n)}_{\otimes}]^T \quad \in \mathbb{C}^{I_n \times I_1 I_2 I_3 / I_n}, \tag{2.27}$$

for n = 1,2,3, where $\mathbf{A}^{(n)}_{\otimes} \in \mathbb{R}^{I_1 I_2 I_3 / I_n \times Q_1 Q_2 Q_3 / Q_n}$ is given by (2.15) and $\mathbf{G}^{[n]} \in \mathbb{C}^{Q_n \times Q_1 Q_2 Q_3 / Q_n}$ is the mode-n unfolding of $\mathscr{G}$.

Initially described as a three-mode extension of the PARAFAC and SVD, the Tucker decomposition may actually be generalized to higher mode analysis. The Tucker decomposition for a N-th order array is given by:

$$x_{p_1,p_2,\ldots,p_N} = \sum_{q_1=1}^{Q_1} \sum_{q_2=1}^{Q_2} \cdots \sum_{q_N=1}^{Q_N} g_{q_1,q_2,\ldots,q_N} a^{(1)}_{p_1,q_1} a^{(2)}_{p_2,q_2} \ldots a^{(N)}_{p_N,q_N}, \tag{2.28}$$

where $a^{(1)}_{p_1,q_1}$, $a^{(2)}_{p_2,q_2}$, ..., $a^{(N)}_{p_N,q_N}$, are elements of the matrix factors of the Tucker decomposition $\mathbf{A}^{(1)} \in \mathbb{R}^{P_1 \times Q_1}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{P_2 \times Q_2}$, ..., $\mathbf{A}^{(N)} \in \mathbb{R}^{P_N \times Q_N}$, and $g_{q_1,q_2,\ldots,q_N}$ forms the core tensor of the Tucker decomposition $\mathscr{G} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}$. In tensor notation, we have (2.28) written as:

$$\mathscr{X} = \mathscr{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times \ldots \times_N \mathbf{A}^{(N)}. \tag{2.29}$$

2.2.4 High-Order Singular Value Decomposition (HOSVD)

The HOSVD, also known as Multinear SVD, is one of the most important tools used in multidimensional data processing, commonly applied to the extraction of relevant

information from multi-way arrays (LATHAUWER *et al.*, 2000b; LATHAUWER *et al.*, 1994; LATHAUWER; VANDEWALLE, 2004). For arrays with dimensions bigger than two, the SVD cannot be used unless the matricization is applied on the data, thus, the HOSVD generalizes the SVD for tensors and can be viewed as a special case of the Tucker decomposition with orthogonal factor matrices.

Given a N-th order tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$, the HOSVD is given by (LATHAUWER *et al.*, 2000b):

$$\mathscr{X} = \mathscr{S} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \ldots \times_N \mathbf{U}^{(N)}, \tag{2.30}$$

where $\mathbf{U}^{(n)} \in \mathbb{C}^{I_n \times I_n}$, for $n = 1, \ldots, N$, is a unitary matrix and $\mathscr{S} \in \mathbb{C}^{I_1 \times \ldots \times I_N}$ is the core tensor. The matrix $\mathbf{U}^{(n)}$ is calculated as the left singular matrix of the transpose of the mode-n unfolded matrix $\mathbf{X}^{(n)} \in \mathbb{C}^{I_1 \ldots I_{n-1} I_{n+1} \ldots I_N \times I_n}$ of the tensor $\mathscr{X}$. A compact HOSVD can also be defined with dimensionality reduction being carried out by neglecting the smallest singular values. In this case, we have $\mathbf{U}^{(n)} \in \mathbb{C}^{I_n \times R_n}$ and $\mathscr{S} \in \mathbb{C}^{R_1 \times \ldots \times R_N}$, where $R_n < I_n$ is the number of singular values used in the n^{th} mode, for $n = 1, \ldots, N$.

Contrarily to the SVD, whose singular value matrix is diagonal, the core tensor $\mathscr{S}$ of the HOSVD is not diagonal. In fact, the tensor $\mathscr{S}$ is all-orthogonal and ordered (LATHAUWER *et al.*, 2000b). This difference between the SVD and HOSVD leads to a significant difference in the orthogonality of the transformed data when dimensionality reduction is carried out by the PCA and HOSVD-based methods.

2.2.5 Multidimensional Discrete Fourier Transform (MDFT)

In mathematical analysis and applications, multidimensional transforms are often used to analyze the frequency content of signals in a domain of two or more dimensions. The multidimensional DFT (MDFT) of a multilinear array $\mathscr{X} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_P}$ is a function of P discrete variables, and defined by complex array $x_{k_1,k_2,\ldots,k_P}$ (GUESSOUM; MERSEREAU, 1986):

$$x_{k_1,k_2,\ldots,k_P} = \sum_{n_1=0}^{N_1-1} \left(e^{\frac{-j2\pi}{N_1} k_1 n_1} \left(\sum_{n_2=0}^{N_2-1} e^{\frac{-j2\pi}{N_2} k_2 n_2} \ldots \left(\sum_{n_P=0}^{N_P-1} e^{\frac{-j2\pi}{N_2} k_P n_P} x_{n_1,n_2,\ldots,n_P} \right) \right) \right), \tag{2.31}$$

with n_1 = 1,...,N_1, n_2 = 1,...,N_2, ..., n_P = 1,...,N_P. Applications of the MDFT are vast, such as in (TSUI *et al.*, 2008) and in (KARMAKAR *et al.*, 2021). Additionally, the MDFT can be

computed using Fast Fourier Transform (FFT) methods (TOLIMIERI *et al.*, 2012; GUESSOUM; MERSEREAU, 1986).

2.3 Machine Learning Basics

Machine learning may be defined as the programming of computers to optimize a performance criterion, generally using data examples or past experience, in order to build systems that can adapt to their environments and learn from their experience. Additionally, ML is most useful in cases where it is not possible to directly write a computer program to solve a given problem, needing example or experience. Indeed, one case where learning is necessary is when human expertise does not exist, when humans are unable to explain their expertise, when the problem to be solved changes in time or depends on the particular environment (ALPAYDIN, 2020). Hence, it is generally preferable to have general purpose systems that can adapt to their circumstances, rather than explicitly writing a different program for each special circumstance.

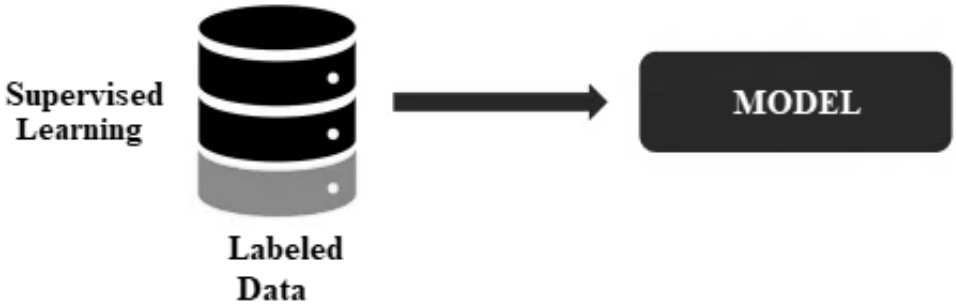
Progress in ML has been driven over the years by the development of learning algorithms, such as ANNs and its variations (MITCHELL, 1997; LIANG *et al.*, 2018), Genetic Algorithms (HOLLAND, 1992), discriminant and component analysis (THARWAT *et al.*, 2017; WOLD *et al.*, 1987), etc. More recently, the development of approaches such as: glsGAN (CRESWELL *et al.*, 2018), Deep Learning (DENG; YU, 2014), Natural Language Processing (NLP) (CHOWDHARY, 2020), Cognitive Computing (GUPTA *et al.*, 2018), Tensorflow (PANG *et al.*, 2020), among others, have put ML into the spotlight of research topics.

In particular, ML is separated into several types regarding the learning processes, which are: supervised, semi-supervised, un-supervised, and reinforcement learning. In this thesis we will be focusing on the first type of learning, the supervised, but the other three are very well documented in the literature (PARDO; SBERVEGLIERI, 2002), (ALPAYDIN, 2020), (JORDAN; MITCHELL, 2015).

In fact, supervised learning is the task of learning a function that maps an input to an output based on example input-output pairs (ZHU; GOLDBERG, 2009). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. Figure 10 illustrates the process of building a supervised model based on labeled data.

Correspondingly, an important branch of machine learning and computational intelligence is classification, which is the process of categorizing information, where ideas, objects or

Figure 10 – Supervised Learning.



Source: Author.

data are recognized, differentiated, understood and then, separated in different tags (PARDO; SBERVEGLIERI, 2002), (ALPAYDIN, 2020). Data classification can be achieved with methods aimed to determine whether or not the data contains specific information, feature, or behavior, which permits the identification of the correct class (KULKARNI *et al.*, 1998; HART *et al.*, 2000).

A well-known learning algorithm that has been widely used on various classification problems is the SVM (VAPNIK, 2013), (VAPNIK, 1999), mostly employed to solve two-class problems (MATHUR; FOODY, 2008), but also used on multi-class solutions (FOODY; MATHUR, 2004). Therefore, it has been widely used for pattern classification and regression problems (BURGES, 1998). For instance, in (JIA *et al.*, 2007), the SVM was used in fingerprint image matching, obtaining excellent results, whereas in (BEGG *et al.*, 2005), gait recognition was achieved by means of SVM. Another example is the work of (LI *et al.*, 2006), which employs SVM in image processing, thus highlighting the SVM as a versatile classifier, as said earlier, being capable of handling a multitude of applications and data. Also, the work of (STEINWART, 2005) showed very consistent classification results with the SVM method against other techniques of the literature.

In this thesis, we focus on the SVM classifier because of its advantages over other techniques. Among these advantages of the SVM, we highlight its versatility, where it can be used for data such as images, text, audio etc (CERVANTES *et al.*, 2020). Indeed, it can be used for data that is not regularly distributed and have unknown distribution. In addition, The SVM provides a very useful technique within it, known as kernel, and by the application of an associated kernel function, complex problems can be solved.

Likewise, the SVM generally performs well when there is a clear indication of separation between classes. Furthermore, the SVM can be used when total number of samples is less than the number of dimensions and performs well in terms of memory. SVM has a nature of

convex optimization which is very helpful as we are assured of optimality in results. It is also important to note that these SVM advantages are carried over to its multilinear extensions, such as the STMs, which will be used in the next chapters of this thesis.

Still in the topic, the literature is rich in methods for data classification, apart from the above mentioned SVM, we can cite: Logistic Regression, Naive-Bayes, *k*-Nearest Neighbors (KNN) or *k*-NN, Decision Trees, ANNs and Genetic Algorithms. Starting with the Logistic Regression method, this technique is capable of classifying data based on calculated scores to predict the target class, like a linear classifier, bringing the relation between categorical dependent variables and the independent variables (NICK; CAMPBELL, 2007). Next, the Naive-Bayes is a classifier that uses the Bayes Theorem. It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class, where the class with the highest probability is considered as the most likely class (YANG, 2018).

The *k*-NN, in turn, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another (JIANG *et al.*, 2007). The decision tree methods are non-parametric supervised learning algorithms, which are utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes, where the leaf nodes represent all the possible outcomes within the dataset (MYLES *et al.*, 2004).

The ANN, on the other hand, is a family of learning techniques, which are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network (LIAO; WEN, 2007). Neural networks rely on training data to learn and improve their accuracy over time. Lastly, the Genetic Algorithms are a new category of evolutionary techniques that uses elements of evolutionary processes to solve search and optimization problems, iteratively improving a large number of possible solutions and combining them with each other (TANG *et al.*, 1996).

Moving on from classification, the PCA is a standard tool in modern data analysis, in diverse fields, from neuroscience (HAN *et al.*, 2018) to computer graphics (YUNQI *et al.*,

2009) because it is a simple, non-parametric method for extracting relevant information from confusing data sets. With minimal effort PCA provides a bridge for how to reduce a complex data set to a lower dimension, revealing the sometimes hidden, simplified structures that often underlie it (SHLENS, 2014).

The central idea of PCA is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. This is achieved by transforming correlated variables into a new set of variables, the principal components (PCs), which are uncorrelated, and are ordered so that the first few retain most of the variation present in all of the original variables.

In this thesis, the aforementioned central idea of the PCA is used to reduce the dimensionality of datasets, reducing the number of features while retaining maximum information. Following are some reasons for performing dimensionality reduction: i) dimensionality reduction helps in data compression, and hence reduced storage space, ii) it reduces computation time, iii) it removes redundant features. Additionally, the PCA helps separating the data, allowing better visualization.

Next, the formulations for both SVM and PCA are described and commented.

2.3.1 Formulations of the Support Vector Machines (SVM)

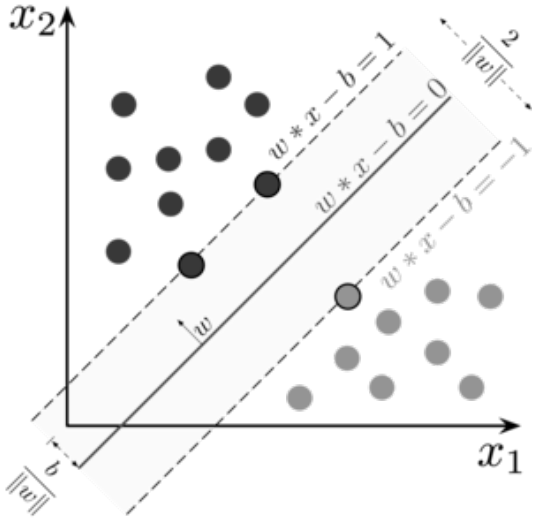
The SVM is a data classification technique developed in the 90's at the AT&T Bell Laboratories by (CORTES; VAPNIK, 1995). SVM algorithms are based on the idea of structural risk minimization (SRM) and it gave rise to new ways of training polynomial, neural networks and radial basis function (RBF) classifiers. SVM has proven to be effective for many classification tasks (BURGES, 1998).

Let us consider the SVM in the binary classification setting. Assume that we have a data set consisting of $\mathbf{x}_p$ samples, for $p = 1,...,P$, of labels $\mathbf{y}_p \in \{-1, 1\}$, and we wish to select, the hyperplane that leaves the maximum margin between the two classes. This is the reason why the SVM is sometimes called maximum margin classifier. Figure 11 illustrates the SVM hyperplane.

The primal formulation of the SVM cost function is given by:

$$\min_{\mathbf{w}} \ \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{p=1}^{P} \xi_p, \tag{2.32}$$

Figure 11 – SVM hyperplane illustration.



Source: Author.

subject to

$$y_p\left(\langle \mathbf{w}, \mathbf{x}_p \rangle + b\right) \geq 1 - \xi_p, \tag{2.33}$$

$$\xi_p \geq 0, \tag{2.34}$$

for $p = 1, ..., P$, where $\mathbf{w} \in \mathbb{C}^M$ is the weight vector, C is the relaxing constant, ξ_p is the slack variable, y_p is the output, $\mathbf{x}_p$ is the input vector and b is the bias term.

In addition, the dual formulation of the SVM cost function is given by:

$$\min_{\alpha} \sum_{p=1}^{P} \alpha_p - \frac{1}{2} \sum_{p_1=1}^{P} \sum_{p_2=1}^{P} \alpha_{p_1} \alpha_{p_2} y_{p_1} y_{p_2} K\left(\mathbf{x}_{p_1}, \mathbf{x}_{p_2}\right) \tag{2.35}$$

subject to

$$\sum_{p=1}^{P} \alpha_p y_p = 0, \tag{2.36}$$

$$0 \leq \alpha_p \leq C, \tag{2.37}$$

for $p = 1, ..., P$, where α_p is the Lagrange multipliers and $K(\cdot, \cdot)$ is the kernel function.

2.3.2 Principal Component Analysis (PCA)

This subsection presents the basic PCA formulation. The PCA can be solved by using the eigenvalue decomposition or by the means of the SVD. Figure 12 illustrates the application of the PCA in high correlated data. Next, the basic mathematics behind the PCA is outlined, using the eigendecomposition method.

Figure 12 – Dataset after using PCA and before.

Source: Author.

Let us consider a data set consisting of a N samples, with M attributes per sample, thus forming a matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$, so, first we extract the mean of $\mathbf{X}$:

$$\hat{\mathbf{X}} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{X}_{:,j}. \tag{2.38}$$

$$\tilde{\mathbf{X}} = \mathbf{X} - \hat{\mathbf{X}}. \tag{2.39}$$

Then, we find the covariance matrix of $\tilde{\mathbf{X}}$:

$$\mathbf{C}_{\mathbf{X}} = \frac{1}{N} \tilde{\mathbf{X}} \tilde{\mathbf{X}}^{T}, \in \mathbb{R}^{M \times M}. \tag{2.40}$$

After that, we calculate the eigendecomposition of $\mathbf{C}_{\mathbf{X}}$ and obtain the transform matrix $\mathbf{P}_{\mathbf{X}} \in \mathbb{R}^{M \times M}$. Matrix $\mathbf{P}_{\mathbf{X}}$ contains the eigenvectors of $\tilde{\mathbf{X}}$. Thus, we put $\tilde{\mathbf{X}}$ in a new basis, obtaining $\mathbf{Y}$, as follows:

$$\mathbf{Y} = \mathbf{P}_{\mathbf{X}}{}^{T} \tilde{\mathbf{X}} \in \mathbb{R}^{M \times N}. \tag{2.41}$$

An important remark is the fact that dimensionality reduction can be achieved by selecting columns of matrix $\mathbf{P}_{\mathbf{X}}$, for instance, by selecting $K < M$ columns, when transformation is performed in (2.41), $\mathbf{Y}$ has dimensions $\mathbb{R}^{K \times N}$. An interesting remark about the PCA is that the principal components are orthogonal.

2.4 Tensor Learning Concepts and Techniques

Tensor decomposition techniques have shown great successes in machine learning and data science by extending classical algorithms based on matrix factorization to multi-

modal and multi-way data (HASHEMIZADEH *et al.*, 2020). As examples in ML we may cite clustering, dimensionality reduction, latent factor models, subspace learning, and well beyond (SIDIROPOULOS *et al.*, 2017). The concept of tensors in subspace learning was already demonstrated in the mid 2000s in the works of (HE *et al.*, 2005) and (YAN *et al.*, 2005). Moreover, the work of (LU *et al.*, 2006) showed promising applications of a tensor learning algorithm in classification of tensor objetcs.

As usual, in ML applications, vectors are the arrays used to represent the input and output data. When these inputs and outputs are represented as matrices or higher-order arrays (tensors), a vectorization of these arrays are almost always done. However, the process of vectorization breaks the structure of the data, which may lead to performance issues (MA *et al.*, 2017), and also leads to input vectors with very large dimensions. However, when the training data size is relatively small compared to the feature vector dimension, it may easily result in the so called curse of dimensionality (CHEN *et al.*, 2019; LI *et al.*, 2006).

Hence, the concept of tensor learning was developed to extend the vector-based learning algorithms to accept tensors as input (TAO *et al.*, 2007b). TL avoids data vectorization and the consequent destruction of the data structure (HE *et al.*, 2017). In tensor learning methods, the multidimensional structural information of the data is preserved, providing a better multidimensional data modeling. Moreover, common tensor learning tasks may include (HASHEMIZADEH *et al.*, 2020):

- Tensor regression
- Tensor completion
- Tensor dimensionality reduction
- Tensor classification

These learning tasks form the bulk of tensor learning techniques and methods scope.

Tensor regression is the extension of linear regression to the multilinear setting (GUO *et al.*, 2011) whereas tensor completion is the process of inferring a tensor from a subset of observed entries (LIU *et al.*, 2012). Tensor dimensionality reduction is a hot research topic in machine learning, which learns data representations by preserving the original data structure while avoiding convert samples into vectors and solving the problem of high dimensionality of tensor data (NIU; MA, 2021). If the entries of a tensor random variable are correlated, the tensor samples may be confined into a subspace, where a low-dimension representation can be achieved, which makes worthwhile the study of a specialized tensor-based feature extraction

technique (LU *et al.*, 2008).

In this context, it is worth mentioning the MPCA (LU *et al.*, 2008), an extension of the PCA for tensor patterns. As well as the PCA, the MPCA reduces data dimensionality and the correlation among the variables. The MPCA is very suitable for classification problems with multidimensional data sets, generating low-dimensional matrix or tensor patterns that can be fed into classifiers (PORGES; FAVIER, 2011).

Another tensor-based dimensionality reduction technique is the multilinear extension of the LDA proposed in (YAN *et al.*, 2005), called Discriminant Analysis with Tensor Representation (DATER). In (YAN *et al.*, 2006), another extension of the LDA was proposed, denoted by MDA, where multiple lower-dimensional discriminative subspaces are derived for feature selection. In these works, iterative algorithms similar to the Alternating Least Squares (ALS) algorithm are used to maximize tensor-based discriminant criteria.

And lastly, tensor classification is the exact process of classifying and categorizing data that is in tensor format. The work of (TAO *et al.*, 2005) proposed in 2005 a Supervised Tensor Learning (STL) method, which uses a rank-one tensor to capture the data structure, thereby alleviating the overfitting and curse of dimensionality problems in the conventional SVM. The STL classifier proposed in (TAO *et al.*, 2005) learns a series of projection vectors to determine the class label of a measurement according to a multilinear decision function, all in an iterative procedure. This work was extended in (TAO *et al.*, 2007b) and it has been extensively studied and expanded in recent years, which led to the development of multilinear models called STMs.

Furthermore, in the context of STL, preserving the structural information and exploiting the discriminating nonlinear relationships of tensor data are crucial for improving the performance of learning tasks (HE *et al.*, 2017), then, the STM operates on high-order data directly to facilitate the learning process.

In addition, there exist many tensor decomposition models (CP/PARAFAC, Tucker, Tensor-Train, HOSVD, etc.) which can be used in modeling ML systems and such decompositions can generate different STL techniques and models. Several modifications of the STM were proposed, such in (KOTSIA *et al.*, 2012), where the SPM technique is modeled for a PARAFAC weights tensor of high rank. In (KOTSIA; PATRAS, 2011), the STuM is proposed by considering the Tucker decomposition and in (CHEN *et al.*, 2019), the Support Tensor-Train Machine STTM is presented, which employs a general and scalable tensor train as the parameter model.

In the next subsections we provide formulations for important tensor learning techniques, the SPM and STuM classifiers, derived from the original STL framework (TAO *et al.*, 2007b), and, the MPCA.

2.4.1 Support Tensor Machines (STM)

A standard SVM model is based on vector inputs and cannot directly deal with matrices or higher dimensional data structures, which are very common in real-life applications (CHEN *et al.*, 2019). The SVM realization on such high dimensional inputs is by reshaping each sample into a vector. However, besides breaking down the data structure, when the training data size is relatively small compared to the feature vector dimension, it may easily result in poor classification performance, known as curse of dimensionality (CHEN *et al.*, 2019; LI *et al.*, 2006).

The STM extends the SVM to tensor patterns by constructing multilinear models to the weights tensor (TAO *et al.*, 2007b). STM techniques have found application in many areas. The work of (ZHOU *et al.*, 2013) applies a regression STM-based model in neuroimaging and, in (CALVI *et al.*, 2019), STM is used in financial forecasting, while the work of (GUO *et al.*, 2014) exploits the tensor learning in hyperspectral image classification. The authors of (MA *et al.*, 2017) used a STM to detect bubble defects in Lithium-ion polymer cell sheets. A complete STM overview with references and recent works can be found in (XIANG *et al.*, 2018).

2.4.1.1 Support PARAFAC Machine (SPM)

The technique proposed in (TAO *et al.*, 2007b) is a STM-type classifier based on the assumption that the weight tensor follows a rank-one PARAFAC model. In (KOTSIA *et al.*, 2012), the higher-rank version of the STM is derived, under the assumption that the weight tensor is expressed as a higher-rank PARAFAC decomposition, which is called SPM. This allows multiple projections along each mode. These algorithms implement multiple SVM-type problems iteratively, one for each mode of the tensor.

The main concept of the STM is the same as for the SVM, where the hyperplane equation is given by:

$$f(\mathcal{Y}_p) = \langle \mathcal{Y}_p, \mathcal{W} \rangle + b, \tag{2.42}$$

where $\mathscr{Y}_p$ is the p-th sample tensor data and $\mathscr{W}$ is the weight tensor. Then, the generic STM primal problem formulation for third-order tensors can be expressed as follows:

$$\min_{\mathscr{W}} \ \frac{1}{2}\langle \mathscr{W}, \mathscr{W} \rangle + C \sum_{p=1}^{P} \xi_p, \tag{2.43}$$

subject to

$$y_p \left(\langle \mathscr{W}, \mathscr{Y}_p \rangle + b \right) \geq 1 - \xi_p, \tag{2.44}$$

$$\xi_p \geq 0, \quad n = 1, ..., P, \tag{2.45}$$

where $\mathscr{W} \in \mathbb{C}^{R_1 \times R_2 \times R_3}$, C is the relaxing constant, ξ_p is the p-th slack variable, $y_p \in$ {-1,1} represents the class tag of the p-th sample, $\mathscr{Y}_p \in \mathbb{C}^{R_1 \times R_2 \times R_3}$ and b is the bias term. The multilinear decision function that classifies the set of tensors patterns is given by $f(\mathscr{Y}_p) = \text{sign}\left(\langle \mathscr{Y}_p, \mathscr{W} \rangle + b \right)$, where sign$(\cdot)$ is the sign function.

The trilinear SPM assumes that the weight tensor follows the PARAFAC decomposition: $\mathscr{W} = \sum_{q=1}^{Q} \mathbf{A}^{(1)}_{.,q} \circ \mathbf{A}^{(2)}_{.,q} \circ \mathbf{A}^{(3)}_{.,q}$, where $\mathbf{A}^{(1)} \in \mathbb{R}^{R_1 \times Q}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{R_2 \times Q}$ and $\mathbf{A}^{(3)} \in \mathbb{R}^{R_3 \times Q}$ are factor matrices and Q denotes the rank of the tensor. The problem (2.43)-(2.45) is solved iteratively by estimating, in an alternating way, one of the factor matrices using the previous estimations of the other factor matrices. Each iteration of the algorithm is composed of three steps, each factor matrix $\mathbf{U}^{(n)}$ being estimated in one step using the standard SVM, by fixing the other factor matrices to their values obtained in the previous iterations. In what follows, the optimization problem defined in (2.43)-(2.45) is expressed in such a way that it can be solved using the standard (vector-based) SVM, assuming that all the factor matrices are known, excepting for one of them.

By using (2.8), the problem (2.43)-(2.45) can be reformulated in terms of the mode-n unfoldings of $\mathscr{W}$ and $\mathscr{Y}_p$, for some $n \in [1,3]$, in the following way:

$$\min_{\mathbf{W}^{[n]}} \ \frac{1}{2} Tr \left[\mathbf{W}^{[n]} (\mathbf{W}^{[n]})^T \right] + C \sum_{p=1}^{P} \xi_p, \tag{2.46}$$

subject to

$$y_p \left(\left[Tr \mathbf{W}^{[n]} (\mathbf{Y}_p^{[n]})^T \right] + b \right) \geq 1 - \xi_p, \tag{2.47}$$

$$\xi_p \geq 0, \quad p = 1, ..., P, \tag{2.48}$$

where $\mathbf{W}^{[n]}, \mathbf{Y}_p^{[n]} \in \mathbb{C}^{R_n \times R_1 R_2 R_3 / R_n}$. Moreover, using (2.18) and assuming that all the factor matrices are known, excepting $\mathbf{A}^{(n)}$, (2.46)-(2.48) is reexpressed as (KOLDA; BADER, 2009;

KOTSIA *et al.*, 2012):

$$\min_{\mathbf{A}^{(n)}} \ \frac{1}{2} Tr\left[\mathbf{A}^{(n)}(\mathbf{A}_{\odot}^{(n)})^T \mathbf{A}_{\odot}^{(n)}(\mathbf{A}^{(n)})^T\right] + C\sum_{p=1}^{P}\xi_p, \tag{2.49}$$

subject to

$$y_p\left(Tr\left[\mathbf{A}^{(n)}(\mathbf{A}_{\odot}^{(n)})^T(\mathbf{Y}_p^{[n]})^T\right] + b\right) \geq 1-\xi_p, \tag{2.50}$$

$$\xi_p \geq 0, \quad p = 1,...,P, \tag{2.51}$$

with $\mathbf{A}_{\odot}^{(n)} \in \mathbb{R}^{R_1R_2R_3/R_n \times Q}$.

By defining $\mathbf{B}^{(n)} = (\mathbf{A}_{\odot}^{(n)})^T \mathbf{A}_{\odot}^{(n)} \in \mathbb{R}^{Q\times Q}$, $\tilde{\mathbf{A}}^{(n)} = \mathbf{A}^{(n)}(\mathbf{B}^{(n)})^{1/2} \in \mathbb{R}^{R_n \times Q}$, with $(\mathbf{B}^{(n)})^{1/2}$ being a square matrix with full row rank, and $\tilde{\mathbf{Y}}_p^{(n)} = \mathbf{Y}_p^{[n]}\mathbf{A}_{\odot}^{(n)}(\mathbf{B}^{(n)})^{-1/2} \in \mathbb{R}^{R_n\times Q}$, we get:

$$\begin{aligned} Tr\left[\mathbf{A}^{(n)}(\mathbf{A}_{\odot}^{(n)})^T\mathbf{A}_{\odot}^{(n)}(\mathbf{A}^{(n)})^T\right] = Tr\left[\tilde{\mathbf{A}}^{(n)}(\tilde{\mathbf{A}}^{(n)})^T\right] = \\ vec\left(\tilde{\mathbf{A}}^{(n)}\right)^T vec\left(\tilde{\mathbf{A}}^{(n)}\right). \end{aligned} \tag{2.52}$$

$$\begin{aligned} Tr\left[\mathbf{A}^{(n)}(\mathbf{A}_{\odot}^{(n)})^T(\mathbf{Y}_p^{(n)})^T\right] = Tr\left[\tilde{\mathbf{A}}^{(n)}(\tilde{\mathbf{Y}}_p^{(n)})^T\right] = \\ vec\left(\tilde{\mathbf{A}}^{(n)}\right)^T vec\left(\tilde{\mathbf{Y}}_p^{(n)}\right). \end{aligned} \tag{2.53}$$

Eqs. (2.49)-(2.50) can be rewritten as:

$$\min_{\tilde{\mathbf{A}}^{(n)}} \ \frac{1}{2} vec\left(\tilde{\mathbf{A}}^{(n)}\right)^T vec\left(\tilde{\mathbf{A}}^{(n)}\right) + C\sum_{p=1}^{P}\xi_p, \tag{2.54}$$

subject to

$$y_p\left(vec\left(\tilde{\mathbf{A}}^{(n)}\right)^T vec\left(\tilde{\mathbf{Y}}_p^{(n)}\right) + b\right) \geq 1-\xi_p, \tag{2.55}$$

$$\xi_p \geq 0, \quad p = 1,...,P, \tag{2.56}$$

The problem formulated in (2.54)-(2.55) is the format of a standard vector-based SVM, with input given by $vec(\tilde{\mathbf{Y}}_p^{(n)})$ and weight vector $vec(\tilde{\mathbf{A}}^{(n)})$. After finding $\tilde{\mathbf{A}}^{(n)}$ using the SVM, the original factor matrix can be computed as: $\mathbf{A}^{(n)} = \tilde{\mathbf{A}}^{(n)}(\mathbf{B}^{(n)})^{-1/2}$, for $n = 1,...,3,$. The matrix $\mathbf{B}^{(n)}$ is assumed to be known as it depends on the other factor matrices, which are also assumed known at this stage of the algorithm. It is important to reemphasize that $(\mathbf{B}^{(n)})^{1/2}$ must be invertible.

The algorithm continues by iteratively computing each factor matrix assuming that the other factor matrices are fixed and it stops when the estimated classes do not change from one iteration to another. Moreover, the factor matrices are initialized randomly. Uniqueness

properties of the PARAFAC decomposition were not considered in this algorithm, as the factor matrices does not need to be unique in order perform classification. This assumption is validated in Chapters 3 and 4, as uniqueness conditions were not imposed and yet classification results achieved great performance. When $Q = 1$, the above presented technique will be denoted by Rank-1 Support PARAFAC Machines (R1-SPM), otherwise, it will be denoted simply by SPM.

It is important to mention here that the initial values for the matrices $\mathbf{U}^{(l)}$, are randomly chosen. That means: at each iteration we solve for the parameters of the mode l while keeping the parameters for all other modes fixed.

2.4.1.2 Support Tucker Machine (STuM)

Roughly, the main idea of the STuM is similar to that of the SPM, which aims at finding a multilinear decision function that classifies a set of P tensors $\mathscr{X}_p \in \mathbb{R}^{R_1 \times R_2 \times R_3}$, with p = 1,...,P. However, instead of assuming a PARAFAC decomposition for the weight tensor, the STuM assumes that $\mathscr{W}$ follows a Tucker decomposition, as such (KOTSIA; PATRAS, 2011): $\mathscr{W} = \mathscr{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)}$, with $\mathscr{G} \in \mathbb{C}^{Q_1 \times Q_2 \times Q_3}$ being the core tensor, $\mathbf{A}^{(n)} \in \mathbb{C}^{R_n \times Q_n}$, for $n = 1,2,3$, the factor matrices and (Q_1, Q_2, Q_3) the trilinear rank of $\mathscr{W}$. In scalar form the tensor $\mathscr{W}$ is given by $w_{r_1,r_2,r_3} = \sum_{q_1=1}^{Q_1} \sum_{q_2=1}^{Q_2} \sum_{q_3=1}^{Q_3} g_{q_1,q_2,q_3} a^{(1)}_{r_1,q_1} a^{(2)}_{r_2,q_2} a^{(3)}_{r_3,q_3}$.

The optimization problem is the same as in (2.43)-(2.45), rewritten as (2.46)-(2.48). Similarly as in the SPM, an iterative approach is adopted by the STuM, in which the factor matrices associated with each of the modes are estimated keeping all the other factor matrices and the core tensor fixed. However, the iterations of the STuM have an additional step for computing the core tensor $\mathscr{G}$, leading to a total of four steps per iteration.

Using (2.27), (2.46)-(2.48) can be expressed as:

$$\min_{\mathbf{A}^{(n)}} \ \frac{1}{2} Tr\left[\mathbf{A}^{(n)} \mathbf{G}^{(n)} (\mathbf{A}^{(n)}_{\otimes})^T \mathbf{A}^{(n)}_{\otimes} (\mathbf{G}^{(n)})^T (\mathbf{A}^{(n)})^T\right] + C \sum_{p=1}^{P} \xi_p, \tag{2.57}$$

subject to

$$y_p \left(Tr[\mathbf{A}^{(n)} \mathbf{G}^{(n)} \left(\mathbf{A}^{(n)}_{\otimes}\right)^T \left(\mathbf{Y}^{[n]}_p\right)^T] + b \right) \geq 1 - \xi_p, \tag{2.58}$$

$$\xi_p \geq 0, \quad p = 1, ..., P, \tag{2.59}$$

with $\mathbf{A}^{(n)}_{\otimes} \in \mathbb{C}^{R_1 R_2 R_3 / R_n \times Q_1 Q_2 Q_3 / Q_n}$ and $\mathbf{G}^{(n)} \in \mathbb{C}^{Q_n \times Q_1 Q_2 Q_3 / Q_n}$. Similarly as in the SPM case, let us define $\mathbf{C}^{(n)} = \mathbf{G}^{(n)} (\mathbf{A}^{(n)}_{\otimes})^T \mathbf{A}^{(n)}_{\otimes} (\mathbf{G}^{(n)})^T \in \mathbb{R}^{Q_n \times Q_n}$, $\tilde{\mathbf{A}}^{(n)} = \mathbf{A}^{(n)} (\mathbf{C}^{(n)})^{1/2} \in \mathbb{R}^{R_n \times Q_n}$, with

$(\mathbf{C}^{(n)})^{1/2}$ being full row rank, and $\tilde{\mathbf{Y}}_p^{(n)} = \mathbf{Y}_p^{[n]}\mathbf{A}_{\otimes}^{(n)}(\mathbf{G}^{(n)})^T(\mathbf{C}^{(n)})^{-1/2} \in \mathbb{R}^{R_n \times Q_n}$. Using these definitions, the problem formulated in (2.57) and (2.59) can then be rewritten as (2.54)-(2.55).

As earlier mentioned, the cost function in (2.54)-(2.55) follows the standard SVM format, with input $vec(\tilde{\mathbf{Y}}_p^{(n)})$ and weight vector $vec(\tilde{\mathbf{A}}^{(n)})$. Similarly as for the SPM, the STuM computes each factor matrix assuming that the other factor matrices and the core tensor are known and equal to their previous estimations. The estimation of the core tensor is obtained using the following vectorization of tensor $\mathscr{W}$ (KOTSIA; PATRAS, 2011): $vec(\mathbf{W}^{[1]}) = \mathbf{A}_{\otimes} vec\left(\mathbf{G}^{[1]}\right)$, where $\mathbf{A}_{\otimes} = \mathbf{A}^{(3)} \otimes \mathbf{A}^{(2)} \otimes \mathbf{A}^{(1)} \in \mathbb{C}^{R_1R_2R_3 \times Q_1Q_2Q_3}$. Indeed, using this equation and (2.46)-(2.48) can be rewritten for $n = 1$ as:

$$\min_{\mathbf{G}^{[1]}} \ \frac{1}{2} vec\left(\mathbf{G}^{[1]}\right)^T \mathbf{A}_{\otimes}^T \mathbf{A}_{\otimes} vec\left(\mathbf{G}^{[1]}\right) + C\sum_{p=1}^{P} \xi_p \tag{2.60}$$

subject to

$$y_p\left(vec\left(\mathbf{G}^{[1]}\right)^T (\mathbf{A}_{\otimes})^T vec\left(\mathbf{Y}_p^{[1]}\right) + b\right) \geq 1 - \xi_p, \tag{2.61}$$

$$\xi_p \geq 0, \quad p = 1, ..., P. \tag{2.62}$$

Finally, by defining $\mathbf{D} = (\mathbf{A}_{\otimes})^T\mathbf{A}_{\otimes} \in \mathbb{R}^{Q_1Q_2Q_3 \times Q_1Q_2Q_3}$, $\tilde{\mathbf{g}}^{(1)} = \mathbf{D}^{1/2} vec(\mathbf{G}^{[1]}) \in \mathbb{R}^{Q_1Q_2Q_3}$, where $\mathbf{D}^{1/2}$ is a square matrix, and $\tilde{\mathbf{y}}_p^{(1)} = \mathbf{D}^{-1/2}\mathbf{A}_{\otimes}^T vec(\mathbf{Y}_p^{[1]}) \in \mathbb{R}^{Q_1Q_2Q_3}$, we get:

$$\min_{\mathbf{G}^{[1]}} \ \frac{1}{2}\left(\tilde{\mathbf{g}}^{(1)}\right)^T \tilde{\mathbf{g}}^{(1)} + C\sum_{p=1}^{P} \xi_p \tag{2.63}$$

subject to

$$y_p\left(\left(\tilde{\mathbf{g}}^{(1)}\right)^T \tilde{\mathbf{y}}_p^{(1)} + b\right) \geq 1 - \xi_p, \tag{2.64}$$

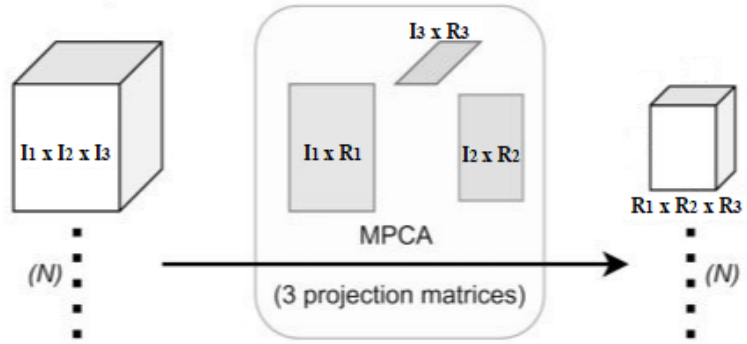
$$\xi_p \geq 0, \quad p = 1, ..., P. \tag{2.65}$$

The final step of each STuM iteration consists in finding $\mathbf{g}^{(1)}$ through the standard SVM procedure in (2.63)-(2.65), and then, estimating the core tensor as: $\mathbf{G}^{[1]} = \mathbf{D}^{-1/2}\tilde{\mathbf{g}}^{(1)}$. Moreover, the factor matrices and the core tensor are randomly initialized, and the stop criterion is achieved when the estimated classes do not change from one iteration to another. Also, matrices $\mathbf{C}^{(n)})^{1/2}$ and $\mathbf{D}^{1/2}$ must be invertible.

Similarly as for the SPM algorithm, no uniqueness condition is imposed for the STuM, and yet the technique achieved high classification rates in Chapters 3 and 4.

The theoretical benefit of the STuM lies in the use of the Tucker decomposition, a generic decomposition with a high number of free parameters. In (PEIXOTO *et al.*, 2021), the

Figure 13 – Illustration of the MPCA approach.



Source: Adapted from (SWIFT *et al.*, 2021).

StuM has shown a great ability to perform classification of volcano-seismic classification events, outperforming other classifiers.

As it can be viewed, the STuM has more parameters to be estimated in comparison to the SPM and one extra step for estimating the core tensor. On the other hand, the STuM has a greater degree of freedom for fitting the weight tensor than the PARAFAC decomposition, which is a great advantage for performing classification.

2.4.2 Multilinear Principal Component Analysis

A typical tensor object in pattern recognition or machine vision applications is commonly in a high-dimensional tensor space. Recognition methods operating on this space suffer from the curse of dimensionality, where handling high-dimensional samples is computationally expensive and many classifiers perform poorly in high-dimensional spaces if the number of training samples is small. Therefore, dimensionality reduction in high-dimensional spaces is sometimes mandatory, which can be achieved by techniques such as the MPCA.

The MPCA is an extension of the PCA for tensors and a powerful method for dimensionality reduction and feature extraction in tensor patterns, often used in classification tasks (LU *et al.*, 2011). The MPCA carries out a multilinear projection with dimensionality reduction using orthonormal transformation matrices, capturing most of the data variance (LU *et al.*, 2008; LU *et al.*, 2006). However, contrarily to the standard PCA that generates fully uncorrelated data, the MPCA is not able to create perfectly uncorrelated variables. The MPCA has a wide range of applications, such as gait, face, fingerprint, age, and fault recognition (LU *et al.*, 2008; LU *et al.*, 2006; PORGES; FAVIER, 2011).

Considering a set of P samples of a N^{th}-order tensor random variable, denoted by $\mathscr{X}_p \in \mathbb{C}^{I_1 \times \ldots \times I_N}$, for $1 \leq p \leq P$, the MPCA can be summarized as follows:

1. Subtract the average: $\hat{\mathscr{X}_p} = \mathscr{X}_p - \bar{\mathscr{X}} \in \mathbb{C}^{I_1 \times \ldots \times I_n}$, for $1 \leq p \leq P$, where $\bar{\mathscr{X}} = \frac{1}{P}\sum_{p=1}^{P} \mathscr{X}_p \in \mathbb{C}^{I_1 \times \ldots \times I_N}$.
2. Calculate the HOSVD of the $(N+1)$-th order tensor $\hat{\mathscr{X}} \in \mathbb{C}^{I_1 \times \ldots \times I_N \times P}$ that contains all the tensor samples $\hat{\mathscr{X}_p}$, denoting by $\mathbf{U}^{(n)} \in \mathbb{C}^{I_n \times R_n}$ the factor matrices of the first N modes, with $R_n \leq I_n$ and $1 \leq n \leq N$.
3. Calculate the projected data tensor $\mathscr{Y} \in \mathbb{C}^{R_1 \times \ldots \times R_N \times P}$:

$$\mathscr{Y} = \hat{\mathscr{X}} \times_1 \mathbf{U}^{(1)^T} \times_2 \mathbf{U}^{(2)^T} \ldots \times_N \mathbf{U}^{(N)^T}. \tag{2.66}$$

The MPCA follows the above steps iteratively until convergence is achieved. If $R_n = I_n$, for $n = 1, \ldots, N$, no dimensionality reduction is carried out, therefore, throughout the next chapters, this method will be denoted by Multilinear Principal Component Analysis - Full Projection (MPCA-FP). On the other hand, if $R_n < I_n$ for some n, dimensionality reduction is performed. This late approach will be denoted by Multilinear Principal Component Analysis - Dimensionality Reduction (MPCA-DR) throughout the rest of this work. A similar abbreviation will be used for the PCA, i.e., Principal Component Analysis - Full Projection (PCA-FP) and Principal Component Analysis - Dimensionality Reduction (PCA-DR). Figure 13 illustrates the projection process of the MPCA.

It is important to note that, contrarily to the standard PCA that generates fully uncorrelated output data, the MPCA is not able to create perfectly uncorrelated variables.

3 FRAMEWORK FOR CLASSIFICATION OF SEISMIC EVENTS

This chapter begins the presentation of the original contributions of this thesis, presenting a supervised tensor-based learning framework for classifying volcano-seismic events from signals recorded at the Ubinas volcano, in Peru, during a period of great activity in 2009. The proposed method presented in this chapter is fully tensorial, as it integrates the three main steps of the automatic classification system (feature extraction, dimensionality reduction and classifier) in a general multidimensional framework for tensor data, joining tensor learning techniques such as the MPCA and the STuM. To the best of the authors' knowledge, no previous work in the literature has used tensor learning techniques for the classification of seismic events.

By exploiting the use of multiple multichannel triaxial sensors, operating simultaneously in two seismic stations, the tensor patterns are constructed as: *stations* $\times$ *channels* $\times$ *features*. The multidimensional structure of the data is then preserved, avoiding the tensor vectorization that often leads to a feature vector with a large dimension, which increases the number of parameters and may cause the "curse of dimensionality". Moreover, the array vectorization breaks down the multidimensional structure of the data, which usually leads to performance degradation.

The results showed a good performance of the proposed multilinear classification system, significantly outperforming its vectorial counterparts. The best result was obtained with the STuM classifier alongside with the MPCA.

This chapter is divided as follows. First, we present the motivations for developing the said framework. Next, the proposed framework and classification system are outlined. Further on, the adopted database is described and the results discussed.

3.1 Motivation

Volcanic eruptions such as the ones of the Volcan de Fuego (Guatemala, June 2018), the Stromboli volcano (Italy, July 2019) and more recently, the Semeru volcano (Indonesia, November 2019 and December 2021), showed the catastrophic effects of volcanic eruptions and their impact on nearby infrastructure. Combined, these eruptions destroyed a large amount of buildings and, unfortunately, made hundreds of victims. Volcano-seismic events can be considered a threat to humans and cities located in nearby volcano areas (RAHMAN *et al.*, 2016).

The automatic detection of volcano-seismic events is of great importance to society due to the violent effects of volcanic eruptions. Indeed, even small volcanic eruptions can be catastrophic to people and small towns surrounding volcanoes, such as the cities next to the valleys of the volcanic chain in southern Peru, which have to deal with this threat constantly. In fact, volcanic activities have been a latent threat to humans since the existence of humanity.

Fortunately, the seismic activity of a volcano can be observed by seismic sensors. When the seismicity of the volcanic increases, the probability of eruption gets high. However, it is necessary to determine whether or not the activity will result in an eruption or in any other harmful event. The volcano-seismic events can be categorized into five main classes (MCNUTT, 2005): Long Period (LP), Tremors (TR), Explosion (EX), Volcano-tectonic (VT), and Hybrid (HB). The problem of detecting these events can be elucidated by analyzing the time series of the signals, in order to predict or detect the eruptive state of a volcano. However, in many places, the volcanic-seismic data series are still analyzed manually, which may lead to errors or big delays in the detection of the events.

The classification of seismic patterns has shown great improvement over the past years, with many methods being developed (MALFANTE *et al.*, 2018). Indeed, supervised learning has drawn a great attention of the scientific community in the area of seismology. For instance, in (SHIMSHONI; INTRATOR, 1998), machine learning techniques are applied for the recognition of volcanic waveforms, with a hierarchy of ANNs being used. In (MALFANTE *et al.*, 2018), the authors try to distinguish natural seismic waveforms of earthquakes from waveforms of man-made explosions.

In (SCARPETTA *et al.*, 2005), automatic classification of local seismic signals and volcano-tectonic earthquakes are proposed with a method based on a supervised neural networks. The work (CURILEM *et al.*, 2009) aimed to construct a system able to classify seismic signals for the Villarrica volcano, one of the most active volcanoes in South America, with an ANN and a genetic algorithm. New techniques for classifying seismic signals can be found in (ZHOU *et al.*, 2012), using of the cepstral domain with the SVM classifier. In addition, the work (MALFANTE *et al.*, 2018) uses attributes in the temporal, spectral, and cepstral domains for the extraction of features, along with the SVM method. In (LARA *et al.*, 2020), an automatic classification system for volcano events is presented using the EMD. More recently, the work (CURILEM *et al.*, 2018) explored deep learning by using CNNs to classify spectrograms of seismic events from a South American volcano.

A few works have used multilinear techniques applied to seismic signals. In (VRABIE *et al.*, 2006), a three-mode model, using polarization, distance and temporal modes, is used for seismic event waves separation, taking into account the specific structure of signals that are recorded with these arrays, providing a data-structure-preserving processing. In (PAULUS; MARS, 2006), the authors used multilinear techniques such as Multicomponent Wideband Spectral-Matrix Filtering (MC-WBSMF) on geophysical data to separate interfering wavefields or to compute the direction of arrival (DOA) on a vector-sensor array. However, to the best of the authors' knowledge, no previous work in the literature has used tensor learning techniques for the classification of seismic events.

In this thesis, supervised tensor learning is used for classifying volcanic-seismic events. In particular, a tensor-based learning framework is proposed to classify the five main events of a volcano from seismic signals recorded at the Ubinas volcano, in Peru. The data tensors are constructed by exploiting the use of multichannel triaxial sensors. Indeed, contrary to the standard approach of the literature of using only a single channel sensor, the database used in this thesis was recorded with sensors that have 3 channels: vertical, east and north. The use of triaxial sensors has shown to offer a better representation of the seismic signals in comparison to single-channel sensors (LARA *et al.*, 2020).

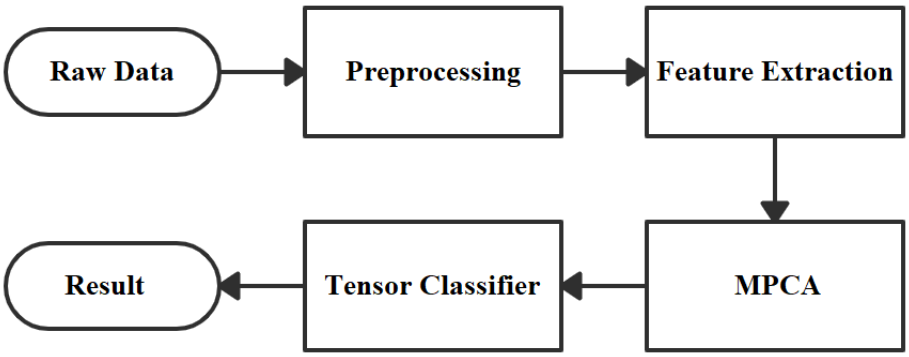
The use of multiple sensors, operating simultaneously in more than one seismic station, is also exploited to build the tensor patterns. The three dimensional feature arrays are constructed as follows $stations \times channels \times features$. The tensor representation of the data is preserved in the proposed approach, avoiding the earlier mentioned drawbacks of the vectorization process.

The present framework can be viewed as a multilinear alternative of the conventional linear ML approaches in seismic classification. The novelty of the proposed method is a framework for processing seismic signals as tensors and classifying them using tensorial classifiers, which provided superior accuracy in comparison to the conventional SVM, as shown in the results section.

3.2 Proposed Classification System

In this section, the proposed classification system based on a tensorial framework is presented. Fig. 14 illustrates the main steps of the tensor-based framework for classifying the volcano-seismic events. Firstly, a preprocessing step, consisting in signal conditioning and

Figure 14 – Steps of the classification system.



Source: (PEIXOTO *et al.*, 2021).

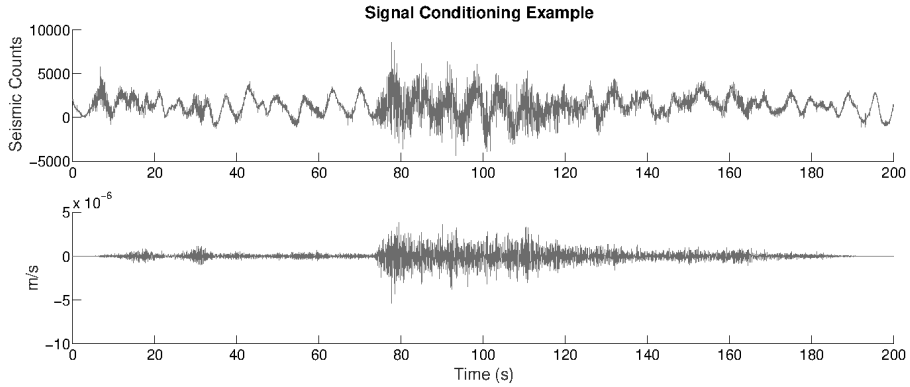
instrumental correction, is performed. Then, the features are extracted from time, frequency, Power Spectral Density (PSD) and Hilbert domains, with a total of 56 attributes being calculated for each one of the 5136 signals. This number of attributes was determined after running preliminary tests in order to find the value that presents the best results. It is assumed that the seismic stations record the same event.

After that, the multilinear dimensionality reduction technique MPCA is applied and, finally, the classifiers are trained and validated, using three versions of the STM: the low-rank R1-SPM, the higher-order SPM and the STuM. The STM preserves the structural information of data tensor and alleviate the small sample size problem, in addition to reduce the number of parameters. Moreover, as earlier mentioned, the PARAFAC and Tucker decomposition were chosen due to their characteristics that are well-suited to the problem considered in the present work.

The learning framework of the classification system is fully tensorial. Indeed, excepting the instrumental correction, all the steps involved use tensor-based techniques. The multidimensional structure of the data is then preserved and the number of parameters to be estimated is diminished. To the best of the authors' knowledge, no previous work in the literature has used tensor-based machine learning algorithms to classify volcano-seismic events.

One of the advantages of the proposed framework is due to the fact that a tensor representation of the data allows the use of tensorial methods in its processing, exploring its diversity. With the MPCA it is possible to extract characteristics from the data set and also project the input tensor into a new multidimensional basis, thus eliminating correlation and retaining only useful information to be used on the classification process. If the data input is not in tensor form, a tensorization step is performed. In the following, the steps of the proposed

Figure 15 – Waveform of a LP signal before (upper) and after instrumental correction (lower).



Source: (PEIXOTO *et al.*, 2021).

system are detailed.

It is also possible to skip the dimensionality reduction step if necessary, meaning that the MPCA output will only project the data into a new basis, maintaining the same dimensions as the input. The final output of the proposed framework is the correct classification of the data.

3.2.1 Preprocessing

The first step of the preprocessing stage is the subtraction of the time-average mean of the signals. After that, the instrumental correction is done by computing the deconvolution associated with the transfer function of the sensor, expressing the seismic signals in their original unity, similarly as in (LARA *et al.*, 2020). The goal is to standardize the velocity waveforms obtained by the sensors, originally measured in Seismic Counts, to the unit meter per second (m/s), making the classifier independent on the type of sensor used. More details about the instrumental correction are given in (HAVSKOV; ALGUACIL, 2016). Next, the signal is smoothed by a Savitzky-Golay filter (SAVITZKY; GOLAY, 1964) and then, a bandpass Butterworth filter from 0.8 to 45 Hz is applied. Fig. 15 illustrates an LP signal before and after the preprocessing.

3.2.2 Feature Extraction

The next step of the classification system is feature extraction. The samples were arranged in $2 \times 3 \times 3000$ third-order tensors, where the first mode represents the station (UBW and UBN), the second mode the channels (HHE, HHN and HHZ) and the third mode the number of time samples. After the feature extraction, the tensor patterns are organized as $2 \times 3 \times 56$

Table 1 – Feature description.

	Fourier	PSD	Hilbert	Time
Windowed average	10	10	10	10
Total average	1	1	1	1
Kurtosis	1	1	1	1
Standard deviation	1	1	1	1
Skewness	1	1	1	1
Total	14	14	14	14

Source: (PEIXOTO *et al.*, 2021).

third-order tensors, with the third mode representing 56 attributes calculated for each signal. The complete database of patterns can be arranged in a single fourth-order tensor ($2 \times 3 \times 56 \times 856$), where the fourth mode represents the samples. In other words, there are 856 third-order tensors, with each one corresponding to a specific seismic event registered.

The 56 attributes used in this work are shown in Table 1, using frequency and time domains. The frequency domain features are calculated from the estimated PSD, as well as using a multilinear approach, by means of the MDFT (TOLIMIERI *et al.*, 2012; TSUI *et al.*, 2008), while the time domain attributes are calculated from the preprocessed time series and their Hilbert transforms, resulting in four types of features, as shown in Table 1: MDFT, PSD, Hilbert and time. The following 14 attributes are calculated for each of these four domains: 10 windowed averages, total average, kurtosis, skewness and standard deviation. The PSD is calculated using the Welch's method with overlapping of 75%. The windowed average is obtained by dividing the time and frequency series into 10 non-overlapping windows and calculating the average of each window, leading to 10 attributes. These attributes have shown to be efficient for describing volcano-seismic events, e.g. (LARA *et al.*, 2020; PEIXOTO *et al.*, 2021).

The use of attributes calculated from both the MDFT and PSD may seem to be redundant, however, preliminary results have shown that the combined use of these two kind of features provides better performances than using only one of them. This is due to the fact that the Fourier transform is obtained by using its multidimensional form, while the standard PSD is used. Moreover, the MDFT provides better frequency resolution, whereas the PSD has a better amplitude accuracy.

Moreover, after extensive tests and evaluations, the proposed classification system achieved its better results when using this combination of attributes.

3.2.3 MPCA Application

After the feature extraction step, the MPCA is applied for dimensionality and data correlation reduction. As the available volcano-seismic database is a set of $P = 856$ third-order tensors $\mathscr{X}_p \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, for $1 \leq p \leq P$, where $I_1 = 2$, $I_2 = 3$ and $I_3 = 56$, the MPCA technique described in Chapter 2 is applied, outputting a tensor $\mathscr{Z}_p \in \mathbb{R}^{R_1 \times R_2 \times R_3}$. Both MPCA-FP, if $R_n = I_n$ for $n = 1,2,3$, and MPCA-DR, if $R_n < I_n$ for $n = 1,2,3$, are tested in the results section.

3.2.4 Classification

The last step of the tensor-based system is the classification algorithm itself. Three multilinear classification methods can be selected in the framework: the R1-SPM, the SPM (KOTSIA *et al.*, 2012) and the STuM (KOTSIA; PATRAS, 2011). The objective of these classifiers is to estimate the classes of the $P = 856$ tensor samples, with inputs given by the components of the MPCA.

In this chapter, the three tensorial classifiers were tested, alongside the SVM, for comparison purposes. We must note that the tensor-based framework can be coupled with the vector-based SVM, however, the tensorial input data must be vectorized first, breaking the multidimensional structure. Table 2 resumes the classification system adopted techniques and classifiers.

Table 2 – Classification framework description.

Preprocessed Data	856 third-order $2 \times 3 \times 3000$ tensors
Feature Extraction	MDFT, PSD, Hilbert
Feature Transformation	MPCA-FP, MPCA-DR, PCA-FP, PCA-DR
Classification	SVM (optional), R1-SPM, SPM, STuM

Source: Author.

3.3 Database Description

The Ubinas volcano is an active andesitic stratovolcano, located 60 km east of the city of Arequipa, with an average occurrence of 6-7 eruptions per century and persistent fumarolic and phreatic activity. The recent episodes are characterized by volcanic eruptions and by the extrusion of a lava dome. The CENVUL (Centro Vulcanológico Nacional) is a service of the IGP (Instituto Geofísico del Peru) in charge to keeping in alert the volcano monitoring in Peru. Recently, three eruptive events were identified by the IGP, into the periods 2006-2009, 2014-2017

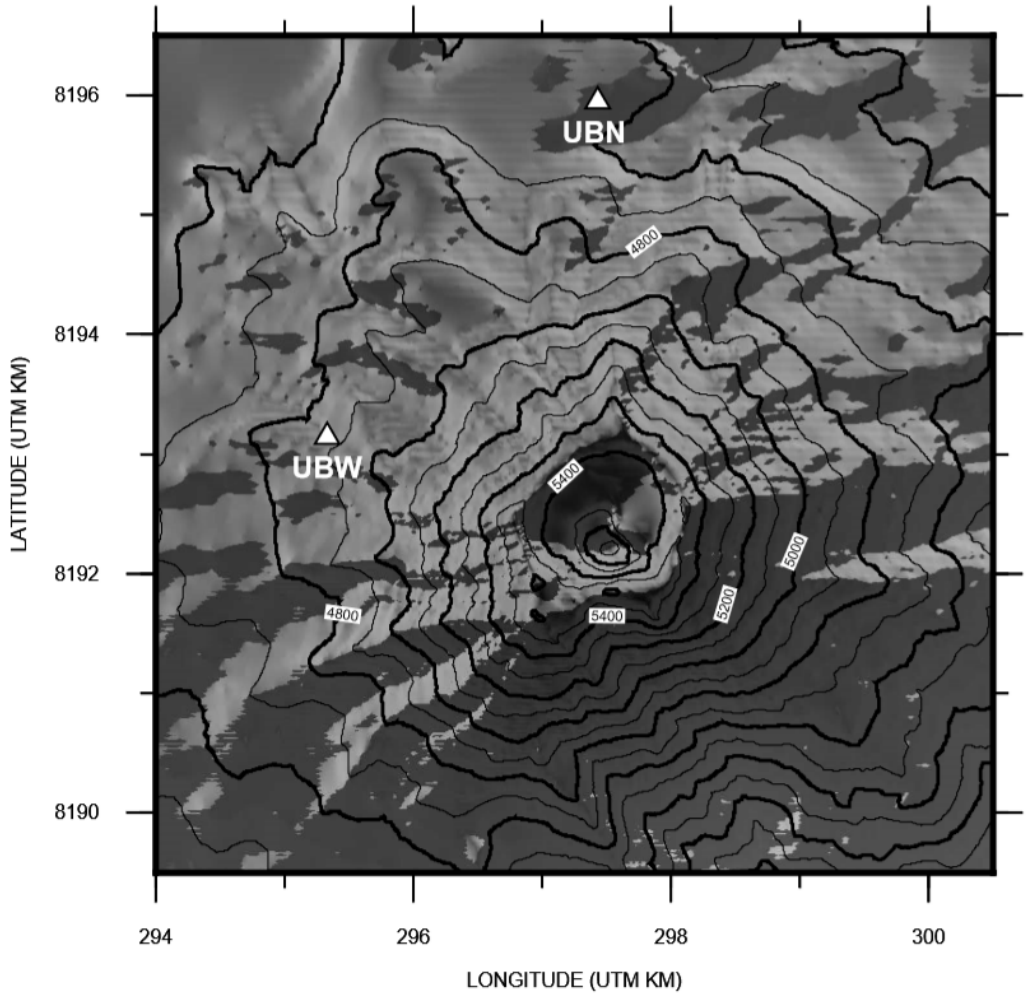
and 2019, with persistent seismicity and observed fumarole, such as the occurrences between 2006 and 2009 (ZANDOMENEGHI *et al.*, 2012; MACEDO *et al.*, 2009).

To monitor the volcanic activity, the IGP built up a seismic telemetry network composed of 6 stations around the volcano (LARA *et al.*, 2020). From May to July 2009, a field seismic experiment was carried out on the Ubinas volcano, where three-component seismometers were deployed in the north and west flanks. The 2009 Ubinas experiment was carried out with international participation of the EU-VOLUME project (MACEDO *et al.*, 2009), in a cooperation between the IGP and the Institut de Recherche pour le Développement (IRD). More details can be found in (INZA *et al.*, 2011).

A catalog of Ubinas seismicity was made by the National Volcanological Center (CENVUL) of the IGP, in the period of 2006 - 2011. This catalog can be found in the IGP website https://repositorio.igp.gob.pe/. According to the catalog, the Ubinas exhibited a diversity of seismic waveform classes, each one associated with a physical process of the volcano. These types of signals were also reported in other volcanoes. The following events have been identified and labeled by the Volcanology experts (CHOUET, 1996; MCNUTT, 2005):

1. Long-period (LP): These events correspond to fluid processes caused by pressure excitation mechanisms, associated with the response of the plumbing system caused by fluid movements (CHOUET, 1996). These fluid movements at the volcano intern cavities result in signals with low frequency components.
2. Tremors (TR): They are characterized by sound waves resonating through the volcano rocks, resulting in rupture. The signals show sustained amplitude that can last from tens of seconds to days. The event is associated with degassing after an eruption (CHOUET, 1996; MCNUTT, 2005).
3. Explosions (EX): Events associated to sudden decompression and the magma fragmentation process (INZA *et al.*, 2011; INZA *et al.*, 2014), which consists in gas exploding from inside to outside the volcano, showing high amplitude signals in time domain. For the Ubinas Volcano, explosions are related to the destruction of the magmatic plug. In (TRAVERSA *et al.*, 2011), it was shown that an overall acceleration of the number of LP leads to eminent explosion.
4. Volcano-tectonic (VT): The VT events are caused by high pressure inside the volcano, resulting in signals with high frequency components. They are associated with stress changes in rocks, induced by magma movement, similar to earthquake events (CHOUET,

Figure 16 – Map of the Ubinas volcano with the UBN and UBW stations.



Source: (PEIXOTO *et al.*, 2021).

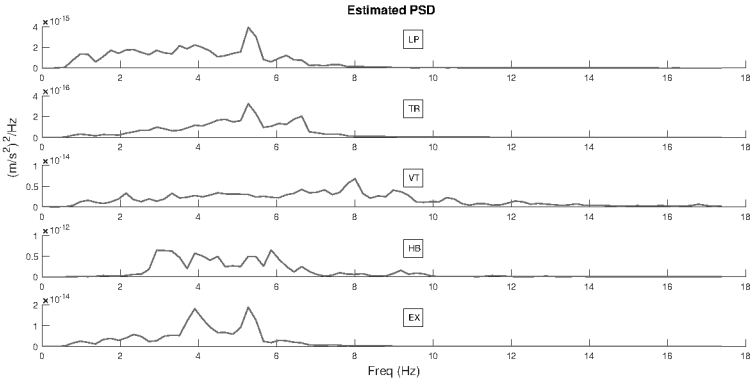
1996; MCNUTT, 2005).

5. Hybrid (HB): They have characteristics of both LP and VT events, where the signal's spectral response corresponds to both high-frequency and low-frequency values (CHOUET, 1996; MCNUTT, 2005).

The database used in this thesis was taken from the IGP catalog, whose seismic events happened during a period of great activity of the Ubinas volcano, in 2009. The signals considered in this work were registered from two seismic stations deployed on the north (3.7 km from the crater) and west (2.7 km from the crater) of the volcano, as shown in Fig. 16, where UBN represents the north station and UBW the west station. Each station is equipped with 3-components (north, east and vertical) broadband seismometers, with a 100 Hz sampling rate and high resolution digitizers Guralp 6TD. As earlier mentioned, it is considered that the signals recorded by different stations, in the same time interval, belong to the same class.

A summary of the main information about the seismic database considered in this

Figure 17 – Estimated PSD of samples of the five classes.



Source: (PEIXOTO *et al.*, 2021).

Table 3 – Seismic database information.

Stations	UBW, UBN
Channels	HHE (east), HHN (north), HHZ (vertical)
Classes	TR (383), LP (397), EX (16), HB (48), VT (12)
N. of Samples	856 third-order $2 \times 3 \times 3000$ tensors

Source: (PEIXOTO *et al.*, 2021).

work is shown in Table 3, with the number of class samples being detailed in parenthesis in the third row. A total of 856 tensor samples are used, each one being organized as $2 \times 3 \times 3000$ third-order tensor, with the first dimension corresponding to the number of sensors, the second dimension the number of channels per sensor and the third dimension corresponds to the number of time-samples per signal. The duration of the signals is set to 30 seconds. As the sampling frequency is equal to 100 Hz, each signal has 3000 time samples. The seismic signals whose duration is less than 30 seconds are completed with zeroes. The total number of signals is then equal to 5136. As it can be viewed, the classes with more samples are the LP and TR, whereas the one with less samples is VT. This is due to the fact that LP and TR events are much more common than the VT and EX events.

The goal of the classification system is to correctly classify the seismic signals into one of the 5 existing classes (TR, LP, EX, HB and VT). Fig. 17 shows the estimated PSD of one sample of the waveforms for each class. As it can be viewed, the classes have different frequency behaviors, which is exploited in the classification process.

3.4 Results

In this section, the results obtained with the database of volcano-seismic events described in Section 3.3 are presented, with the K-fold cross-validation method being used. Some preliminary simulations, whose results are omitted, were carried out in order to find the

best values for the constant C, the number of folds K and the ranks of the SPM and STuM, with $C = 100$, $K = 10$, $Q = 3$ and $Q_1 = 2$, $Q_2 = 3$ and $Q_3 = 5$ providing the best results. When not stated otherwise, these values were used in the rest of the results.

For comparison purposes, the R1-SPM was also tested, as well as the standard PCA and SVM techniques using the vectorization of the tensor samples, which yielded pattern vectors of dimension 336. The proposed approach is also compared with a CNN inspired by the work (CURILEM *et al.*, 2018) and with the method presented in (LARA *et al.*, 2020), where the EMD is performed before attribute extraction, but using the same attributes of the present work. Multiclass SVM and STM were implemented using the *one vs. all* approach.

The results are presented in the form of accuracy, which is defined as the number of correctly classified samples over the total number of samples, execution time, in seconds, of the feature technique plus classifier, and, confusion tables. Later on, the floating point operations per second (FLOPS) count is shown as a metric of computational cost.

The processing times were obtained with an 9^{th} generation Intel Core i3 processor, running MATLAB version 2017b. The SVM functions used to model the classifiers are *fitcsvm* and *predict*, respectively.

3.4.1 MPCA with Full Projection

In this subsection, results using the PCA and MPCA with full projection are presented (PCA-FP and MPCA-FP). Table 4 shows the Accuracy provided by different configurations of classifiers and PCA. It can viewed from this table that the best result was obtained by the MPCA-FP with STuM, which provided an accuracy of 84.5%. This is due to the fact that these techniques are tensorial, which do not break the multidimensional structure of the data. Moreover, among the tensor-based classifiers, the STuM is the one that has the highest number of free parameters. In other words, the Tucker decomposition has a degree of freedom to fit the weight tensor higher than the PARAFAC decomposition.

The better performance of the STuM with respect to the other classifiers is confirmed by comparing the accuracy of the three tensor-based classifiers when using the same type of PCA. Indeed, for the three tested kinds of PCA (MPCA, standard PCA and no PCA), the STuM always provides the best classification rates, in other words, the best accuracy, while the R1-SPM has the worst performance among the tensorial classifiers. The low-rank constraint of the R1-SPM diminishes the computational complexity, however, it does not have the same capacity to fit the

Table 4 – Accuracy for the different system configurations - with PCA-FP and MPCA-FP.

PCA Type	**Classifier**	**Accuracy**	**Execution Time** (s)
None ($2 \times 3 \times 56$)	SVM	75.5%	**2107.2**
	R1-SPM	77.1%	2513.2
	SPM	77.9%	2699.1
	STuM	**78.6**%	3122.3
PCA-FP (336)	SVM	79.7%	**2654.1**
	R1-SPM	80.6%	3043.7
	SPM	81.1%	3123.5
	STuM	**81.8**%	3577.9
MPCA-FP ($2 \times 3 \times 56$)	SVM	81.6%	**8214.8**
	R1-SPM	82.4%	9842.2
	SPM	83.8%	11329.8
	STuM	**84.5**%	13287.0

Source: Author.

weight tensor. Nevertheless, the accuracy of the R1-SPM is significantly higher than the ones of the standard SVM in all the tested cases, with the same type of PCA. The better accuracy provided by the tensor-based classifiers suggest that they have a higher capacity for separating the 5 classes than the conventional SVM.

By comparing the performance of the two types of PCA, it can be viewed that the STuM with MPCA provided a better accuracy than with PCA. In addition, the PCA with STuM showed a slightly better success rate than the case with no PCA. The same conclusion can be drawn for the other two tensor-based classifiers: the best accuracy is achieved with MPCA while the worst is obtained with no PCA. This behavior shows that the MPCA is more efficient in transforming the tensor attributes than the standard PCA, although the vector-based PCA provides a small gain in classification rate with respect to the case with no PCA.

Without using PCA, all algorithms performed worse, thus showing the importance of the PCA and MPCA techniques, even if it does not reduce the dimensionality of the data. The reason for this results is due to the fact the PCA and MPCA techniques project the data into a new basis, extracting useful information and eliminating correlation, enhancing the classification process. Moreover, the tensorial classifiers also perform better than the SVM because they provide better subspace representation, alleviating the overfitting and curse of dimensionality.

Regarding the execution times, in seconds, presented in Table 4, it is shown that the fastest of the presented scenarios was without PCA, whereas the slowest was with MPCA plus the STuM classifier. This clearly shows that better accuracy are achieved at the cost of processing speeds. It is also evident that the tensorial classifiers demand more execution times in comparison to the SVM technique.

3.4.2 MPCA with Dimensionality Reduction

In this subsection, results using the PCA and MPCA with dimensionality reduction are presented (PCA-DR and MPCA-DR). Many tests were carried out in order to find the number of the MPCA components that provides the highest success rate, the best results being obtained with $R_1 = 2$, $R_2 = 2$ and $R_3 = 15$, i.e. $\mathscr{Y}_p \in \mathbb{R}^{2\times2\times15}$. Given that the original data have dimensions $2 \times 3 \times 56$, one can see that no dimensionality reduction was made at the first dimension, which is associated with the station number. This is due to the fact that the two eigenvalues of the first mode, equal to 1.95×10^{17} and 0.90×10^{17}, have significant values. Hence, ignoring one of the eigenvalues would result in significant loss of information. This means that both the stations provide relevant information for the system.

Regarding the second dimension, associated with the sensor channel, using only the first 2 components provided a better accuracy than using 3 components in preliminary tests. This is due to the fact that the eigenvalues associated with the second mode are equal to 3.03×10^{17}, 0.86×10^{17} and 0.01×10^{17}, which allows ignoring the last eigenvalue. Moreover, a significant reduction in the dimensionality of the third mode, associated with the features, was also possible, without significant performance losses. Indeed, reducing the number of eigenvalues from 56 to 15 corresponds to maintain 92% of the data variance. For the standard PCA, the best results were obtained with the first 60 components.

Table 5 shows the accuracy provided by different configurations of classifiers, PCA-DR and MPCA-DR, with the table organized exactly as Table 4. However, in Table 5, the results with no PCA are not shown, to avoid redundancy. Comparing the results of Tables 4 and 5, it can be viewed that the MPCA-DR provided higher accuracy than the MPCA-FP, for all the tested classifiers. Similarly, PCA-DR provided higher success rates than the PCA-FP also for all tested classifiers. This result shows that the dimensionality reduction, in addition to decreasing the number of parameters, improves the classification rates, especially in the tensor-based case.

Moreover, we can draw from 5 the same conclusions that were drawn for Table 4. Indeed, the best result was obtained by the MPCA-DR with STuM, which provided an accuracy of 89.5%. In addition, the tensor-based classifiers performed better than the SVM, with the STuM providing the best classification rate and the R1-SPM giving the lowest success rate among the tensorial classifiers. Besides, the MPCA-DR outperformed the PCA-DR for all the classifiers. Also, the SVM showed a slight increase in accuracy when coupled with the MPCA-DR and tensorial parameters extraction, in comparison to the results of Table 4, again showing the better

Table 5 – Accuracy for the different system configurations - with PCA-DR and MPCA-DR.

PCA Type	**Classifier**	**Accuracy**	**Execution Time** (s)
PCA-DR (60)	SVM	81.9%	**824.1**
	R1-SPM	82.9%	1213.1
	SPM	83.0%	1422.8
	STuM	**83.2**%	1798.9
MPCA-DR ($2 \times 2 \times 15$)	SVM	84.7%	**3664.4**
	R1-SPM	86.0%	3998.2
	SPM	86.7%	4359.9
	STuM	**89.5**%	4881.2

Source: Author.

effectiveness of the MPCA-DR in comparison to the PCA-DR.

The execution times shown in Table 5 show that both PCA-DR and MPCA-DR achieved faster processing speeds in comparison to the results of the PCA-FP and MPCA-FP showed in Table 4. This is due the fact that the reduced dimensionality of the data demands less processing, facilitating the classification process, thus, the framework runs faster. We can see clearly a trade-off between processing time and accuracy.

Furthermore, in Table 6, the accuracy and processing times of the proposed approach, with the StuM classifier, is compared with the one provided by the method presented in (LARA *et al.*, 2020), where the EMD is performed before the attribute extraction step, combined with the standard SVM. This method is denoted by EMD + SVM. Table 6 also compares the proposed method with a deep learning approach, using a CNN with design inspired by (CURILEM *et al.*, 2018). These two techniques are implemented using the same attributes of the present work, but vectorized, in the following cases: no PCA, PCA-DR and MPCA-DR.

It can be observed from Table 6 that the proposed approach using the STuM provided the best classification rates for the three cases (no PCA, PCA-DR and MPCA-DR) and, once again, the MPCA-DR combined with the STuM reached the best accuracy. This is due to the fact that the techniques of (LARA *et al.*, 2020) and (CURILEM *et al.*, 2018) break the tensor structure of the data. The MPCA-DR has also improved the performance of the EMD + SVM and CNN methods, with the worst accuracy being achieved by the CNN technique.

Tab. 6 also showed interesting results regarding the execution times. As can be seen, the proposed framework presented faster execution times, in comparison to the ones of (LARA *et al.*, 2020) and (CURILEM *et al.*, 2018). This clearly shows the overall better performance of the proposed framework on all scenarios, where it achieves higher accuracy and faster processing times.

Next, Table 7 shows the confusion matrix for the best configuration: MPCA-DR

Table 6 – Comparison between the proposed approach and the ones of (LARA *et al.*, 2020; CURILEM *et al.*, 2018) in terms of Accuracy.

PCA Type	**Classifier**	**Accuracy**	**Execution Time** (s)
None	EMD + SVM (LARA *et al.*, 2020)	78.1%	3404.2
	CNN (CURILEM *et al.*, 2018)	51.5%	7099.7
	STuM	**78.6%**	**3122.3**
PCA-DR	EMD + SVM (LARA *et al.*, 2020)	82.9%	2433.1
	CNN (CURILEM *et al.*, 2018)	56.8%	4194.2
	STuM	**83.2%**	**1798.9**
MPCA-DR	EMD + SVM (LARA *et al.*, 2020)	85.5%	4511.3
	CNN (CURILEM *et al.*, 2018)	62.35%	5355.6
	STuM	**89.5%**	**4881.2**

Source: Author.

Table 7 – Confusion Matrix for the STuM with MPCA-DR.

		True class					
		EX	HB	LP	TR	VT	Overall
Predicted class	EX	**11**	0	2	0	0	
	HB	0	**10**	11	0	5	
	LP	0	36	**377**	22	0	
	TR	0	0	0	**361**	0	
	VT	5	2	7	0	**7**	
Accuracy(%)		68.7	20.8	94.9	94.2	58.3	**89.5**

Source: Author.

Table 8 – Confusion Matrix for the STuM with MPCA-DR, without preprocessing.

		True class					
		EX	HB	LP	TR	VT	Overall
Predicted class	EX	**10**	0	1	0	0	
	HB	0	**0**	0	0	7	
	LP	1	46	**384**	117	0	
	TR	0	0	3	**260**	0	
	VT	5	2	9	6	**5**	
Accuracy(%)		62.5	0	96.7	67.8	41.6	**76.9**

Source: (PEIXOTO *et al.*, 2021).

with STuM. It can be viewed from this table that the classes TR and LP provided the highest accuracy, while the HB showed the worst success rate. This is due to the fact that the classes TR and LP are very distinct from the EX and VT classes, making it easier to the classifier to separate them. Also, these two classes have the biggest sample size in comparison to the EX and VT classes. The HB class, as described in Section 3.3, has characteristics of both the LP and VT classes, which explains the misclassification of its samples to one of these classes.

3.4.3 Effects of the Preprocessing Steps and Tensor Ranks

In this subsection, the effects of the preprocessing steps and the choice of the tensor ranks are evaluated. Table 8 shows the confusion matrix obtained by the MPCA-DR with STuM

Table 9 – Accuracy for different rank configurations - with MPCA-DR and STuM.

Rank Configuration	Accuracy
$Q_1 = 1, Q_2 = 2, Q_3 = 3$	50.7%
$Q_1 = 2, Q_2 = 1, Q_3 = 3$	51.8%
$Q_1 = 2, Q_2 = 2, Q_3 = 2$	55.6%
$Q_1 = 2, Q_2 = 2, Q_3 = 3$	67.5%
$Q_1 = 2, Q_2 = 3, Q_3 = 3$	78.4%
$Q_1 = 2, Q_2 = 2, Q_3 = 4$	61.0%
$Q_1 = 2, Q_2 = 3, Q_3 = 4$	85.5%
$Q_1 = 2, Q_2 = 2, Q_3 = 5$	69.3%
$Q_1 = 2, Q_2 = 3, Q_3 = 5$	**89.5**%

Source: Author.

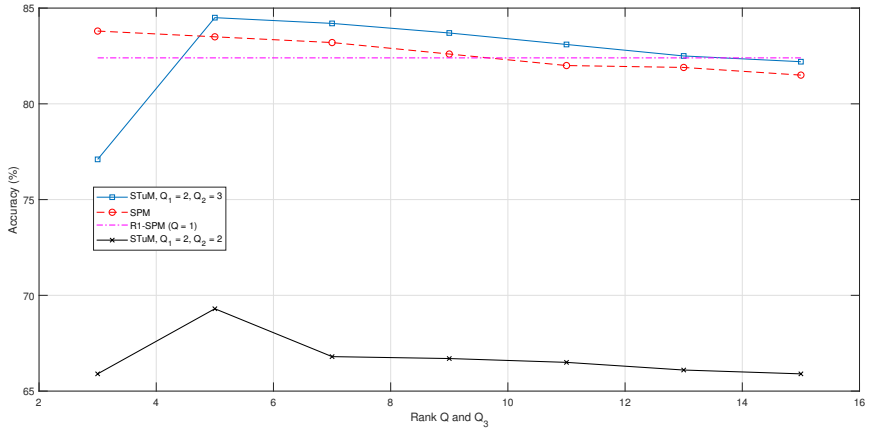
without preprocessing the data. The goal is to illustrate the effect of the preprocessing steps, including the instrumental correction. Comparing the results of Tables 7 and 8, it can be viewed that this preprocessing has a deep impact on the success rate. Indeed, without these steps, a significant reduction in the success rate is observed for all the classes. The main reason to this result is the fact that valuable information related to the physical energy of the signals is lost when the instrumental correction is not applied. This step gives the energy of the signals a physical sense, providing valuable information to the classifiers.

In order to show the impact of the trilinear rank on the system performance, Table 9 shows the accuracy provided by the STuM with MPCA-DR for some configurations of (Q_1, Q_2, Q_3). Other configurations were tested, however, to improve the presentation, this table shows only some of the obtained results, with Figures 19 exploring further in these values. It can be viewed from these results that the ranks of the Tucker decomposition have a very significant impact on the accuracy. The worst result was obtained with $Q1 = 1, Q2 = 2, Q3 = 3$, which indicates that using small values for the tensor ranks may limit the ability of the Tucker decomposition fit the weight tensor.

The last results of this subsection are presented in Figures 18 and 19, which shows the impact of the rank Q of the SPM and rank Q_3 of the STuM, for the MPCA-FP and MPCA-DR cases, respectively, which aims to analyze the impact of the ranks on the accuracy of said classifiers.

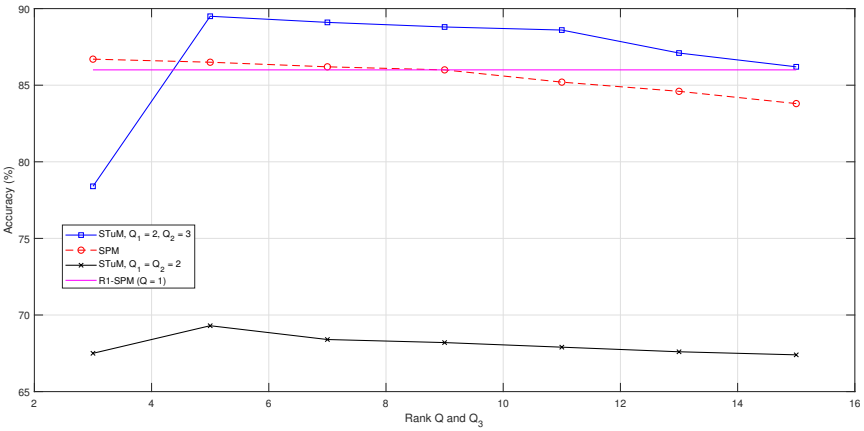
As Figures 18 and 19 showed, higher values of Q and Q_3 resulted, in general, worse classification performance, for both SPM and STuM, with $Q = 3$ and $Q_3 = 5$ providing the best results. The STuM, however, showed more variation in accuracy, with rank Q_2 improving the success rate by more than 20% when changed from 2 to 3, thus validating the choice made of

Figure 18 – Impact of the ranks Q and Q_3 of the STM-based techniques in classification rates, for the MPCA-FP case.



Source: Author.

Figure 19 – Impact of the ranks Q and Q_3 of the STM-based techniques in classification rates, for the MPCA-DR case.



Source: Author.

ranks $Q_1 = 2$, $Q_2 = 3$ and $Q_3 = 5$ for the previous results. From Figs. 18 and 19, its possible to see that values of Q and Q_3 approximately higher than 8 may cause overfitting, as the accuracy drops. Comparing both figures 18 and 19, it is clear that the impact of the ranks is similar in both cases, providing lower accuracy as the rank values increase.

3.4.4 Computational Cost Analysis of the Framework

Now, the computational cost of the framework techniques is analyzed in terms of time complexity, which is considered as a function of the input parameters, using the big-O notation. The time complexity is estimated by counting the number of elementary operations performed by the algorithm, considering the worst-case time complexity, which is the maximum amount of elementary operations required for inputs of a given size.

Table 10 – Time-complexity of the framework techniques, in big-O notation.

MDFT	$\mathcal{O}(DlogD)$
PCA-FP	$\mathcal{O}(2PI_1^2I_2^2I_3^2 + I_1^3I_2^3I_3^3)$
PCA-DR	$\mathcal{O}(PI_1^2I_2^2I_3^2 + I_1^3I_2^3I_3^3 + PI_1I_2I_3R_1R_2R_3)$
MPCA-FP	$\mathcal{O}(2PI_1I_2I_3[I_1 + I_2 + I_3] + I_1^3 + I_2^3 + I_3^3)$
MPCA-DR	$\mathcal{O}(PI_1I_2I_3[I_1 + I_2 + I_3] + I_1^3 + I_2^3 + I_3^3 + PI_1I_2I_3[R_1 + R_2 + R_3])$
SVM	$\mathcal{O}(DP^2)$
R1-SPM	$\mathcal{O}(P^2[R_1 + R_2 + R_3]N_{it})$
SPM	$\mathcal{O}(QP^2[R_1 + R_2 + R_3]N_{it})$
STuM	$\mathcal{O}(P^2[R_1Q_1 + R_2Q_2 + R_3Q_3 + Q_1Q_2Q_3]N_{it})$

Source: Author.

Then first, to illustrate the time-complexity demand of the used techniques, they are organized in Table 10, in big-O notation. Then, Table 10 shows the time-complexity for the MDFT, PCA, trilinear MPCA, or TPCA, R1-SPM, SPM and STuM, with P being the total number of samples, I_1, I_2 and I_3 the original dimensions of the input tensors (full projection scenario), R_1, R_2 and R_3 the dimensions after dimensionality reduction and D the total vectorized dimension excluding samples ($I_1I_2I_3$ or $R_1R_2R_3$). The number of iterations of the tensor-based classifiers algorithms is given by N_{it}, which variate, in average, between 5 and 30 iterations.

The expressions exhibited in Table 10 confirms the execution times shown in Tables 4-5, in which techniques with increased time-complexity demand tend to run slower, whereas the techniques with smaller time-complexity generally run faster. Also, it is important to note the increased complexity of the PCA-FP and MPCA-FP in comparison to the PCA-DR and MPCA-DR, which is reflected in higher execution times.

In Table 11, the FLOPS, which is a measure of computer performance, of several algorithms are presented, including the ones of the proposed framework. From Table 11, it can be viewed that the linear SVM is the less demanding in terms of FLOPS, whereas the STuM is the most complex method, due to the fact that the STuM has more parameters to estimate. Moreover, the deep learning CNN provided a computational cost roughly close to the SPM and STuM. This result pictures a trade-off between accuracy performance and computational cost of these techniques, for instance, the R1-SPM versus the SPM, in which the first runs faster and demands less computational cost, whereas the SPM achieves higher classification rates. The same analysis can be applied to the SPM versus STuM.

In the next chapter, some of the framework techniques are further explored with a novel approach proposed in this thesis, generating more results that validate tensor learning algorithms.

Table 11 – Floating Point Operations per Second (FLOPS) counts for several classifiers.

Classifier	**FLOPS**
SVM	1.1×10^8
R1-SPM	1.4×10^8
CNN (CURILEM *et al.*, 2018)	1.5×10^9
SPM	2.1×10^9
STuM	2.9×10^9

Source: (PEIXOTO *et al.*, 2021).

3.5 Conclusions

In this chapter, a tensorial framework was proposed for classifying volcano-seismic signals into five different classes using tensor learning techniques such as the MPCA and STMs. The proposed method integrates feature extraction, dimensionality reduction and classification into a framework that can be fed with multidimensional data. The database used in this work consists in three-dimensional data samples recorded during a period of great activity of the Ubinas volcano, Peru, in 2009. The tensor structure of the patterns, organized as *stations* $\times$ *channels* $\times$ *features*, is built by exploring the use of multiple multichannel triaxial sensors, operating simultaneously in two seismic stations.

The results showed the very significant gains in performance provided by the tensorial framework, when compared with their vector-based counterparts. The best result was obtained with the STuM classifier along with the MPCA-DR. The best accuracy provided by the tensor-based configurations is due to the fact they preserve the multidimensional structure of the data, avoiding the drawbacks of tensor vectorization. The best performance of the STuM with respect to the SPM is due to the fact that it has a greater degree of freedom for fitting the weight tensor. The tensor learning approach also surpassed, in terms of accuracy, a deep learning CNN method and the technique of (LARA *et al.*, 2020). The obtained results also showed that the preprocessing, which includes the instrumental correction, has a great impact on the data classification, as it gives to the energy of the signals a physical sense.

4 MULTILINEAR DIMENSIONALITY REDUCTION

In this chapter, a technique for reduction of dimensionality of tensor data denoted by LC-MDR is proposed. The method optimizes a cost function that takes into account the data correlation, generating variables with low correlation. The LC-MDR fits the input data correlation into a new tensor decomposition called EONPD, a proposed extension of the NPD for higher-order tensors.

The presented EONPD holds for even-order tensors and it has a core tensor that interacts with all the factor matrices. A complete description of the EONPD is given, with both analytical and recursive expressions being derived, using scalar and tensor notations. Moreover, a generalization of the EONPD, denoted by HONPD, is also presented, assuming the nesting of PARAFAC tensors of generic orders.

Contrarily to existing approaches that use orthogonal transformation matrices, the LC-MDR does not have impose this constraint on the transformation matrices in order to minimize the output correlation. The LC-MDR method aims to overcome this drawback of the MPCA by optimizing a cost function that takes into account the correlation of the data. The LC-MDR tries to fit the input data correlation into a new tensor decomposition denoted by EONPD, presented in the next section, which is an extension of the NPD for higher-order tensors.

The proposed technique was evaluated in a classification system of volcano-seismic events, using a the same data as in Chapter 3, which was obtained from the Ubinas volcano, in 2009, using a full tensorial classification framework. The results showed significant gains of the LC-MDR when compared with concurrent techniques, in terms of accuracy, data correlation and processing time.

The contents of this chapter are presented as follows. First, we outline the motivations for dimensionality reduction and the development of the LC-MDR technique. Next, the proposed EONPD and HONPD decompositions are presented. Further on, the LC-MDR formulation and its algorithm is presented and the obtained validation results are discussed.

4.1 Motivation

In many ML and signal processing problems, it is common to have datasets with a large numbers of variables. However, dealing with high-dimensional data have some drawbacks.

Firstly, the computational complexity of the techniques become higher, as there is a high number of features to be processed. Another problem is the so-called curse of dimensionality (BELARBI *et al.*, 2017), which refers to several phenomena that arise when dealing with high-dimensional data. Indeed, when the number of features is bigger or closer to the sample size, the ML techniques tend to perform poorly.

Dimensionality reduction techniques can be applied to avoid these issues by transforming the data from a high-dimensional space into a low-dimensional space without losing significant information of the original dataset. When the input samples are correlated, they may be confined into a subspace, where an adequate low-dimensional space representation is possible. Within this context, linear techniques such as the NMF (PAUCA *et al.*, 2006), LDA (THARWAT *et al.*, 2017), CCA (HARDOON *et al.*, 2004) and PCA (WOLD *et al.*, 1987) are popular solutions for dimensionality reduction. In particular, the PCA is the most commonly used dimensionality reduction technique. Indeed, this method is a powerful tool that transforms a set of correlated data into uncorrelated data using an orthonormal basis (RODARMEL; SHAN, 2002).

Within the context of tensor learning, dimensionality reduction for tensor patterns is of great interest, with the MPCA (LU *et al.*, 2008; LU *et al.*, 2006) being the most popular alternative. The MPCA is an extension of the PCA for tensor patterns that uses the HOSVD, also known as multilinear SVD, for performing orthonormal multilinear transformation. The MPCA is a powerful method for dimensionality reduction and feature extraction in tensor patterns, often used in classification tasks (LU *et al.*, 2011). The MPCA has a wide range of applications, such as gait, face, fingerprint, age, and fault recognition (LU *et al.*, 2008; LU *et al.*, 2006).

On the other hand, contrarily to the standard PCA that generates fully uncorrelated output data, the MPCA is not able to create perfectly uncorrelated variables. Having uncorrelated data may be beneficial in several kinds of problems, as, for instance, in the ML tasks of classification, regression and clustering (LU *et al.*, 2011). Some works of the literature highlights the importance of correlation reduction in classification tasks, such as in (FU; WANG, 2003), where a separability-correlation measure approach, to rank the importance of attributes, was proposed, improving classification performance. Again in (EL-HASNONY *et al.*, 2015), correlation-based feature selection is performed, eliminating features that do not provide contribution to class differentiation. Also, in (FU; HUANG, 2008), tensor correlation analysis is performed, improving the classification performance.

In order to avoid this correlation problem of the MPCA, this thesis proposes a multilinear dimensionality reduction technique called LC-MDR. The proposed technique optimizes a cost function that takes into account the correlation of the input and output data, generating variables with much less correlation than the MPCA. In particular, the LC-MDR tries to fit the input data correlation into a new tensor decomposition called EONPD, which is itself an extension of the NPD for higher-order tensors.

The LC-MDR generates output tensor samples with low correlation and low dimensions, which can be used for dimensionality reduction and feature extraction in tensor patterns. As well as the MCPA, the LC-MDR carries out a multilinear projection with dimensionality reduction, capturing most of the data information. However, while the MPCA is based on the HOSVD and it uses orthogonal transformation matrices, in contrast, the LC-MDR is based on the EONPD and the orthogonality constraint on the transformation matrices is removed, in order to minimize the output correlation.

Ultimately, the motivation to apply and validate the proposed LC-MDR technique with the volcano-seismic database of Chapter 3 is the following. Although the MPCA provided interesting results and good dimensionality reduction, data correlation is still high, as shown later on. This high correlation of the data, if reduced, could provided higher accuracy by the classifiers. Also, the MPCA technique demands high computational cost and high execution times. Within this scope, the LC-MDR was proposed to solve this issues, reducing data correlation, achieving greater classification rates and speeding up the execution times by requiring less computational power.

4.2 Proposed Nested PARAFAC Decompositions for Higher-Order Tensors

There are several ways of extending the NPD for higher-order tensors. Indeed, when multiple nested PARAFAC tensors are considered, different decompositions are obtained depending on which factors nest the PARAFAC tensors and on how the extra indices of the higher-order tensor are concatenated to form the nested PARAFAC tensors.

In this section, an extension of the NPD for higher-order tensors, denoted by EONPD, is presented. The described decomposition holds for even-order tensors, and it assumes that the indices associated with the tensor rankings are always concatenated in a same dimension, forming a new tensor that will nest the next PARAFAC tensor. That leads to a core tensor that interacts with all the factor matrices, as it will be detailed in the sequel.

An extension of the NPD for higher-order tensors, denoted Generalized Nested PARAFAC Decomposition (GNPD), was used in (FREITAS *et al.*, 2017) to model a communication system with multiple relays. However, the nesting structure used by the GNPD is different from the one of the EONPD, leading to a completely different tensor decomposition. Moreover, in (FREITAS *et al.*, 2017), only a recursive description of the EONPD is given, with no analytic formulas being provided. Besides, only the matrix formulation is given in (FREITAS *et al.*, 2017), with no scalar and tensor descriptions being presented. On the other hand, in the sequel, a complete description of the EONPD is given, with both analytical and recursive expressions being derived, using scalar and tensor notations.

In addition, at Subection 4.2.2, we present a generalization of the EONPD, denoted by HONPD, assuming the nesting of PARAFAC tensors of generic orders. The HONPD is more generic than the GNPD and EONPD, as these two decompositions are restricted to the nesting of third-order PARAFAC tensors.

4.2.1 Even-Order Nested PARAFAC Decomposition (EONPD)

The EONPD of a N^{th}-order tensor $\mathscr{Y} \in \mathbb{C}^{I_1 \times I_2 \times \ldots \times I_N}$, for N even, is defined by:

$$y_{i_1,i_2,i_3,\ldots,i_N} = \sum_{q_1=1}^{Q_1} \cdots \sum_{q_{N/2}=1}^{Q_{N/2}} b_{q_1,\ldots,q_{N/2}} a^{(1)}_{i_1,q_1} a^{(2)}_{i_2,q_1} a^{(3)}_{i_3,q_2} a^{(4)}_{i_4,q_2} \ldots a^{(N-1)}_{i_{N-1},q_{N/2}} a^{(N)}_{i_N,q_{N/2}}, \tag{4.1}$$

where $\mathscr{B} \in \mathbb{C}^{Q_1 \times \ldots \times Q_{N/2}}$ is the core tensor, $Q_1,\ldots,Q_{N/2}$ are the ranks of the decomposition and $\mathbf{A}^{(n)} \in \mathbb{C}^{I_n \times Q_k}$, for $n = 1,\ldots,N/2-1$, is a factor matrix, with $k = n/2$ for n even and $k = (n+1)/2$ for n odd. Concatenating the indices i_3, i_4, $\ldots$ i_N into one index, in the following way: $y_{i_1,i_2,j_1} = y_{i_1,i_2,i_3,\ldots,i_N}$, with $j_1 = (i_3-1)I_4 \ldots I_N + \ldots + (i_{N-1}-1)I_N + i_N$, eq. (4.1) can be written as a tridimensional PARAFAC model:

$$y_{i_1,i_2,j_1} = \sum_{q_1=1}^{Q_1} a^{(1)}_{i_1,q_1} a^{(2)}_{i_2,q_1} w^{(1)}_{j_1,q_1}, \tag{4.2}$$

where the matrix $\mathbf{W}^{(1)} \in \mathbb{C}^{J_1 \times Q_1}$, for $J_1 = I_3 \ldots I_N$, is the mode-1 unfolded matrix of the tensor $\mathscr{W}^{(1)} \in \mathbb{C}^{Q_1 \times I_3 \times \ldots \times I_N}$, given by:

$$w^{(1)}_{q_1,i_3,i_4,\ldots,i_N} = \sum_{q_2=1}^{Q_2} \cdots \sum_{q_{N/2}=1}^{Q_{N/2}} b_{q_1,\ldots,q_{N/2}} a^{(3)}_{i_3,q_2} a^{(4)}_{i_4,q_2} \ldots a^{(N-1)}_{i_{N-1},q_{N/2}} a^{(N)}_{i_N,q_{N/2}}. \tag{4.3}$$

The tensor $\mathscr{W}^{(1)}$ can also be unfolded into a tridimensional PARAFAC tensor, with factor matrices $\mathbf{A}^{(3)} \in \mathbb{C}^{I_3 \times Q_2}$, $\mathbf{A}^{(4)} \in \mathbb{C}^{I_4 \times Q_2}$ and $\mathbf{W}^{(2)} \in \mathbb{C}^{J_2 \times Q_2}$, with $J_2 = Q_1 I_5 \ldots I_N$. This

process goes on, each time forming a different third-order PARAFAC model by unfolding a tensor denoted by $\mathscr{W}^{(n)} \in \mathbb{C}^{Q_1 \times \ldots \times Q_n \times I_{2n+1} \times \ldots \times I_N}$, for $n = 0, ..., N/2-1$, defined as:

$$w^{(n)}_{q_1,\ldots,q_n,i_{2n+1},\ldots,i_N} = \sum_{q_{n+1}=1}^{Q_{n+1}} \ldots \sum_{q_{N/2}=1}^{Q_{N/2}} b_{q_1,\ldots,q_{N/2}} a^{(2n+1)}_{i_{2n+1},q_{n+1}} a^{(2n+2)}_{i_{2n+2},q_{n+1}} \ldots a^{(N-1)}_{i_{N-1},q_{N/2}} a^{(N)}_{i_N,q_{N/2}}, \tag{4.4}$$

which can be re-expressed recursively as:

$$w^{(n)}_{q_1,\ldots,q_n,i_{2n+1},\ldots,i_N} = \sum_{q_{n+1}=1}^{Q_{n+1}} a^{(2n+1)}_{i_{2n+1},q_{n+1}} a^{(2n+2)}_{i_{2n+2},q_{n+1}} w^{(n+1)}_{q_1,\ldots,q_{n+1},i_{2n+3},\ldots,i_N}, \tag{4.5}$$

where $w^{(0)}_{i_1,\ldots,i_N} = y_{i_1,i_2,i_3,\ldots,i_N}$ and $w^{(N/2)}_{q_1,\ldots,q_{N/2}} = b_{q_1,\ldots,q_{N/2}}$.

Indeed, by concatenating the indices $q_n, \ldots, q_1, i_{2n+3}, \ldots, i_N$ into one index, in the following way: $w^{(n)}_{i_{2n+1},i_{2n+2},j_{n+1}} = w^{(n)}_{q_1,\ldots,q_n,i_{2n+1},\ldots,i_N}$, with $j_{n+1} = (q_n - 1)\ Q_{n-1} \ \ldots \ Q_1\ I_{2n+3}$ $\ldots\ I_N + \ldots + (q_1 - 1)\ I_{2n+3}\ \ldots\ I_N + (i_{2n+3} - 1)\ I_{2n+4}\ \ldots\ I_N + \ldots + (i_{N-1} - 1)I_N + i_N$, for $n = 1, ..., N/2-1$, eq. (4.5) can be re-expressed as the following tridimensional PARAFAC model:

$$w^{(n)}_{i_{2n+1},i_{2n+2},j_{n+1}} = \sum_{q_{n+1}=1}^{Q_{n+1}} a^{(2n+1)}_{i_{2n+1},q_{n+1}} a^{(2n+2)}_{i_{2n+2},q_{n+1}} w^{(n+1)}_{j_{n+1},q_{n+1}}. \tag{4.6}$$

Note that the EONPD assumes that the indices $q_1, ..., q_N$ associated with the tensor rankings, as well as the indices $i_{2n+3}, ..., i_N$, are always concatenated in the same dimension, forming a new tensor that will nest the next PARAFAC tensor.

It can be viewed from (4.6) that, for each value of n in $[0 \quad N/2-1]$, a third-order PARAFAC tensor $\mathscr{W}^{(n)} \in \mathbb{C}^{I_{2n+1} \times I_{2n+2} \times J_{n+1}}$, for $J_{n+1} = Q_n \ldots Q_1 I_{2n+3} \ldots I_N$, is obtained, with factor matrices given by $\mathbf{A}^{(2n+1)} \in \mathbb{C}^{I_{2n+1} \times Q_{n+1}}$, $\mathbf{A}^{(2n+2)} \in \mathbb{C}^{I_{2n+2} \times Q_{n+1}}$ and $\mathbf{W}^{(n+1)} \in \mathbb{C}^{J_{n+1} \times Q_{n+1}}$, where $\mathbf{W}^{(n+1)}$ is the mode-$(n+1)$ unfolded matrix of $\mathscr{W}^{(n+1)}$. In the last tridimensional PARAFAC model, that is, for $n = N/2-1$, we have $\mathbf{W}^{(N/2)} = \mathbf{B}^{(N/2)} \in \mathbb{C}^{J_{N/2} \times Q_{N/2}}$, where $\mathbf{B}^{(N/2)}$ is the mode-$(N/2)$ unfolded matrix of $\mathscr{B}$, with $J_{N/2} = Q_{N/2-1} \ldots Q_1$.

The above recursive formulation of the EONPD shows that the tensor $\mathscr{Y}$ can be viewed as the PARAFAC nesting of the following $N/2$ third-order tensors: $\mathscr{Y} = \mathscr{W}^{(0)}$, $\mathscr{W}^{(1)}$, …, $\mathscr{W}^{(N/2-1)}$ and $\mathscr{B} = \mathscr{W}^{(N/2)}$.

The EONPD can also be expressed in a tensor notation, using the contraction operation. Indeed, the tensor $\mathscr{Y}$ can be written as the multiple contractions over the modes q_1, q_2, q_3, …, q_N, as follows:

$$\mathscr{Y} = \mathscr{B} \underset{q_1}{*} \mathscr{R}^{(1)} \underset{q_2}{*} \mathscr{R}^{(2)} \ldots \underset{q_{N/2}}{*} \mathscr{R}^{(N/2)}, \tag{4.7}$$

where $\mathscr{R}^{(n)} \in \mathbb{C}^{I_{2n-1} \times I_{2n} \times Q_n}$, for $n = 1, ..., N/2$, is defined as:

$$r^{(n)}_{i_{2n-1}, i_{2n}, q_n} = a^{(2n-1)}_{i_{2n-1}, q_n} a^{(2n)}_{i_{2n}, q_n}. \tag{4.8}$$

The mode-3 unfolded matrix of the tensor $\mathscr{R}^{(n)}$ is given by $\mathbf{A}^{(2n-1)} \odot \mathbf{A}^{(2n)}$.

Similarly to the NPD, where $\mathbf{B}$ is a factor that interacts with the two PARAFAC tensors $\mathscr{R}^{(1)}$ and $\mathscr{R}^{(2)}$, in the EONPD, the tensor $\mathscr{B}$ interacts with all the tensors $\mathscr{R}^{(1)}$,...,$\mathscr{R}^{(N)}$ by means of the contraction operation in the indices q_1, q_2, ..., q_N. Note also that, contrarily to other nested decompositions, such as the NPD, the Nested Tucker Decomposition (NTD) (FAVIER *et al.*, 2016) and the High-Order Nested Tucker Decomposition (HONTD) (ROCHA *et al.*, 2019), the EONPD does not follow the Tensor-Train (TT) structure (OSELEDETS; TYRTYSHNIKOV, 2010; OSELEDETS, 2011). In order to follow the TT structure, the tensor $\mathscr{B}$ would have to verify the following structure: $b_{q_1,...,q_{N/2}} = b_{q_1,q_2} b_{q_2,q_3} \dots b_{q_{N/2-1},q_{N/2}}$.

4.2.2 Higher-Order Nested PARAFAC Decomposition (HONPD)

The EONPD can be generalized to the case of a nesting of M PARAFAC tensors of generic orders. This decomposition will be denoted by HONPD. The HONPD of the tensor $\mathscr{Y} \in \mathbb{C}^{I_1 \times ... \times I_{S_N}}$ is defined by:

$$y_{i_1,...,i_{S_M}} = \sum_{q_1=1}^{Q_1} ... \sum_{q_M=1}^{Q_M} b_{q_1,...,q_M} a^{(1)}_{i_1,q_1} ... a^{(N_1)}_{i_{N_1},q_1} a^{(N_1+1)}_{i_{N_1+1},q_2} ... a^{(N_1+N_2)}_{i_{N_1+N_2},q_2} ... a^{(S_{M-1}+1)}_{i_{S_{M-1}+1},q_M} ... a^{(S_M)}_{i_{S_M},q_M}, \tag{4.9}$$

where $S_m = N_1 + \ldots + N_m$, for $m = 1, ..., M$, which can be re-expressed recursively as:

$$w^{(m)}_{q_1,...,q_m,i_{S_{m+1}},...,i_{S_M}} = \sum_{q_{m+1}=1}^{Q_{m+1}} a^{(S_m+1)}_{i_{S_m+1},q_{m+1}} \dots a^{(S_{m+1})}_{i_{S_{m+1}},q_{m+1}} w^{(m+1)}_{q_1,...,q_{m+1},i_{S_{m+1}+1},...,i_{S_M}}, \tag{4.10}$$

for $m = 0, ..., M$, where $w^{(0)}_{i_1,...,i_{S_M}} = y_{i_1,...,i_{S_M}}$ and $w^{(M)}_{q_1,...,q_M} = b_{q_1,...,q_M}$.

By concatenating the indices $q_m, \dots, q_1, i_{S_{m+1}+1}, \dots, i_{S_M}$ into one index in the following way: $w^{(m)}_{i_{S_m+1},...,i_{S_{m+1}},j_{m+1}} = w^{(m)}_{q_1,...,q_m,i_{S_m+1},...,i_{S_M}}$, with $j_{m+1} = (q_m - 1)\, Q_{m-1} \,...\, Q_1\, I_{S_{m+1}+2} \,...\, I_{S_M} + ... + (q_1 - 1)\, I_{S_{m+1}+1} \,...\, I_{S_M} + (i_{S_{m+1}+1} - 1) I_{S_{m+1}+2} \,...\, I_{S_M} + ... + (i_{S_{M-1}} - 1) I_{S_M} + i_{S_M}$, for $m = 0, ..., M-1$. Hence, eq. (4.11) can be rewritten as the following $(N_{m+1}+1)^{th}$-order PARAFAC model:

$$w^{(m)}_{i_{S_m+1},...,i_{S_{m+1}},j_{m+1}} = \sum_{q_{m+1}=1}^{Q_{m+1}} a^{(S_m+1)}_{i_{S_m+1},q_{m+1}} \dots a^{(S_{m+1})}_{i_{S_{m+1}},q_{m+1}} w^{(m+1)}_{j_{m+1},q_{m+1}} \tag{4.11}$$

The recursive formulation (4.11) shows that the original tensor $\mathscr{Y}$ can be viewed as the PARAFAC nesting of the $M+1$ tensors $\mathscr{W}^{(m)}$, for $m = 0, ..., M$, with $\mathscr{W}^{(0)} = \mathscr{Y}$ and $\mathscr{W}^{(M)} = \mathscr{B}$.

The HONPD can also be expressed using a tensor notation, similarly as the EONPD, as follows:

$$\mathscr{Y} = \mathscr{B} \underset{q_1}{*} \mathscr{R}^{(1)} \underset{q_2}{*} \mathscr{R}^{(2)} \dots \underset{q_M}{*} \mathscr{R}^{(M)}, \tag{4.12}$$

where $\mathscr{R}^{(m)} \in \mathbb{C}^{I_{S_{m-1}+1} \times I_{S_m} \times Q_m}$, for $m = 1, ..., M$, is defined as

$$r^{(m)}_{i_{S_{m-1}+1}, ..., i_{S_m}, q_m} = a^{(S_{m-1}+1)}_{i_{S_{m-1}+1}, q_m} \dots a^{(S_m)}_{i_{S_m}, q_m}. \tag{4.13}$$

That shows that the core tensor $\mathscr{B}$ interacts with all the PARAFAC tensors $\mathscr{R}^{(1)} \dots \mathscr{R}^{(M)}$ by means of the contraction operation.

4.3 Low-Correlation Multilinear Dimensionality Reduction (LC-MDR)

In many pattern recognizing and signal processing applications, it is common to work with a large numbers of variables. However, high-dimensional data have two main drawbacks: a high computational complexity and the so-called curse of dimensionality (BELARBI *et al.*, 2017; LU *et al.*, 2008; YAN *et al.*, 2006). Dimensionality reduction techniques can be applied to solve this issue by decreasing the data dimensionality. Indeed, if the entries of input samples are correlated, they may be confined into a subspace, where a low-dimension representation can be possible.

For vector data, the PCA is the most commonly used dimensionality reduction technique. For the case of tensor patterns, the MPCA is the most popular alternative. However, as earlier mentioned, the MPCA does note generate uncorrelated variables. Having uncorrelated data may be beneficial in several kinds of problems, as, for instance, in the machine learning tasks (LU *et al.*, 2008; YAN *et al.*, 2006).

In this section, a proposed technique for dimensionality reduction in tensor patterns, denoted by LC-MDR, is presented. The method tries to minimize the correlation of the output tensor by fitting the input correlation tensor into an EONPD, as detailed in the sequel. The LC-MDR generates output tensor samples with low correlation and low dimensions, which can be used for dimensionality reduction and feature extraction in tensor patterns. In the sequel, the modeling of the input correlation tensor as a EONPD is presented. After, the algorithm of the LC-MDR is detailed.

4.3.1 EONPD Modeling of the Input Correlation Tensor

Let $\mathscr{X} \in \mathbb{C}^{I_1 \times \ldots \times I_N}$ be a N-th order tensor random variable. The problem considered in the present work is to perform a multilinear transformation in $\mathscr{X}$ such that the output (transformed) tensor has smaller dimensions than $\mathscr{X}$, with most of the information being preserved in the output data. However, contrarily to the MPCA, which uses orthogonal transformation matrices, this orthogonality constraint is not used by the LC-MDR, in order to minimize the output correlation.

The N-th order output tensor $\mathscr{Y} \in \mathbb{C}^{R_1 \times \ldots \times R_N}$ is obtained by means of the following multilinear transformation:

$$\mathscr{Y} = \mathscr{X} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \ldots \times_N \mathbf{A}^{(N)}, \tag{4.14}$$

where $\mathbf{A}^{(n)} \in \mathbb{C}^{R_n \times I_n}$, for $n = 1, ..., N$, is a transformation matrix, with $R_n \leq I_n$. Eq. (4.14) can be written in scalar form as follows:

$$y_{r_1,\ldots,r_N} = \sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x_{i_1,\ldots,i_N} a^{(1)}_{r_1,i_1} \ldots a^{(N)}_{r_N,i_N}. \tag{4.15}$$

Assuming that the transformation matrices $\mathbf{A}^{(n)}$ are full row rank, the tensor $\mathscr{X}$ can be obtained from $\mathscr{Y}$ in the following way:

$$\mathscr{X} = \mathscr{Y} \times_1 \mathbf{A}^{(1)^+} \ldots \times_N \mathbf{A}^{(N)^+}, \tag{4.16}$$

or, in scalar form

$$x_{i_1,\ldots,i_N} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} y_{r_1,\ldots,r_N} a^{(1)^+}_{i_1,r_1} \ldots a^{(N)^+}_{i_N,r_N}, \tag{4.17}$$

where $a^{(n)^+}_{i_N,r_n}$ is a typical element of $\mathbf{A}^{(n)^+}$, for $n = 1, ..., N$.

Let us form a $(2N)^{th}$-order tensor, denoted by $\mathscr{R}^{(x)} \in \mathbb{C}^{I_1 \times I_1 \times I_2 \times I_2 \ldots I_N \times I_N}$, with the all correlations of the input data, given by:

$$r^{(x)}_{i_1,i'_1,i_2,i'_2,\ldots,i_N,i'_N} = E[x_{i_1,i_2,\ldots,i_N} x^*_{i'_1,i'_2,\ldots,i'_N}]. \tag{4.18}$$

Without loss of generality, it is considered that the input data $x_{r_1,\ldots,r_N}$ have zero average. If that is not the case, their averages must be subtracted. Substituting (4.17) into (4.18) leads to:

$$r^{(x)}_{i_1,i'_1,\ldots,i_N,i'_N} = \sum_{r_1=1}^{R_1} \sum_{r'_1=1}^{R'_1} \cdots \sum_{r_N=1}^{R_N} \sum_{r'_N=1}^{R'_N} r^{(y)}_{r_1,r'_1,\ldots,r_N,r'_N} a^{(1)^+}_{i_1,r_1} a^{(1)^{+*}}_{i'_1,r'_1} \ldots a^{(N)^+}_{i_N,r_N} a^{(N)^{+*}}_{i'_N,r'_N}, \tag{4.19}$$

where $r^{(y)}_{r_1,r'_1,r_2,r'_2,\ldots,r_N,r'_N} = E[y_{r_1,r_2,\ldots,r_N} y^*_{r'_1,r'_2,\ldots,r'_N}]$. Note that the output correlations may also form a $(2N)^{th}$-order tensor, denoted by $\mathscr{R}^{(y)} \in \mathbb{C}^{R_1 \times R_1 \times R_2 \times R_2 \ldots R_N \times R_N}$.

The objective of the proposed technique is to minimize the correlation between the elements of the output tensor $\mathscr{Y}$. In the ideal case, the output data is fully uncorrelated, as follows:

$$r^{(y)}_{r_1,r'_1,\ldots,r_N,r'_N} = E[y_{r_1,\ldots,r_N} y^*_{r'_1,\ldots,r'_N}] = \begin{cases} \sigma^2_{y_{r_1,\ldots,r_N}}, & \text{if } r_1 = r'_1, \ldots, r_N = r'_N, \\ 0, & \text{otherwise} \end{cases} \tag{4.20}$$

where $\sigma^2_{y_{r_1,\ldots,r_N}} \neq 0$ is the variance of $y_{r_1,\ldots,r_N}$. Substituting (4.20) into (4.19), we get:

$$r^{(x)}_{i_1,i'_1,\ldots,i_N,i'_N} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} \tilde{r}^{(y)}_{r_1,\ldots,r_N} a^{(1)+}_{i_1,r_1} a^{(1)+*}_{i'_1,r_1} \ldots a^{(N)+}_{i_N,r_N} a^{(N)+*}_{i'_N,r_N}, \tag{4.21}$$

where $\tilde{r}^{(y)}_{r_1,\ldots,r_N} = r^{(y)}_{r_1,r_1,\ldots,r_N,r_N}$. The variances $\tilde{r}^{(y)}_{r_1,\ldots,r_N}$ form a N^{th}-order tensor $\tilde{\mathscr{R}}^{(y)} \in \mathbb{C}^{R_1 \times R_2 \times \ldots \times R_N}$.

Note that (4.21) corresponds to the EONPD in (4.1), with the following correspondences:

$$\left(\mathscr{R}^{(x)}, \tilde{\mathscr{R}}^{(y)}, \mathbf{A}^{(n)+}\right) \Leftrightarrow \left(\mathscr{Y}, \mathscr{B}, \mathbf{A}^{(n)}\right), \tag{4.22}$$

$$(2N, I_1, I_1, \ldots, I_N, I_N, R_1, \ldots, R_N) \Leftrightarrow \left(N, I_1, I_2, \ldots, I_{N-1}, I_N, Q_1, \ldots, Q_{N/2}\right). \tag{4.23}$$

In other words, in (4.21), the core tensor is given by $\tilde{\mathscr{R}}^{(y)}$, the factor matrices are $\mathbf{A}^{(n)+}$ and the ranks are $R_1, \ldots R_N$. Moreover, the constraint that the order of the EONPD must be even is naturally verified by the tensor $\mathscr{R}^{(x)}$.

The successive nests in $\mathscr{R}^{(x)}$ can be viewed explicitly by rewriting (4.21) recursively, as follows:

$$\begin{aligned} u^{(n)}_{r_1,r_2,\ldots,r_{n+1},i_{n+1},i'_{n+1},\ldots,i_N,i'_N} = \sum_{r_{n+1}=1}^{R_{n+1}} & a^{(n+1)+}_{i_{n+1},r_{n+1}} a^{(n+1)+*}_{i'_{n+1},r_{n+1}} \times \\ & \times u^{(n+1)}_{r_1,r_2,\ldots,r_{n+1},i_{n+2},i'_{n+2},\ldots,i_N,i'_N}, \end{aligned} \tag{4.24}$$

for $n = 0, \ldots, N$, where

$$\begin{aligned} u^{(n)}_{r_1,r_2,\ldots,r_n,i_{n+1},i'_{n+1},\ldots,i_N,i'_N} = \sum_{r_{n+1}=1}^{R_{n+1}} \cdots \sum_{r_N=1}^{R_N} & \tilde{r}^{(y)}_{r_1,r_2,\ldots,r_N} \times \\ & \times a^{(n+2)+}_{i_{n+1},r_{n+1}} a^{(n+1)+*}_{i'_{n+1},r_{n+1}} \ldots a^{(N)+}_{i_N,r_N} a^{(N)+*}_{i'_N,r_N}. \end{aligned} \tag{4.25}$$

with $u^{(0)}_{i_1,i'_1,\ldots,i_N,i'_N} = r^{(x)}_{i_1,i'_1,\ldots,i_N,i'_N}$ and $u^{(N)}_{r_1,\ldots,r_N} = \tilde{r}^{(y)}_{r_1,\ldots,r_N}$.

Similarly as in Subsection 4.3.1, by concatenating the indices r_n, ..., r_1, i_{n+2}, i'_{n+2}, ..., i_N, i'_N into one index j_{n+1}, with $1 \leq j_{n+1} \leq J_{n+1}$ and $J_{n+1} = R_n \ldots R_1 I^2_{n+2} \ldots I^2_N$, eq. (4.24) can be expressed as the following third-order PARAFAC model:

$$u^{(n)}_{i_{n+1},i'_{n+1},j_{n+1}} = \sum_{r_{n+1}=1}^{R_{n+1}} a^{(n+1)^+}_{i_{n+1},r_{n+1}} a^{(n+1)^{+*}}_{i'_{n+1},r_{n+1}} u^{(n+1)}_{j_{n+1},r_{n+1}}. \tag{4.26}$$

The above presented modeling of the input correlation tensor $\mathscr{R}^{(x)}$ shows that, if the output signals are uncorrelated, the tensor $\mathscr{R}^{(x)}$ must follow an EONPD model. This result is exploited in the next subsection to derive the estimation algorithm of the LC-MDR technique.

Note that, when $N = 2$, i.e. if the inputs are matrices, eq. (4.21) becomes:

$$r^{(x)}_{i_1,i'_1,i_2,i'_2} = \sum_{r_1=1}^{R_1} \sum_{r_2=2}^{R_2} \tilde{r}^{(y)}_{r_1,r_2} a^{(1)^+}_{i_1,r_1} a^{(1)^{+*}}_{i'_1,r'_1} a^{(2)^+}_{i_2,r_2} a^{(2)^{+*}}_{i'_2,r'_2}, \tag{4.27}$$

which corresponds to the standard NPD in (2.20).

4.3.2 Estimation Algorithm

The main idea of the LC-MDR is to estimate the matrices $\mathbf{A}^{(n)^+}$ by fitting the tensor $\mathscr{R}^{(x)}$ into a $(2N)^{th}$-order EONPD model. The transformation matrix $\mathbf{A}^{(n)}$, for $n = 1,...,N$, is then estimated by calculating the pseudoinverse of $\mathbf{A}^{(n)^+}$. In other words, the transformation matrices are estimated so that the output correlations approach the ideal case described by (4.20), where the output data are uncorrelated.

The LC-MDR estimates successively the factors of the PARAFAC decompositions by means of the ALS algorithm (COMON *et al.*, 2009). The ALS is an iterative algorithm with multiple alternated steps that is widely used for estimating the factors of a PARAFAC tensor. At each step of each iteration, the ALS algorithm estimates one of the factor matrices using the LS method by assuming that the other factor matrices are known, using the previous estimates of these factor matrices.

The cost function of the LC-MDR is an LS-type function that fits the tensor $\mathscr{R}^{(x)}$ into an EONPD model, using unfolded matrix expressions. Eq. (4.26) can be rewritten by means of unfolded matrices, as follows:

$$\begin{aligned} \mathbf{U}^{(n)}_{[n+1]} &= \left(\mathbf{A}^{(n+1)^{+*}} \odot \mathbf{U}^{(n+1)}_{[n+1]}\right) \mathbf{A}^{(n+1)^{+T}}, \\ \mathbf{U}^{(n)}_{[n+1,n+2]} &= \left(\mathbf{A}^{(n+1)^+} \odot \mathbf{A}^{(n+1)^{+*}}\right) \mathbf{U}^{(n+1)^T}_{[n+1]}, \end{aligned} \tag{4.28}$$

where $\mathbf{U}_{[n+1]}^{(n+1)} \in \mathbb{C}^{J_{n+1} \times R_{n+1}}$ is the mode-$(n+1)$ unfolded matrix of $\mathscr{U}^{(n+1)}$, $\mathbf{U}_{[n+1]}^{(n)} \in \mathbb{C}^{I_{n+1}J_{n+1} \times I_{n+1}}$ is the mode-$(n+1)$ unfolded matrix of $\mathscr{U}^{(n)}$ and $\mathbf{U}_{[n+1,n+2]}^{(n)} \in \mathbb{C}^{I_{n+1}^2 \times J_{n+1}}$ is a unfolded matrix of $\mathscr{U}^{(n)}$ the combines the modes $(n+1)$ and $(n+2)$ in the first dimension, with $\mathscr{U}^{(0)} = \mathscr{R}^{(x)}$ and $\mathscr{U}^{(N)} = \mathscr{R}^{(y)}$. Note that $\mathbf{U}_{[n+1,n+2]}^{(n)}$ is a unfolded matrix that is not in the standard format defined in Chapter 2.

It is important to note that, for the identifiability of the LC-MDR, both products $\left(\mathbf{A}^{(n+1)^{+*}} \odot \mathbf{U}_{[n+1]}^{(n+1)}\right)$ and $\left(\mathbf{A}^{(n+1)^{+}} \odot \mathbf{A}^{(n+1)^{+*}}\right)$ need to be full column rank. As such, we need that $I_{n+1} \geq R_{n+1}$ and $J_{n+1} \geq R_{n+1}$.

At the n^{th} stage, the cost function of the LC-MDR is given by the following LS function:

$$\begin{aligned} J\left(\mathbf{C}^{(n+1)}, \mathbf{U}_{[n+1]}^{(n+1)}\right) &= \left\| \mathbf{U}_{[n+1]}^{(n)} - \left(\mathbf{C}^{(n+1)^*} \odot \mathbf{U}_{[n+1]}^{(n+1)}\right) \mathbf{C}^{(n+1)^T} \right\|^2 \\ &= \left\| \mathbf{U}_{[n+1,n+2]}^{(n)} - \left(\mathbf{C}^{(n+1)} \odot \mathbf{C}^{(n+1)^*}\right) \mathbf{U}_{[n+1]}^{(n+1)^T} \right\|^2, \end{aligned} \tag{4.29}$$

where, for simplifying the notation, we defined $\mathbf{C}^{(n+1)} = \mathbf{A}^{(n+1)^{+}} \in \mathbb{C}^{I_{n+1} \times R_{n+1}}$.

The proposed LC-MDR is presented in Algorithm 1, being composed of N stages, each stage having two iterative steps. At the i^{th} iteration of the n^{th} stage, there are two steps for estimating, in an alternated way, the matrices $\mathbf{C}^{(n+1)}$ and $\mathbf{U}_{[n+1]}^{(n+1)}$ from the estimation of the tensor $\mathscr{U}^{(n)}$ obtained at the end of previous stage, using the LS method based on (4.28). The estimates of $\mathbf{C}^{(n+1)}$ and $\mathbf{U}_{[n+1]}^{(n+1)}$ at the i^{th} iteration are denoted respectively by $\hat{\mathbf{C}}_i^{(n+1)}$ and $\hat{\mathbf{U}}_{[n+1],i}^{(n+1)}$. There is no step for estimating $\hat{\mathbf{C}}_i^{(n+1)^*}$, as it is obtained simply as the conjugate of $\hat{\mathbf{C}}_i^{(n+1)}$.

In Algorithm 1, $\Lambda_{i,n} \in \mathbb{C}^{R_{n+1} \times R_{n+1}}$ corresponds to a diagonal matrix with the norms of the columns of $\hat{\mathbf{C}}_i^{(n+1)}$ in its diagonal elements. At the end of each iteration, the columns of $\hat{\mathbf{C}}_i^{(n+1)}$ are normalized and, in order to maintain the consistency of the model, the columns of $\hat{\mathbf{U}}_{[n+1],i}^{(n+1)}$ are multiplied by squared norms of the columns of $\hat{\mathbf{C}}_i^{(n)}$. Indeed, in (4.28), $\mathbf{U}_{[n+1]}^{(n)}$ and $\mathbf{U}_{[n+1,n+2]}^{(n)}$ remain unchanged if $\hat{\mathbf{C}}_i^{(n+1)}$ and $\hat{\mathbf{C}}_i^{(n+1)^*}$ are post multiplied by $\Lambda_{i,n}^{-1}$, and $\hat{\mathbf{U}}_{[n+1],i}^{(n+1)}$ is post multiplied by $\Lambda_{i,n}^2$. Although these normalization steps are not mandatory, some preliminary simulations showed that they improve the performance of the LC-MDR in the classification of seismic events, as considered in Section 4.5.

Each ALS stage stops when $|e(i) - e(i-1)| < \varepsilon_1$ and $|e(i)| < \varepsilon_2$, where ε_1 and ε_2 are small scalar constants and $e(i)$ is the normalized squared error of the reconstructed tensor, at

Algorithm 1: LC-MDR estimation

Input: $\hat{\mathscr{U}}^{(0)} = \mathscr{R}^{(x)}$
Outputs: $\hat{\mathbf{A}}_i^{(1)}, ..., \hat{\mathbf{A}}_i^{(N)}$
Initialize $\hat{\mathbf{C}}_0^{(1)}, ..., \hat{\mathbf{C}}_0^{(N)}$ and $\hat{\mathbf{U}}_0^{(1)}, ..., \hat{\mathbf{U}}_0^{(N)}$ randomly
for $n = 0 : N$ **do**
  $\beta = 0$
  $i = 0$
  while $\beta = 0$ **do**
    $i = i + 1;$
    $\hat{\mathbf{C}}_i^{(n+1)} = \left[\left(\hat{\mathbf{C}}_{i-1}^{(n+1)^*} \odot \hat{\mathbf{U}}_{[n+1],i-1}^{(n+1)} \right) \hat{\mathbf{U}}_{[n+1]}^{(n)} \right]^T$
    $\hat{\mathbf{U}}_{[n+1],i}^{(n+1)} = \left[\left(\hat{\mathbf{C}}_i^{(n+1)} \odot \hat{\mathbf{C}}_i^{(n+1)^*} \right) \hat{\mathbf{U}}_{[n+1,n+2]}^{(n)} \right]^T$
    $\hat{\mathbf{C}}_i^{(n+1)} \leftarrow \hat{\mathbf{C}}_i^{(n+1)} \Lambda_{i,n}^{-1}$
    $\hat{\mathbf{U}}_{[n+1],i}^{(n+1)} \leftarrow \hat{\mathbf{U}}_{[n+1],i}^{(n+1)} \Lambda_{i,n}^{2}$
    if $|e(i) - e(i-1)| < \varepsilon_1 \quad and \quad |e(i)| < \varepsilon_2$ **then**
      $\beta = 1$
    end
  end
  $\hat{\mathbf{A}}_i^{(n)} = \hat{\mathbf{C}}_i^{(n)^+}$
end

the i^{th} iteration, given by:

$$e(i) = \frac{\left\| \hat{\mathbf{U}}_{[n+1,n+2]}^{(n)} - \left(\hat{\mathbf{C}}_i^{(n+1)} \odot \hat{\mathbf{C}}_i^{(n+1)^*} \right) \hat{\mathbf{U}}_{[n+1],i}^{(n+1)^T} \right\|_F^2}{\left\| \hat{\mathbf{U}}_{[n+1,n+2]}^{(n)} \right\|_F^2}, \tag{4.30}$$

where $\|.\|_F$ denotes the Frobenius norm. The constraint $|e(i) - e(i-1)| < \varepsilon_1$ tests the convergence of the algorithm, while $|e(i)| < \varepsilon_2$ assures that it have converged to an acceptable point. The latter constraint assures that no relevant information is lost by the multilinear transformation, ensuring that the tensor may be reconstructed.

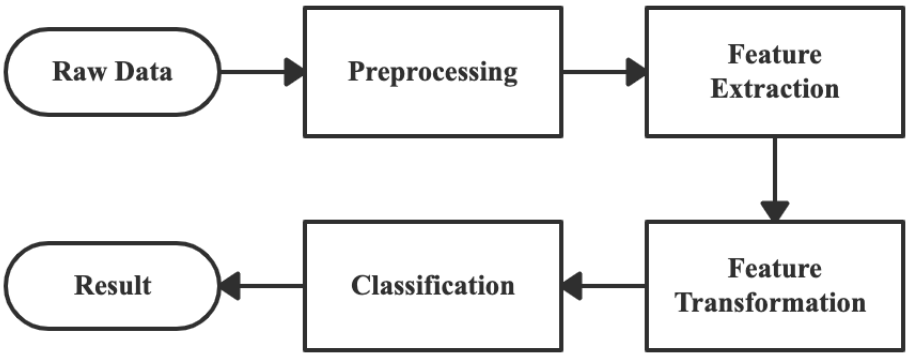
The fact that the columns of the factor matrices $\hat{\mathbf{A}}^{(n)}$ are not orthogonal could generate redundancy in the transformed data. However, this problem is avoided due the criterion of the LC-MDR that decreases the correlation of the output data, making it unnecessary for the factor matrices to be orthogonal. Note that the LC-MDR may also be applied to the case where the inputs are matrices, simply by using $N = 2$.

It's also worth mentioning that the LC-MDR does not impose uniqueness constraints on the EONPD, as ambiguity of the factor matrices does not need to be removed, since the LC-MDR manages to decrease the correlation of the data and reduce its dimensionality nev-

ertheless. The PARAFAC decompositions computed by the LC-MDR at each stage may be essentially unique if some constraints are verified (BERGE; SIDIROPOULOS, 2002; STEGEMAN; SIDIROPOULOS, 2007; SIDIROPOULOS; BRO, 2000), but the global EONPD is not unique. Imposing some constraints on the EONPD could lead to a unique decomposition, however, it is uncertain if this uniqueness would lead to a more efficient dimensionality and correlation reduction. This topics falls outside the scope of this paper.

4.4 Classification system

Figure 20 – Steps of the classification system in which the LC-MDR is tested.



Source: Author.

The LC-MDR technique proposed in this thesis is applied in the classification of the volcano-seismic events presented in Chapter 3. The technique is used before the classifier itself, in order to reduce the number of attributes and decrease the correlation between them. The classification system follows the tensor-based learning framework for multichannel volcano-seismic classification presented in (PEIXOTO *et al.*, 2021). The steps of the classification system are shown in Fig. 20, where the LC-MDR method corresponds to the block “Feature Transformation”.

The first step of the volcano-seismic classification system is a series of preprocessing operations, starting with the subtraction of the time-average mean of the signals, followed by an instrumental correction that is carried out by computing the deconvolution associated with the transfer function of the sensors, expressing the seismic signals in their original unity, similarly as in (LARA *et al.*, 2020).

In the second step, the feature extraction is performed, with the attributes being calculated from four different domains: (i) directly from the prepossessed signals (time domain);

(ii) from the Hilbert transform of the prepossessed signals; (iii) from the estimated PSD; (iv) from the MDFT (TOLIMIERI *et al.*, 2012).

The following 14 attributes are calculated for each of these four domains: 10 windowed averages, total average, kurtosis, skewness and standard deviation. The PSD is calculated using the Welch's method with overlapping of 75%. The windowed average is obtained by dividing the time and frequency series into 10 non-overlapping windows and calculating the average of each window, leading to 10 attributes. That leads to a total of 56 features per signal. These attributes are the same used in (PEIXOTO *et al.*, 2021), where they have proved to be very efficient for classifying volcano-seismic events.

After the feature extraction step, feature transformation is applied on the data, by means of the LC-MDR. Comparing Figure 20 with Figure 14, the framework used in Chapter 3 is very similar to the one in which the LC-MDR is employed. In fact, for comparison purposes, the MPCA (tensor-based) and PCA (vector-based) are also tested, as well as the case where no technique is used for feature transformation, similarly as in Chapter 3.

For instance, by using the LC-MDR technique, the correlation of the output tensor is minimized by fitting the input correlation tensor into an EONPD, as detailed earlier, providing data easier to classify. Also, the LC-MDR generates output tensor samples with low correlation and low dimensions, which can be used for dimensionality reduction of the input data. Additionally, dimensionality reduction and correlation reduction can also be achieved using the MPCA or the PCA techniques. However, contrarily to the standard PCA that generates fully uncorrelated data, the MPCA is not able to create perfectly uncorrelated variables. The performance of the used feature transformation techniques is discussed later.

In the sequel, the multidimensional data is used to feed the tensor-based classifier. In this chapter, the following classifiers were selected: SVM, R1-SPM, SPM and STuM. Table 12 resumes the classification system adopted techniques and classifiers.

Table 12 – Classification framework description.

Preprocessed Data	856 third-order $2 \times 3 \times 3000$ tensors
Feature Extraction	MFT, PSD, Hilbert
Feature Transformation	LC-MDR (2D), LC-MDR (3D), MPCA, PCA
Classification	SVM, R1-SPM, SPM, STuM

Source: Author.

4.5 Results and Discussion

In this section, results that evaluate the proposed LC-MDR as a tool used in statistical classification are presented and discussed. In particular, the LC-MDR is used for reduction of feature dimensionality and correlation, using the database and framework described in Section 4.4. Although the results presented in Chapter 3 focused in highlighting the classification accuracy of the framework techniques (MPCA, SPM, STuM, etc), in this chapter the goal is to highlight the performance of the LC-MDR in reducing correlation, dimensionality and also improving classification.

As earlier mentioned, the tensor-based classifiers R1-SPM, SPM and STuM are used for performing the classification, with K-fold cross-validation using $K = 10$. The conventional (vector-based) SVM classifier is also tested for comparison purposes, as well as the dimensionality reduction techniques MPCA (tensor-based) and PCA (vector-based).

Several figures of merit are used to evaluate the impact of the dimensionality reduction techniques on the the classification system. The accuracy and confusion table are used to measure the number of errors of the classifier, where the accuracy is defined as the number of correctly classified samples over the total number of samples. The correlation of the output data is also used as a figure of merit for the techniques.

Moreover, to measure computer performance, the floating point operations per second (FLOPS) and the average execution time, of the feature technique plus classifier, are considered. The processing times were obtained with an 9^{th} generation Intel Core i3 processor, running MATLAB version 2017b. The SVM functions used to model the classifiers are *fitcsvm* and *predict*, respectively.

Some preliminary tests, whose results are omitted, were carried out in order to find the best hyperparameters of the classifiers. The best results were obtained with the relaxing constant $C = 100$ and the ranks of the STuM $Q_1 = 2$, $Q_2 = 3$ and $Q_3 = 5$. When not stated otherwise, these values are used. The parameters of ε_1 and ε_2 of the EONPD are set to 10^{-6} and 10^{-3} respectively.

Regarding the dimensionality of the input data, two cases are considered: bidimensional (2D) inputs ($N = 2$) and tridimensional (3D) inputs ($N = 3$). In the 3D case, the data fed into the dimensionality reduction techniques are $2 \times 3 \times 56$ tensors. In the 2D case, the first two modes are concatenated, and the resulting data are 6×56 matrices. The vector-based techniques (PCA and SVM) perform the vectorization of the tensor samples, yielding pattern vectors of

Table 13 – TCR with full projection.

Feature Transformation	Dimensions	TCR
LC-MDR (2D)	6×56	0.7121
MPCA	$2 \times 3 \times 56$	0.8001
LC-MDR (3D)	$2 \times 3 \times 56$	**0.7042**
None	$2 \times 3 \times 56$	0.8569

Source: Author.

dimension 336. Moreover, when the PCA is used in conjunction with the tensor-based STuM, the data is re-tensorized before entering the classifier.

4.5.1 Correlation Reduction with Full Projection

In this subsection, results with full projection of the data, i.e. with no dimensionality reduction, are presented and discussed. The objective of these results is to evaluate the impact of the correlation reduction on the accuracy of the classifiers.

To measure the correlation in the transformed data tensor $\mathscr{Y}$ (the input to the classifier), the following tensor correlation ratio (TCR) is used:

$$r_{\mathscr{Y}} = \frac{r_C^{(y)}}{r_C^{(y)} + r_V^{(y)}}, \tag{4.31}$$

where $r_V^{(y)}$ stands for the sum of the elements of $\mathscr{R}^{(y)}$ that corresponds to variances, i.e. when the indices verify $r_1 = r_1', \ldots, r_N = r_N'$, that is:

$$\begin{aligned} r_V^{(y)} &= \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} r^{(y)}_{r_1,r_1,\ldots,r_N,r_N} \\ &= \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} \sigma^2_{y_{r_1,\ldots,r_N}}. \end{aligned} \tag{4.32}$$

Moreover, $r_C^{(y)}$ stands for the sum of the absolute values of the elements of $\mathscr{R}^{(y)}$ that corresponds to correlations, i.e. when $r_1 = r_1', \ldots, r_N = r_N'$ is not verified, that is: $r_C^{(y)} = r_T^{(y)} - r_V^{(y)}$, where $r_T^{(y)}$ is the sum of the absolute values of all the elements of $\mathscr{R}^{(y)}$. The TCR is used in the sequel to measure the decorrelation of the data after being processed by the MPCA and LC-MDR techniques.

Table 13 shows TCR, defined in (4.31), obtained by the MPCA and LC-MDR methods, including the bidimensional LC-MDR, with full projection. The TCR of original data, i.e. without any correlation reduction technique being applied, is also shown for comparison purposes. It can be seen from Table 13 that the LC-MDR has significantly reduced the TCR, in

Table 14 – Accuracy obtained by different techniques with full projection.

Feature Transformation (Dimensions)	**Classifier**	**Accuracy**	**Execution time (s)**
PCA (336)	SVM	79.4%	2663.8
	R1-SPM	80.5%	2911.3
	SPM	81.2%	3133.2
	STuM	81.7%	3517.4
LC-MDR (2D) (6×56)	SVM	80.8%	**2245.2**
	R1-SPM	82.6%	2664.1
	SPM	83.9%	3201.4
	STuM	85.1%	3721.7
MPCA ($2 \times 3 \times 56$)	SVM	81.6%	8335.4
	R1-SPM	82.2%	11454.3
	SPM	82.6%	1292.2
	STuM	84.3%	13239.0
LC-MDR (3D) ($2 \times 3 \times 56$)	SVM	80.2%	2723.7
	R1-SPM	84.9%	3154.1
	SPM	85.8%	3877.9
	STuM	**86.2**%	4779.4
None ($2 \times 3 \times 56$)	SVM	75.2%	2117.5
	R1-SPM	76.8%	2455.6
	SPM	77.2%	2598.0
	STuM	78.8%	3002.1

Source: Author.

comparison to the original data, even when dimensionality reduction is not performed. Besides, the reduction of correlation carried out by the LC-MDR, for both the 2D and 3D versions, is much more significant than the one provided by the MPCA. This is due to the fact that the LC-MDR is explicitly designed for reducing the output correlation, contrarily to the MCPA. In other words, the LC-MDR reduces the TCR even if the original dimensions are maintained.

Regarding the classification of the volcano-seismic events, Table 14 presents the accuracy provided using the the PCA, the MPCA and the LC-MDR methods with full projection of the data, along with both the SVM and the STM-based classifiers. Table 14 also shows the accuracy obtained in the case where no technique is used for feature transformation, i.e. the preprocessed data is feed to the classifiers. The execution times (feature transformation plus classification) are also shown in this table.

As it can be viewed in Table 14, the best accuracy is obtained by the LC-MDR (3D) with the STuM, achieving an accuracy of 86.2% (this accuracy is highlighted in bold). The second best accuracy was obtained by the LC-MDR (2D) with STuM, showing that the stronger reduction of correlation achieved by the proposed technique facilitates the classification. Moreover, the third better accuracy was obtained by the MPCA with STuM, which also uses 3D data, showing that the preservation of the tensor structure of data is beneficial for the classification system.

In addition, the accuracy provided by the STuM are higher than the ones of the other tensor-based classifiers and the SVM, for all the tested methods. Also, all tensor-based classifiers provided higher accuracy in comparison to the SVM, corroborating with the fact that breaking the tensor structure of the data induces a worse classification. Moreover, the SPM always showed better success rates when comparing to the simpler R1-SPM, with $Q = 1$. As can be seen, the rank value impacts on the classification, which corroborates the results of Figure 18 of Chapter 3.

Using no technique at all delivered the worst accuracy of the table, showing the efficacy of the MPCA and LC-MDR with full projection of the data, although the vector-based PCA provides a small gain in CCR with respect to the case with no PCA. The case with no feature transformation showed the lowest accuracy, behind the vector-based PCA.

It can also be viewed in Table 14 that the execution time of the STM-based classifiers is higher than the one of the SVM for all the tested cases. This is due to the fact that the STMs executes several standard vector-based SVMs (with smaller input dimensions) for each input sample. Regarding the execution time of the techniques for dimensionality and correlation reduction, the smaller execution time were provided by case with no technique, as expected.

The LC-MDR (2D) has the lowest execution time among the transformation techniques, when used with the SVM. The bidimensional LC-MDR is modeled as a standard NPD (4.27), requiring only one nesting, whereas the tridimensional case, which is modeled by the EONPD, requires two nestings, thus demanding more running time. Moreover, the proposed LC-MDR was executed much faster than the MPCA, which is by far the most time-expending algorithm. These results show that the LC-MDR is much faster than the other tensor-based technique (MPCA) and it has an execution time comparable with the vector-based method (PCA). Comparing the LC-MDR with no use of transformation on the data, it is clear that the use of the LC-MDR technique is a trade-off between execution time and accuracy.

Table 15 – TCR for the MPCA and the LC-MDR for several array dimensions.

Technique	**Dimensions**	**TCR** ($r_{\mathscr{Y}}$)
LC-MDR (2D)	6×15	0.3923
	4×15	0.3415
	6×10	0.2894
	4×10	0.1930
	6×5	0.2007
	4×5	0.1445
MPCA	$2 \times 3 \times 15$	0.6517
	$2 \times 2 \times 15$	0.6153
	$2 \times 3 \times 10$	0.5678
	$2 \times 2 \times 10$	0.4926
	$2 \times 3 \times 5$	0.3020
	$2 \times 2 \times 5$	0.2634
LC-MDR (3D)	$2 \times 3 \times 15$	0.3896
	$2 \times 2 \times 15$	0.3328
	$2 \times 3 \times 10$	0.2851
	$2 \times 2 \times 10$	0.1844
	$2 \times 3 \times 5$	0.2157
	$2 \times 2 \times 5$	**0.1394**
None	$2 \times 3 \times 56$	0.8569

Source: Author.

4.5.2 Correlation and dimensionality reduction

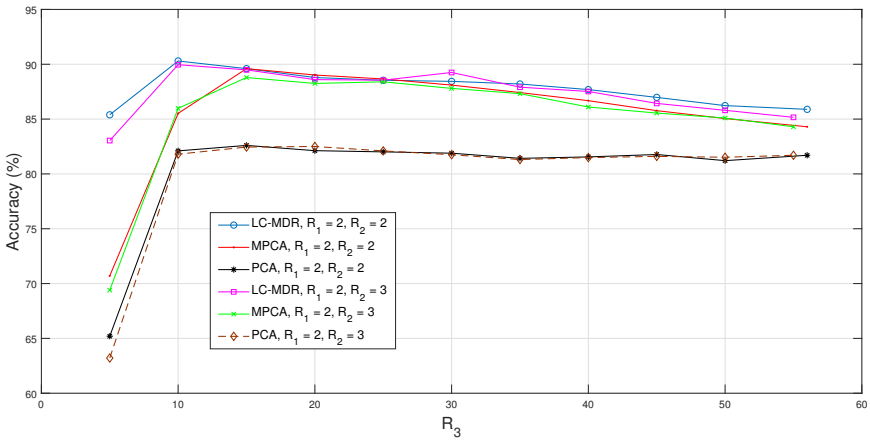
In this subsection, results with dimensionality reduction are presented and discussed. The objective is to assess the impact of a simultaneous reduction on the correlation and dimensionality on the classification.

Table 15 shows the TCR, defined by (4.31), obtained by the MPCA and LC-MDR methods, for several configurations of the dimensions of $\mathscr{Y}$. Table 15 also shows the TCR in the case where no technique is applied. It can be viewed from this table that, for all the tested array dimensions, the LC-MDR provided much smaller correlations than the MPCA, confirming that the proposed technique has a much higher ability to the reduce the correlations than the MPCA technique. As already explained, this is due to the fact that the cost function of the LC-MDR that explicitly takes the output correlation into account. In comparison to the 2D LC-MDR, the 3D LC-MDR showed slightly lower TCRs.

It can also be noted from Table 15 that the smaller values of TCR were obtained when the tensor dimensions are small. In particular, the lowest TCR is obtained when the dimensions of $\mathscr{Y}$ are $2 \times 2 \times 5$. As expected, this results shows that it is easier to remove the correlation when there are less data to process.

For the next result, which show the accuracy, many tests were carried out in order to

Figure 21 – Accuracy obtained by different techniques varying R_3.



Source: Author.

find the ranks R_1, R_2 and R_3 that provide the highest rates. Some of these results are shown in Fig. 21, which presents the classification accuracy versus R_3 (number of attributes), obtained with the techniques PCA, MPCA and the LC-MDR. The methods MPCA and the LC-MDR were used with the STuM classifier while the PCA was used with the SVM. Regarding the values of R_1 and R_2, this figure considers two cases: (i) $R_1 = R_2 = 2$ and (ii) $R_1 = 2, R_2 = 3$. For the PCA, the number of used components is equal to $R = R_1R_2R_3$, with R_1 and R_2 fixed as mentioned.

It can be observed in this figure that the LC-MDR provided the highest accuracy, outclassing the MPCA and PCA for the two configurations of R_1 and R_2. Other simulations similar to the ones of Fig. 21 have been carried out, where the best result was obtained with $R_1 = 2$, $R_2 = 2$ and $R_3 = 10$ for the LC-MDR, with $R_1 = 2$, $R_2 = 2$ and $R_3 = 15$ for the MPCA and $R = 60$ components for the PCA ($R_1 = 2, R_2 = 2$ and $R_3 = 15$). The remaining results of this chapter were obtained using these parameters.

Furthermore, Figure 21 showed a decrease in accuracy of the LC-MDR and MPCA algorithms as the rank R_3 went higher than 30. This can be explained as follows.

Table 16 shows the accuracy obtained by the PCA, MPCA and LC-MDR methods, with dimensionality reduction, using both the SVM and STM-based classifiers. To have a better comparison, the MPCA and 3D LC-MDR were tested with both configurations $R_1 = 2$, $R_2 = 2$, $R_3 = 10$ and $R_1 = 2$, $R_2 = 2$, $R_3 = 15$, which were the best results obtained. The execution times of the techniques are also shown in this table. As in Table 14, the best accuracy (90.3%) was obtained with the tridimensional LC-MDR and the STuM (with dimensions $2 \times 2 \times 10$), which confirms that the the stronger reduction of correlation achieved by the LC-MDR improves the classification.

Table 16 – Accuracy obtained by the dimensionality reduction techniques.

Technique (Dimensions)	**Classifier**	**Accuracy**	**Execution time (s)**
PCA (60)	SVM	82.6%	1132.3
	R1-SPM	82.9%	1883.7
	SPM	83.1%	2011.1
	STuM	83.4%	2233.1
LC-MDR (2D) (4×10)	SVM	83.9%	**601.3**
	R1-SPM	86.2%	899.2
	SPM	87.5%	1004.7
	STuM	88.9%	1235.8
MPCA ($2 \times 2 \times 15$)	SVM	84.0%	3665.4
	R1-SPM	87.2%	3989.8
	SPM	88.1%	4314.9
	STuM	89.6%	4889.6
MPCA ($2 \times 2 \times 10$)	SVM	83.8%	3288.2
	R1-SPM	84.9%	3774.1
	SPM	85.2%	4107.8
	STuM	85.5%	4325.7
LC-MDR (3D) ($2 \times 2 \times 15$)	SVM	84.0%	702.5
	R1-SPM	86.9%	1001.7
	SPM	88.3%	1224.8
	STuM	89.5%	1411.9
LC-MDR (3D) ($2 \times 2 \times 10$)	SVM	84.1%	655.1
	R1-SPM	88.6%	947.6
	SPM	89.3%	1105.5
	STuM	**90.3**%	1371.3

Source: Author

The second best accuracy was obtained by the MPCA with the STuM (with dimensions $2 \times 2 \times 15$). In addition, the accuracy provided by the STuM, SPM and R1-SPM are higher than the ones of the SVM, for all the tested methods, which corroborates the fact that the preservation of the tensor structure of data helps the classification. Using the dimensions $2 \times 2 \times 15$, the difference in accuracy between the MPCA and the 3D LC-MDR is only 0.1, with the LC-MDR running almost 3 times faster. It should also be noted that the matrix-based 2D LC-MDR with dimensions 4×10 provided an accuracy higher than the one of the vector-based PCA and even the MPCA with dimensions $2 \times 2 \times 10$.

Moreover, comparing the results of Tables 14 and 16, it can be viewed that the better

Table 17 – Confusion Matrix provided by the tridimensional LC-MDR with STuM.

		True class					
		EX	HB	LP	TR	VT	Overall
Predicted class	EX	**11**	0	3	2	0	
	HB	0	**10**	0	0	0	
	LP	0	36	**378**	11	5	
	TR	0	0	5	**367**	0	
	VT	5	2	11	3	**7**	
Accuracy(%)		68.7	20.8	95.2	95.6	58.3	**90.3**

Source: Author.

accuracy are obtained when dimensionality reduction is carried out, compared with the full projection case, for the two tested classifiers. The dimensionality reduction also led to faster execution times than the full projection cases, for both the classifiers. However, in Table 16, the smaller execution time was obtained with the proposed LC-MDR, showing that this technique may benefit from a significant gain in execution time when it is implemented with dimensionality reduction.

Still in the execution time metric of Table 16, the tridimensional LC-MDR showed running times very close for the two dimensionality reduction cases ($2 \times 2 \times 15$ and $2 \times 2 \times 10$), which highlights the ability of the LC-MDR to enhance running times in the classification framework while reducing the dimensionality, with small execution time differences regarding the dimension size. Also, the 2D LC-MDR performs faster in comparison to the 3D case, which was explained earlier.

Finally, Table 17 presents the confusion matrix for the classification of the data processed by the proposed technique (LC-MDR) coupled with the STuM classifier, for $R_1 = 2$, $R_2 = 2$ and $R_3 = 10$, which is the highest accuracy result obtained. It can be viewed from this table that the classes TR and LP provided the highest accuracy, while the HB showed the worst accuracy. The HB class, as described in Chapter 3, has characteristics of both the LP and VT classes, which explains the misclassification of its samples to one of these classes.

The above results showcase the technique advantages, in comparison to the MPCA and the PCA, as an approach that achieves higher classification rates and speeds. This is due to the fact that the LC-MDR does not impose orthogonality on the transformation matrices, in order to guarantee the correlation tensor as diagonal, thus, making the transformed data uncorrelated and more likely to be classified.

Table 18 – Time complexity of the LC-MDR, PCA and MPCA for $N = 3$ for dimensionality reduction.

Technique	Time Complexity
PCA	$\mathcal{O}(PI_1^2I_2^2I_3^2 + I_1^3I_2^3I_3^3 + PI_1I_2I_3R_1R_2R_3)$
MPCA	$\mathcal{O}(PI_1I_2I_3[I_1 + I_2 + I_3] + I_1^3 + I_2^3 + I_3^3 + PI_1I_2I_3[R_1 + R_2 + R_3])$
LC-MDR	$\mathcal{O}\left(PN_{it}[I_1^3R_1^2J_1^2 + I_2^3R_2^2J_2^2 + I_3^3R_3^2J_3^2]\right)$

Source: Author.

4.5.3 Computational Cost Analysis

In the sequel, a brief computational complexity analysis of the estimation algorithm of the LC-MDR. The time complexity is considered as a function of the input parameters, using the big-O notation. The time complexity is estimated by counting the number of elementary operations performed by the algorithm, considering the worst-case time complexity, which is the maximum amount of elementary operations required for inputs of a given size. Moreover, the big-O notation corresponds to the asymptotic behavior of the complexity when the input size increases.

The complexity analysis of the used techniques is of great importance because it shows how much computer power it is needed to run the applications and the original proposition (LC-MDR) of this paper.

Table 18 shows the time complexity of the LC-MDR, as well as those of the concurrent PCA and MPCA techniques, for $N = 3$, i.e. for third-order input tensors, where N_{it} the number of iterations. The computational complexity of the LC-MDR depends on the dimensions of the input tensor, the ranks of the PARAFAC decomposition, the number of iterations of each ALS loop and the number of samples.

Although the time-complexity of the tridimensional LC-MDR seems higher than the one of the MPCA, the results of Tables 14 and 16 showed smaller execution times for the LC-MDR in comparison to the MPCA. This is due the fact that the expressions in Table 18 are for the worst case scenario, however, the application of this chapter may not be this case, thus executing faster than the MPCA.

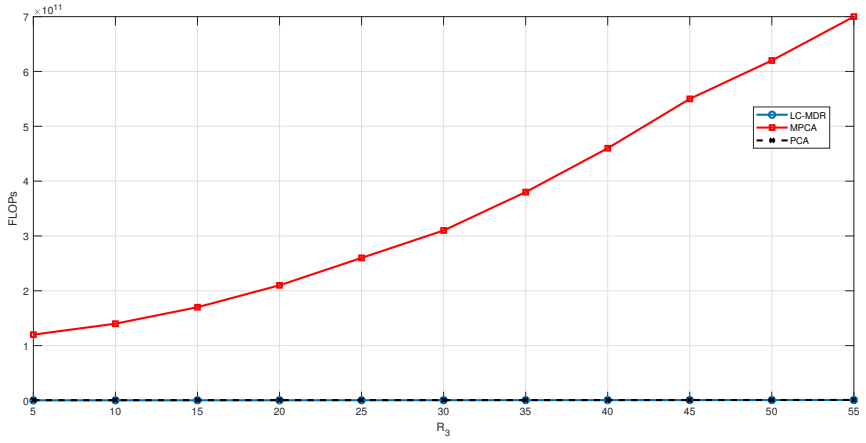
To have a better evaluation of the computational cost of the techniques, Table 19 shows the FLOPS of the techniques with dimensionality reduction and output dimensions R_1 = 2, R_2 = 2 and R_3 = 10 for the MPCA and 3D LC-MDR, $R_1R_2 = 4$, $R_3 = 10$ for the 2D LC-MDR and $R_1R_2R_3 = 40$ for the PCA. From Table 19, it can be viewed that the 2D LC-MDR is the less demanding algorithm in terms of FLOPS, whereas the 3D LC-MDR is the technique that

Table 19 – FLOPS counts for the different techniques with dimensionality reduction.

Technique	**FLOPS**
PCA	5.0×10^8
MPCA	1.4×10^{11}
LC-MDR (2D)	1.9×10^8
LC-MDR (3D)	2.2×10^8

Source: Author.

Figure 22 – FLOPS of the transformation techniques when varying R_3, for $R_1 = R_2 = 2$.



Source: Author.

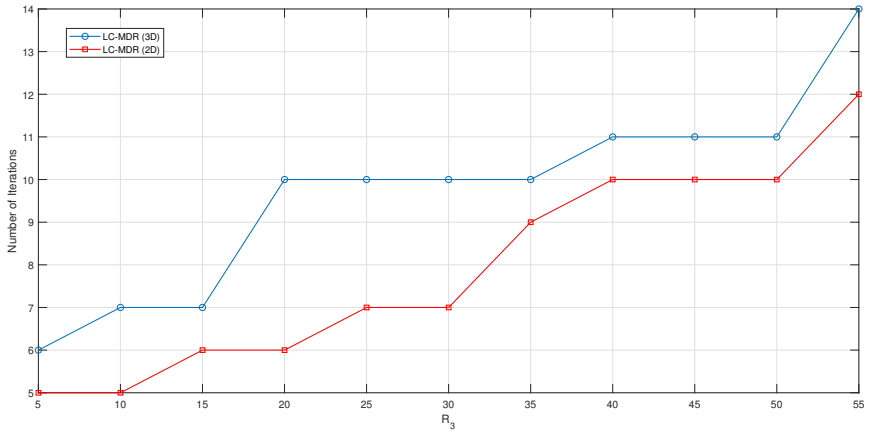
provided the second lowest FLOPS and the MPCA is the most time-complex method.

Moreover, both versions of the LC-MDR provided a FLOP count roughly 2.3 times less than the standard PCA. This result pictures a clear trade-off between accuracy performance and computational cost of the two LC-MDR techniques, where the 2D runs faster, but it achieves lower accuracy values in comparison to the 3D case.

In Figure 22, the FLOP count versus the number of attributes R_3 is shown, for the 3D LC-MDR, the PCA and the MPCA, with $R_1 = R_2 = 2$. As it can be viewed, the MPCA demands high computational cost, in terms of FLOPS, when compared to the LC-MDR. In Tab. 19 only one scenario was shown, which was for dimensions $2 \times 2 \times 10$, however, in Figure 22 we can see a better picture of the FLOPS for various attribute sizes. It is clear that when the attribute size R_3 augments, the FLOP count increases dramatically for the MPCA, showing that this technique is very computational demanding. The vector-based PCA performed closely to the tensor-based LC-MDR, thus further demonstrating the lower computational complexity of the LC-MDR, even it being a tensor-based technique.

Next, in Figure 23, we have the analysis of the number of iterations necessary to achieve convergence of the LC-MDR algorithm, in function of the parameter R_3, which is the

Figure 23 – Iterations needed for convergence of the 3D LC-MDR when varying R_3, with $R_1 = R_2 = 2$.



Source: Author.

number of attributes and with $R_1 = R_2 = 2$. As Fig. 23 shows, the increase in R_3 demands more iterations, however, not by a large margin, thus explaining the small difference in execution times of Table 16 for the dimensions $2 \times 2 \times 10$ and $2 \times 2 \times 15$.

4.6 Conclusions

In this chapter, a dimensionality reduction technique for tensor data is proposed. The presented method optimizes a cost function that takes into account the data correlation, generating variables with much less correlation than the MPCA. In particular, the LC-MDR fits the input data correlation into a new tensor decomposition denoted by EONPD, which is an extension of the NPD for higher-order tensors. Moreover, the LC-MDR is based on the ALS for estimating the factor matrices. A complete description of the EONPD is given, with both analytical and recursive expressions being derived. In addition, a generalization of the EONPD, denoted by HONPD, is also presented, assuming the nesting of PARAFAC tensors of generic orders.

The proposed technique is used for the classification of five different volcano-seismic events using a database of 3D samples recorded from the Ubinas volcano, Peru, in 2009, with the tensor samples being organized as *stations* $\times$ *channels* $\times$ *features*.

The results showed very significant gains in accuracy of the LC-MDR over the concurrent PCA and MPCA. The better performance of the proposed method is observed with and without dimensionality reduction. The best accuracy (90.3%) was achieved with the 3D LC-MDR coupled with the STuM classifier, using dimensionality reduction. In addition, the

results showed the LC-MDR provides output data with much less correlation than the MPCA, which explains the better accuracy of the LC-MDR. Moreover, the LC-MDR showed a faster execution time in comparison with the MPCA. Besides, the proposed technique also provides smaller execution times than the PCA if dimensionality reduction is carried out.

5 MULTILINEAR SAMPLING IN SUPPORT VECTOR MACHINES FOR PHOTONIC DATA CLASSIFICATION

In this chapter, the concept of multilinear sampling is proposed, with the objective of presenting classification algorithms that exploit a multidimensional structure of the samples. The multilinear sampling is used to perform a modification on the SVM classifier, where it is considered two tensorial decompositions for the slack variables. The proposed classification methods exploit the multidimensional modeling of both PARAFAC and Tucker decompositions, therefore, the modifications are called SVM-MPS and SVM-MTS, respectively. Then, the modified SVM algorithms are used to classify the binary outputs obtained from a MZI. Usually, the the output data obtained from interferometers depend on many parameters, which can be arranged in a multilinear sample structure. The multilinear sampling approach is then used by storing the input samples into a four-way array, whose modes corresponds to the two interferometer inputs and two phase shifts.

Results showed better accuracy of the proposed SVM modifications in classifying the logic levels of the MZI output, comparing to the standard SVM and other classic classifers. The proposed multidimensional sampling structure proved to be effective in classification when compared to the conventional vectorization processes. Also computational cost and running times were taken as parameters to validate the proposed approach.

This chapter is organized as follows. First, the motivations of the proposed technique are outlined, then, the modeling of the multilinear sampling structure is described, with the proposed modifications for the SVM, with its Quadratic Programming (QP) and algorithms formulated in the following. Next, the used photonic database is described and detailed. For last, the obtained results are discussed and commented.

5.1 Motivation

An alternative to conventional approaches for ML are the tensor learning frameworks, where many conventional learning machines can be generalized to take n-order tensors as inputs (TAO *et al.*, 2007b). As earlier highlighted, this avoids data vectorization and the consequent destruction of the data structure (HE *et al.*, 2017). In tensor learning methods, the multidimensional structural information of the data is preserved.

In classical tensor learning, the multidimensional structure lies in the attributes, which are considered to be tensors. On the other, hand, in the multilinear approach proposed in

this chapter, the multidimensional structure lies in the sampling. In other words, all but one of the dimensions of the tensor correspond to the sampling modes, which can be exploited during the processing and classification process. The main idea behind the multilinear sampling approach is to exploit the multidimensional structure of the samples.

Moreover, multilinear sampling can be combined with conventional linear techniques to generate algorithms that support multidimensional data, without the need for vectorization of the sampling modes.

The multidimensional structure of the multilinear sampling approach relies in the way samples were obtained and not in the structure of the input vector, as each input sample is considered to be a vector, as in standard ML techniques. In a traditional ML approach, all the samples would be concatenated into a single sampling index, however, this will break the sampling structure of the data. This would cause loss of information and, hence, loss of performance, in accuracy terms.

There is no work in the literature that exploits a similar multilinear sampling structure for ML tasks, as in this thesis. In fact, the proposed multilinear sampling approach is original in the context of ML and TL, as it joins both topics.

Hence, the focus of this chapter is to present an extension of the SVM algorithm that can make use of multidimensional sampling. The main original contributions of the present chapter are summarized as follows. (i) The concept of multilinear sampling approach, (ii) PARAFAC-based and Tucker-based modifications of the SVM algorithm, (iii) a QP formulation of the proposed problems and (iv) application of the proposed technique for the classification of logic levels at the output of a MZI.

5.2 Multilinear Sampling in Support Vector Machines

In this section, the multilinear sampling approach is presented in the context of a SVM for binary data classification. Firstly, the multilinear sampling approach is presented and, then, an adaptation of the primal formulation of the SVM to the multilinear sampling is presented. After that, in the next section, two solutions for the proposed formulation are presented, based on the modeling of the slacks variables as PARAFAC and Tucker decompositions. These solutions give rise to two new classifier's that exploit the multilinear sampling approach.

5.2.1 Multilinear Sampling

Let us consider a training data set consisting of P input vectors denoted by $\mathbf{x}_p \in \mathbb{C}^N$, for $p = 1,...,P$, where N is the dimension of the input vector. The basic assumption of the multilinear sampling approach is that the P samples were originally obtained obeying a multidimensional structure with N dimensions, i.e. the P samples were obtained as a $(P+1)^{th}$ order tensor $\mathscr{X} \in \mathbb{C}^{N \times P_1 \times ... \times P_N}$, with the last N dimensions corresponding to sampling dimensions denoted by $P_1,...,P_N$, where $P = P_1,...,P_N$. The indices associated with the N sampling dimensions are denoted by $p_1,...,p_N$, where $p = p_N + (p_{N-1} - 1)P_N + ... + (p_1 - 1)P_N...P_2$, for $1 \geq p_n \geq P_n$ and $n = 1,...,N$. In other words, index p corresponds to the concatenation of the indices $p_1,...,p_N$, as well as the the dimension P corresponds to the concatenation of the dimensions $P_1,...,P_N$.

When ML methods are used to model dynamic systems, it is usual to construct a training database in which a scan of the input variables is carried out, i.e. the training database is constructed by varying each input in a certain range at each time. This scan of input of variables usually follows the above mentioned multilinear sampling structure.

Indeed, let us consider that the training database is constructed by making the n^{th} element of the input vector assume the values $\{\beta_1^{(n)}...\beta_{P_n}^{(n)}\}$, for $n = 1,...,N$, where P_n is number of tested values of the n^{th} input. In this case, the whole database is composed of $P = P_1,...,P_N$ samples, given by:

$$\mathbf{x}_p = \mathbf{x}_{p_1,...,p_N} = \begin{bmatrix} \beta_{p_1}^{(1)} \\ \vdots \\ \beta_{p_N}^{(N)} \end{bmatrix} \in \mathbb{C}^{N \times 1}, \tag{5.1}$$

for $p_1 = 1,...,P_1$,..., $p_N = 1,...,P_N$. Notice that the database contains all combinations of $\beta_{p_n}^{(n)}$, for $1 \leq p_n \leq P_n$ and $n = 1,...,N$. The database can then be organized in a tensor $\mathscr{X} \in \mathbb{C}^{N \times P_1 \times ... \times P_N}$, whose elements are denoted by $x_{n,p_1,...,p_N}$.

Therefore, the main idea behind the multilinear sampling is to exploit the multidimensional structure of the samples. Note that the refereed multidimensional structure relies in the way the samples were obtained and not in the structure of the input vector, as each input sample is a vector. In a traditional ML approach, all the samples will be concatenated into a single sampling index p, however, this will break the sampling structure of the data. This could cause loss of information and, hence, loss of performance.

The above multidimensional structure for the samples is common in the literature, specially when ML techniques are used to model dynamic systems, as, for instance, in optics and photonics fields. In (XIE *et al.*, 2020), an arbitrary ratio optical power splitter is designed with data structured in a similar way as in this chapter. Moreover, in (KUMAR *et al.*, 2014) and in (ARAÚJO *et al.*, 2015; CORREIA *et al.*, 2017; SOUSA *et al.*, 2014), logic gates are implemented through MZIs, with sets of parameters and inputs similar to the ones used in this thesis, which could exploit the multlinear sampling structures presented here. Additionally, the works of (MENGU *et al.*, 2022) and (TAHERSIMA *et al.*, 2019) that exploit inverse logic design in ML, utilizes similar arrangements as the ones used here, for parameters and data obtained from MZIs and other optic devices.

In the sequel, the multilinear sampling is applied in the task of statistical classification using the SVM, which is one of the most popular classifiers in the literature, used in a wide variety of applications (BURGES, 1998; MATHUR; FOODY, 2008; CERVANTES *et al.*, 2020).

In particular, the primal cost function of the SVM is modified to cope with the multiple indices of the multilinear sampling approach, aiming to exploit the multilinear structure of the sampling. This gives rise to two different modifications on the SVM algorithm. The first one based on the PARAFAC decomposition, while the second one uses the Tucker decomposition.

5.2.2 Primal Formulation of the SVM with Multilinear Sampling

Considering the data samples follows the earlier described multilinear sampling, the SVM problem given by (2.32)-(2.34) can be stated as:

$$\min_{\mathbf{w},b,\mathscr{E}} \ \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle + C\sum_{p_1=1}^{P_1}\ldots\sum_{p_N=1}^{P_N}\xi_{p_1,\ldots,p_N}, \tag{5.2}$$

subject to

$$y_{p_1,\ldots,p_N}\left(\langle \mathbf{w},\mathbf{x}_{p_1,\ldots,p_N}\rangle + b\right) \geq 1-\xi_{p_1,\ldots,p_N}, \tag{5.3}$$

$$\xi_{p_1,\ldots,p_N} \geq 0, \tag{5.4}$$

for $p_1 = 1,\ldots,P_1$,..., $p_N = 1,\ldots,P_N$, where $\mathscr{E} \in \mathbb{C}^{P_1\times\ldots\times P_N}$ is a N^{th} order tensor that contains the slack variables ξ. Comparing the above formulation with the one of Chapter 2, we have that the index p in (2.32)-(2.34) was replaced by the multiple sampling indices $p_1,\ldots,p_N$ in (5.2)-(5.4). In the next two sections, two methods for solving the formulation (5.2)-(5.4) are proposed, using the PARAFAC and Tucker decompositions of the slack tensor $\mathscr{E}$.

5.3 SVM with Multilinear PARAFAC Sampling (SVM-MPS)

In this section, the problem formulated in (5.2)-(5.4) is modified by using the PARAFAC decomposition of the tensor $\mathscr{E}$, which contains the slack variables ξ. After that, the problem is expressed as a QP problem and, then, the algorithm of the SVM classifier with multilinear PARAFAC sampling (SVM-MPS) is presented.

5.3.1 Problem Formulation

A solution to (5.2)-(5.4) can be obtained by exploiting the PARAFAC decompositions of the slack tensor $\mathscr{E}$ and using multiple convectional SVMs. The PARAFAC decomposition of slack tensor $\mathscr{E}$ can expressed as:

$$\xi_{p_1,\ldots,p_N} = \sum_{q=1}^{Q} a^{(1)}_{p_1,q} \ldots a^{(N)}_{p_N,q}, \tag{5.5}$$

where Q is the tensor rank and $a^{(1)}_{p_1,q}$,..., $a^{(N)}_{p_N,q}$ are the factors of the PARAFAC decomposition, which can be organized in the factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{P_n \times Q}$, for $n = 1,\ldots,N$.

Using (5.5), eqs. (5.2)-(5.4) can be rewritten as:

$$\min_{\mathbf{w},b,\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}} \quad \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle + C\left(\sum_{p_1=1}^{P_1} \ldots \sum_{p_N=1}^{P_N} \sum_{q=1}^{Q} a^{(1)}_{p_1,q} \ldots a^{(N)}_{p_N,q}\right), \tag{5.6}$$

subject to

$$y_{p_1,,\ldots,p_N}\left(\langle \mathbf{w}, \mathbf{x}_{p_1,\ldots,p_N}\rangle + b\right) \geq 1 - \sum_{q=1}^{Q} a^{(1)}_{p_1,q} \ldots a^{(N)}_{p_N,q}, \tag{5.7}$$

$$\sum_{q=1}^{Q} a^{(1)}_{p_1,q} \ldots a^{(N)}_{p_N,q} \geq 0, \tag{5.8}$$

for p_1 = 1,...,P_1, ..., p_N = 1,...,P_N.

The proposed classifier is an iterative algorithm with multiple alternated steps at each iteration, similarly to the ALS algorithm (COMON *et al.*, 2009). At each step of each iteration, one of the factor matrices of the slack tensor $\mathscr{E}$ is estimated assuming that the other factor matrices are known, using the previous estimates of these factor matrices. The process goes on until convergence, estimating a different factor matrix at each step. The weight vector $\mathbf{w}$ is estimated in each step of each iteration, being refined with each new step.

Assuming that all the factor matrices are known, except for $\mathbf{A}^{(n)}$, eqs. (5.6)-(5.8) can be rewritten as follows:

$$\min_{\mathbf{w},b,\mathbf{A}^{(n)}} \quad \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle + C\left(\sum_{q=1}^{Q} C_q^{[n]} \sum_{p_n=1}^{P_n} a_{p_n,q}^{(n)}\right), \tag{5.9}$$

subject to

$$y_{p_1,\ldots,p_N}\left(\langle \mathbf{w}, \mathbf{x}_{p_1,\ldots,p_N}\rangle + b\right) \geq 1 - \sum_{q=1}^{Q} a_{p_n,q}^{(n)} u_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,q}^{[n]}, \tag{5.10}$$

$$\sum_{q=1}^{Q} a_{p_n,q}^{(n)} u_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,q}^{[n]} \geq 0, \tag{5.11}$$

for $p_1 = 1,\ldots,P_1$, ..., $p_N = 1,\ldots,P_N$, where

$$C_q^{[n]} = \sum_{p_1=1}^{P_n} \ldots \sum_{p_{n-1}=1}^{P_{n-1}} \sum_{p_{n+1}=1}^{P_{n+1}} \ldots \sum_{p_N=1}^{P_N} a_{p_1,q}^{(1)} \ldots a_{p_{n-1},q}^{(n-1)} a_{p_{n+1},q}^{(n+1)} \ldots a_{p_N,q}^{(N)} \tag{5.12}$$

and

$$u_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,q}^{[n]} = a_{p_1,q}^{(1)} \ldots a_{p_{n-1},q}^{(n-1)} a_{p_{n+1},q}^{(n+1)} \ldots a_{p_N,q}^{(N)} \tag{5.13}$$

are assumed to be known. Despite being different from the standard SVM formulation (2.32)-(2.34), the optimization problem in (5.9)-(5.11) can also be expressed as QP, as it will be shown in the next subsection.

5.3.2 Quadratic Programming (QP) Formulation

QP is a type of nonlinear programming for solving optimization problems involving quadratic functions subject to linear constraints. There are a great variety of well-established method for QP, including interior point, active set, augmented Lagrangian, conjugate gradient etc (NOCEDAL; WRIGHT, 2006). A general formulation for a QP is given by:

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\mathbf{q} \tag{5.14}$$

subject to

$$\mathbf{R}\mathbf{x} \leq \mathbf{s}, \tag{5.15}$$

The cost function (5.9) can be expressed as in the general form (5.14) with the following correspondences:

$$\mathbf{x} = \left[\mathbf{w}\ b\ a_{1,1}^{(n)} \ldots a_{1,Q}^{(n)} \cdots a_{P_n,1}^{(n)} \ldots a_{P_n,Q}^{(n)}\right]^T \in \mathbb{R}^{(N+P_nQ+1)\times 1}, \tag{5.16}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_{N,N} & \mathbf{0}_{N,(P_nQ+1)} \\ \mathbf{0}_{(P_nQ+1),N} & \mathbf{0}_{(P_nQ+1),(P_nQ+1)} \end{bmatrix} \in \mathbb{R}^{(N+P_nQ+1)\times(N+P_nQ+1)}, \tag{5.17}$$

$$\mathbf{q} = \begin{bmatrix} \mathbf{0}_{(N+1),1} \\ \\ \mathbf{1}_{P_n,1} \otimes \mathbf{c}^{[n]} \end{bmatrix} \in \mathbb{R}^{(N+P_nQ+1)\times 1}, \tag{5.18}$$

where $\mathbf{c}^{[n]} = [C_1^{[n]} ... C_Q^{[n]}] \in \mathbb{R}^{Q\times 1}$.

Moreover, the constraints (5.10) and (5.11) can be expressed as in the general form (5.15) with the following correspondences:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}^{(1)} \\ \\ \mathbf{R}^{(2)} \end{bmatrix} \in \mathbb{R}^{2P\times(2+P_nQ+1)}, \ \mathbf{s} = \begin{bmatrix} \mathbf{s}^{(1)} \\ \\ \mathbf{s}^{(2)} \end{bmatrix} \in \mathbb{R}^{2P\times 1}, \tag{5.19}$$

where $\mathbf{s}^{(1)} = -\mathbf{1}_{P,1} \in \mathbb{R}^{P\times 1}$ and the p^{th} row of $\mathbf{R}^{(1)} \in \mathbb{R}^{P\times(N+P_nQ+1)}$, for $p = 1,...,P$ and $P = P_1...P_N$, is given by:

$$[\mathbf{R}^{(1)}]_{p,:} = -\begin{bmatrix} y_{p_1,...,p_N} x_{1,p_1,...,p_N} \\ \vdots \\ y_{p_1,...,p_N} x_{N,p_1,...,p_N} \\ y_{p_1,...,p_N} \\ \mathbf{0}_{Q(p_n-1),1} \\ -u^{[n]}_{p_1,...,p_{n-1},p_{n+1},...,p_N,1} \\ \vdots \\ -u^{[n]}_{p_1,...,p_{n-1},p_{n+1},...,p_N,Q} \\ \mathbf{0}_{Q(P_n-p_n),1} \end{bmatrix}^T \in \mathbb{R}^{P\times(N+P_nQ+1)}, \tag{5.20}$$

with p corresponding to the concatenation of the indices $p_1,...,p_N$ in the following way $p = p_N + (p_{N-1}-1)P_N + ... + (p_1-1)P_N...P_2$, for $1 \geq p_n \geq P_n$ and $n = 1,...,N$.

Furthermore, $\mathbf{s}^{(2)} = \mathbf{0}_{P,1} \in \mathbb{R}^{P\times 1}$ and the p^{th} row of $\mathbf{R}^{(2)} \in \mathbb{R}^{P\times(N+P_nQ+1)}$, for $p =$

Algorithm 2: SVM-MPS Estimation Algorithm

Input: ε, C, $x_{n,p_1,p_2,...,p_N}$ and $y_{p_1,p_2,...,p_N}$, for $p_n = 1,...,P_n$ and $n = 1,...,N$.
Outputs: $\hat{\mathbf{w}}, \hat{b}, \hat{\mathscr{E}}$
Initialization: $\beta = 0$, $i = 0$, $\hat{\mathbf{A}}_0^{(2)}, \ldots, \hat{\mathbf{A}}_0^{(N)}$ are random matrices
while $\beta = 0$ **do**
  $i = i + 1$
  for $n = 1 : N$ **do**
    Construct $\mathbf{P}$, $\mathbf{q}$, $\mathbf{R}$ and $\mathbf{s}$ from (5.17)-(5.21) using $\hat{\mathbf{A}}_i^{(1)}$,..., $\hat{\mathbf{A}}_i^{(n-1)}$, $\hat{\mathbf{A}}_{i-1}^{(n+1)}$, ...,$\hat{\mathbf{A}}_{i-1}^{(N)}$.
    Using any QP method, find $\hat{\mathbf{w}}_i, \hat{b}_i, \hat{\mathbf{A}}_i^{(n)}$
  end
  Build $\hat{\mathscr{E}}_i$ using $\hat{\mathbf{A}}_i^{(1)}, \ldots, \hat{\mathbf{A}}_i^{(N)}$ using (5.5)
  Unfold $\hat{\mathscr{E}}_i$ into $\hat{\mathbf{E}}_i^{[1]}$ using (2.18)
  if $|e_i| < \varepsilon$ **then**
    $\beta = 1$
    $\hat{\mathbf{w}} = \hat{\mathbf{w}}_i, \hat{b} = \hat{b}_i, \hat{\mathscr{E}} = \hat{\mathscr{E}}_i$
  end
end

$1,...,P$ and $P = P_1,...,P_N$, is given by:

$$[\mathbf{R}^{(2)}]_{p,:} = \begin{bmatrix} \mathbf{0}_{N+1+Q(p_n-1),1} \\ -u^{[n]}_{p_1,...,p_{n-1},p_{n+1},...,p_N,1} \\ \vdots \\ -u^{[n]}_{p_1,...,p_{n-1},p_{n+1},...,p_N,Q} \\ \mathbf{0}_{Q(P_n-p_1),1} \end{bmatrix}^T, \tag{5.21}$$

where p corresponds to the concatenation of the indices $p_1,...,p_N$, as in (5.20). Once the problem shown in (5.9)-(5.11) of the SVM-MPS is expressed as QP, it can be solved by any of the earlier mentioned QP methods.

5.3.3 Estimation Algorithm

The estimation algorithm of the SVM-MPS is shown in Algorithm 2. Given a variable $\mathbf{z}$, the estimation of $\mathbf{z}$ at the i^{th} iteration is denoted by $\hat{\mathbf{z}}_i$. Each iteration of the algorithm is composed of N stages, in each stage, one factor matrix of the PARAFAC decomposition of the slack tensor $\mathscr{E}$ is estimated using the previous estimation of the other factor matrices. This estimation is carried out using any QP method with the formulation presented given by (5.17)-(5.21). In each stage of each iteration, the weight vector $\mathbf{w}$ and the bias scalar b are also estimated.

At the end of the N stages, the tensor $\hat{\mathscr{E}}_i$ is constructed, unfolded and a test for convergence is performed using the binary flag variable β and:

$$e_i = \frac{\|\hat{\mathbf{E}}_i^{[1]} - \hat{\mathbf{E}}_{i-1}^{[1]}\|_F^2}{\|\hat{\mathbf{E}}_{i-1}^{[1]}\|_F^2}, \tag{5.22}$$

where ε is a small scalar constants. When the algorithm stops, the last values obtained for $\mathbf{w}$ and b are saved to be used during the test phase of the classification.

5.4 SVM with Multilinear Tucker Sampling (SVM-MTS)

In this section, the problem formulated in (5.2)-(5.4) is modified by using the Tucker decomposition of the tensor $\mathscr{E}$. Then, the problem is expressed as a QP problem and the algorithm of the SVM classifier with multilinear Tucker sampling (SVM-MTS) is presented.

5.4.1 Problem Formulation

Let $\mathscr{E} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}$ be the n-th order tensor, with n = 1,...,N, formed from $\xi_{p_1,p_2,\ldots,p_N}$. Let us assume that the slack tensor $\mathscr{E}$ follows a Tucker decomposition:

$$\xi_{p_1,p_2,\ldots,p_N} = \sum_{q_1=1}^{Q_1} \sum_{q_2=1}^{Q_2} \ldots \sum_{q_N=1}^{Q_N} g_{q_1,q_2,\ldots,q_N} a^{(1)}_{p_1,q_1} a^{(2)}_{p_2,q_2} \ldots a^{(N)}_{p_N,q_N}, \tag{5.23}$$

where $a^{(1)}_{p_1,q_1}$, $a^{(2)}_{p_2,q_2}$, ..., $a^{(N)}_{p_N,q_N}$, form the matrix factors of the Tucker decomposition $\mathbf{A}^{(1)} \in \mathbb{R}^{P_1 \times Q_1}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{P_2 \times Q_2}$, ..., $\mathbf{A}^{(N)} \in \mathbb{R}^{P_N \times Q_N}$, and $g_{q_1,q_2,\ldots,q_N}$ forms the core tensor of the Tucker decomposition $\mathscr{G} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}$.

Hence, the optimization problem of (5.2)-(5.4) may be stated as follows:

$$\min_{\mathbf{w},b,\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}} \quad \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \left(\sum_{p_1=1}^{P_1} \sum_{p_2=1}^{P_2} \ldots \sum_{p_N=1}^{P_N} \sum_{q_1=1}^{Q_1} \sum_{q_2=1}^{Q_2} \ldots \sum_{q_N=1}^{Q_N} g_{q_1,q_2,\ldots,q_N} a^{(1)}_{p_1,q_1} a^{(2)}_{p_2,q_2} \ldots a^{(N)}_{p_N,q_N} \right), \tag{5.24}$$

subject to

$$y_{p_1,p_2,\ldots,p_N} - (\langle \mathbf{w}, \mathbf{x}_{p_1,p_2,p_3,p_4} \rangle + b) \leq 1 - \sum_{q_1=1}^{Q_1} \sum_{q_2=1}^{Q_2} \ldots \sum_{q_N=1}^{Q_N} g_{q_1,q_2,\ldots,q_N} a^{(1)}_{p_1,q_1} a^{(2)}_{p_2,q_2} \ldots a^{(N)}_{p_N,q_N}, \tag{5.25}$$

$$\sum_{q_1=1}^{Q_1} \sum_{q_2=1}^{Q_2} \ldots \sum_{q_N=1}^{Q_N} g_{q_1,q_2,\ldots,q_N} a^{(1)}_{p_1,q_1} a^{(2)}_{p_2,q_2} a^{(N)}_{p_N,q_N} \geq 0, \tag{5.26}$$

for p_1 = 1,...,P_1, p_2 = 1,...,P_2, ..., p_N = 1,...,P_N.

In order to find $a^{(1)}_{p_1,q_1}$, $a^{(2)}_{p_2,q_2}$, ..., $a^{(N)}_{p_N,q_N}$ and $g_{q_1,q_2,...,q_N}$, we estimate them iteratively. Similar to the SVM-MPS estimation, the proposed SVM-MTS is an iterative algorithm with multiple alternated stages. In the first N stages, at each iteration, one of the factor matrices of the slack tensor $\mathscr{E}$ is estimated assuming that the other factor matrices and the core tensor are known, using the previous estimates of them. The process goes on, estimating a different factor matrix at each stage. After all factor matrices are estimated, the core tensor obtained in the following stage, with the estimated factor matrices used to estimate the core tensor. This process repeats until convergence, where the weight vector $\mathbf{w}$ is estimated in each step of each iteration, being refined with each new step.

Assuming that the core tensor and all the factor matrices are known, except for $\mathbf{A}^{(n)}$, eqs. (5.24)-(5.26) are rewritten as follows:

$$\min_{\mathbf{w},b,\mathbf{A}^{(n)}} \quad \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle + C\left(\sum_{q_n=1}^{Q_n} C^{[n]}_{q_n} \sum_{p_n=1}^{P_n} a^{(n)}_{p_n,q_n}\right), \tag{5.27}$$

subject to

$$y_{p_1,p_2,...,p_N} - (\langle \mathbf{w}, \mathbf{x}_{p_1,p_2,...,p_N}\rangle + b) \leq 1 - \sum_{q_n=n}^{Q_n} a^{(n)}_{p_n,q_n} u^{[n]}_{p_1,...,p_{n-1},p_{n+1},...,p_N,q_n}, \tag{5.28}$$

$$\sum_{q_n=1}^{Q_n} a^{(n)}_{p_n,q_n} u^{[n]}_{p_1,...,p_{n-1},p_{n+1},...,p_N,q_n} \geq 0. \tag{5.29}$$

for p_1 = 1,...,P_1, p_2 = 1,...,P_2, ..., p_N = 1,...,P_N, where $C^{[n]}_{q_n}$ and $u^{[n]}_{p_1,...,p_{n-1},p_{n+1},...,p_N,q_n}$ are given by:

$$\begin{aligned} C^{[n]}_{q_n} = \sum_{p_1=1}^{P_1} \cdots \sum_{p_{n-1}=1}^{P_{n-1}} \sum_{p_{n+1}=1}^{P_{n+1}} \cdots \sum_{p_N=1}^{P_N} \sum_{q_1=1}^{Q_1} \cdots \sum_{q_{n-1}=1}^{Q_{n-1}} \sum_{q_{n+1}=1}^{Q_{n+1}} \cdots \sum_{q_N=1}^{Q_N} g_{q_1,q_2,...,q_n} \times \\ \times\, a^{(1)}_{p_1,q_1} \ldots a^{(n-1)}_{p_{n-1},q_{n-1}} a^{(n+1)}_{p_{n+1},q_{n+1}} \ldots a^{(N)}_{p_N,q_N}, \end{aligned} \tag{5.30}$$

$$u^{[n]}_{p_1,...,p_{n-1},p_{n+1},...,p_N,q_n} = \sum_{q_1=1}^{Q_1} \cdots \sum_{q_{n-1}=1}^{Q_{n-1}} \sum_{q_{n+1}=1}^{Q_{n+1}} \cdots \sum_{q_N=1}^{Q_N} g_{q_1,q_2,...,q_N} a^{(1)}_{p_1,q_1} \ldots a^{(n-1)}_{p_{n-1},q_{n-1}} a^{(n+1)}_{p_{n+1},q_{n+1}} \ldots a^{(N)}_{p_N,q_N}. \tag{5.31}$$

Both $C^{[n]}_{q_n}$ and $u^{[n]}_{p_1,...,p_{n-1},p_{n+1},...,p_N,q_n}$ are assumed to be known. After the estimations of $\mathbf{A}^{(1)}_{p_1,q_1}$, $\mathbf{A}^{(2)}_{p_2,q_2}$, ..., $\mathbf{A}^{(N)}_{p_N,q_N}$, the core tensor $g_{q_1,q_2,...,q_N}$ is estimated. Hence, eqs. (5.24)-(5.26) are rewritten in the following way:

$$\min_{\mathbf{w},b,g_{q_1,q_2,\ldots,q_N}} \quad \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle + C\left(\sum_{q_1=1}^{Q_1}\sum_{q_2=1}^{Q_2}\ldots\sum_{q_N=1}^{Q_N} C^{[g]}_{q_1,q_2,\ldots,q_N} g_{q_1,q_2,\ldots,q_N}\right), \tag{5.32}$$

subject to

$$y_{p_1,p_2,\ldots,p_N} - (\langle \mathbf{w}, \mathbf{x}_{p_1,p_2,\ldots,p_N}\rangle + b) \leq 1 - \sum_{q_1=1}^{Q_1}\sum_{q_2=1}^{Q_2}\ldots\sum_{q_N=1}^{Q_N} g_{q_1,q_2,\ldots,q_N} u^{[g]}_{p_1,p_2,\ldots,p_N}, \tag{5.33}$$

$$\sum_{q_1=1}^{Q_1}\sum_{q_2=1}^{Q_2}\ldots\sum_{q_N=1}^{Q_N} g_{q_1,q_2,\ldots,q_N} u^{[g]}_{p_1,p_2,\ldots,p_N} \geq 0, \tag{5.34}$$

with $p_1 = 1,\ldots,P_1$, $p_2 = 1,\ldots,P_2$, ..., $p_N = 1,\ldots,P_N$, where $C^{[g]}_{q_1,q_2,\ldots,q_N}$ and $u^{[g]}_{p_1,p_2,\ldots,p_N,q_1}$ are given by:

$$C^{[g]}_{q_1,q_2,\ldots,q_N} = \sum_{p_1=1}^{P_1}\sum_{p_2=1}^{P_2}\ldots\sum_{p_N=1}^{P_N} a^{(1)}_{p_1,q_1} a^{(2)}_{p_2,q_2}\ldots a^{(N)}_{p_N,q_N}, \tag{5.35}$$

$$u^{[g]}_{p_1,p_2,\ldots,p_N} = a^{(1)}_{p_1,q_1} a^{(2)}_{p_2,q_2}\ldots a^{(N)}_{p_N,q_N}. \tag{5.36}$$

Both $C^{[g]}_{q_1,q_2,\ldots,q_N}$ and $u^{[g]}_{p_1,p_2,\ldots,p_N}$ are assumed to be known. The optimization problems in (5.27)-(5.29) and (5.32)-(5.34) can also be expressed as QP, which is shown in the next subsection.

5.4.2 Quadratic Programming Formulation

The QP formulation for the Tucker representation of the slack variables is very similar to the PARAFAC one. In fact, (5.16)-(5.18), can be reused by just substituting Q by Q_N, with $\mathbf{c}^{[n]} \in \mathbb{R}^{Q_n \times 1}$ containing all $C^{[n]}_{q_n}$ values, in the following way:

$$\mathbf{x} = \left[\mathbf{w}\ b\ a^{(n)}_{1,1}\ldots a^{(n)}_{1,Q_n}\cdots a^{(n)}_{P_n,1}\ldots a^{(n)}_{P_n,Q_n}\right]^T \in \mathbb{R}^{(N+P_nQ_n+1)\times 1}, \tag{5.37}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_{N,N} & \mathbf{0}_{N,(P_nQ_n+1)} \\ \mathbf{0}_{(P_nQ_n+1),N} & \mathbf{0}_{(P_nQ_n+1),(P_nQ_n+1)} \end{bmatrix} \in \mathbb{R}^{(N+P_nQ_n+1)\times(N+P_nQ_n+1)}, \tag{5.38}$$

$$\mathbf{q} = \begin{bmatrix} \mathbf{0}_{(N+1),1} \\ \\ \mathbf{1}_{P_n,1}\otimes\mathbf{c}^{[n]} \end{bmatrix} \in \mathbb{R}^{(N+P_nQ_n+1)\times 1}, \tag{5.39}$$

where $\mathbf{c}^{[n]} = [C_1^{[n]} ... C_{Q_n}^{[n]}] \in \mathbb{R}^{Q_n \times 1}$. Next, (5.19) stays the same but with dimensions related to rank Q_n instead of Q, and $\mathbf{s}$ is not changed at all, thus:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}^{(1)} \\ \\ \mathbf{R}^{(2)} \end{bmatrix} \in \mathbb{R}^{2P \times (2+P_n Q_n+1)}, \tag{5.40}$$

with $\mathbf{R}^{(1)}$ given by (5.20) but with dimensions $\mathbb{R}^{P \times (N+P_n Q_n+1)}$ and $\mathbf{R}^{(2)}$ is given by (5.21), with dimensions $\mathbb{R}^{P \times (N+P_n Q_n+1)}$. After these QP procedures, one extra step is performed in order to estimate the core tensor $\mathscr{G}$, which is described in the following. We have that the cost function (5.32) can be expressed as in the general form (5.14) with the following correspondences:

$$\mathbf{x}^{[g]} = \left[\mathbf{w}\ b\ g_{1,1,...,1} \cdots g_{Q_1,1,...,1} \cdots g_{Q_1,Q_2,...,1} \cdots g_{Q_1,Q_2,...,Q_N}\right]^T \in \mathbb{R}^{(N+Q_1 ... Q_N+1) \times 1}, \tag{5.41}$$

$$\mathbf{P}^{[g]} = \begin{bmatrix} \mathbf{I}_{N,N} & \mathbf{0}_{N,(Q_1 ... Q_N+1)} \\ \mathbf{0}_{(Q_1 ... Q_N+1),N} & \mathbf{0}_{(Q_1 ... Q_N+1),(Q_1 ... Q_N+1)} \end{bmatrix} \in \mathbb{R}^{(N+Q_1 ... Q_N+1) \times (N+Q_1 ... Q_N+1)}, \tag{5.42}$$

$$\mathbf{q}^{[g]} = \begin{bmatrix} \mathbf{0}_{(N+1),1} \\ \\ \mathbf{1}_{Q_1 ... Q_N,1} \otimes \mathbf{c}^{[g]} \end{bmatrix} \in \mathbb{R}^{(N+Q_1 ... Q_N+1) \times 1}, \tag{5.43}$$

where $\mathbf{c}^{[g]} = [C^{[g]}_{1,...,1} ... C^{[g]}_{Q_1,...,Q_N}] \in \mathbb{R}^{Q_1 ... Q_N \times 1}$, with $q_1 = 1, ..., Q_1$, ..., $q_N = 1, ..., Q_N$. Moving on, $\mathbf{R}^{[g]}$ and $\mathbf{s}^{[g]}$ are obtained the same way as in (5.40), hence:

$$\mathbf{R}^{[g]} = \begin{bmatrix} \mathbf{R}^{[g](1)} \\ \\ \mathbf{R}^{[g](2)} \end{bmatrix} \in \mathbb{R}^{2Q_1 ... Q_N \times (N+Q_1 ... Q_N+1)},\ \mathbf{s}^{[g]} = \begin{bmatrix} \mathbf{s}^{[g](1)} \\ \\ \mathbf{s}^{[g](2)} \end{bmatrix} \in \mathbb{R}^{2Q_1 ... Q_N \times 1}, \tag{5.44}$$

where $\mathbf{s}^{[g](1)} = -\mathbf{1}_{Q_1 ... Q_N,1}$, $\mathbf{s}^{[g](2)} = \mathbf{0}_{Q_1 ... Q_N,1}$ and a single row of $\mathbf{R}^{[g](1)} \in \mathbb{R}^{Q_1 ... Q_N \times (N+Q_1 ... Q_N+1)}$, is given by:

$$[\mathbf{R}^{[g](1)}]_{q_1,\ldots,q_N,:} = -\begin{bmatrix} y_{p_1,\ldots,p_N} x_{1,p_1,\ldots,p_N} \\ \vdots \\ y_{p_1,\ldots,p_N} x_{N,p_1,\ldots,p_N} \\ y_{p_1,\ldots,p_N} \\ \mathbf{0}_{Q_2\ldots Q_N(q_1-1),1} \\ -u^{[g]}_{p_1,\ldots,p_N} \\ \mathbf{0}_{Q_2\ldots Q_N(Q_1-q_1),1} \\ -u^{[g]}_{p_1,\ldots,p_N} \\ \mathbf{0}_{Q_1Q_3\ldots Q_N(q_2-1),1} \\ -u^{[g]}_{p_1,\ldots,p_N} \\ \mathbf{0}_{Q_1Q_3\ldots Q_N(Q_2-q_2),1} \\ \vdots \\ -u^{[g]}_{p_1,\ldots,p_N} \\ \vdots \\ \mathbf{0}_{Q_1Q_2\ldots Q_{N-1}(Q_N-q_N),1} \end{bmatrix}^T \in \mathbb{R}^{Q_1\ldots Q_N\times(N+Q_1\ldots Q_N+1)}, \tag{5.45}$$

for $q_1 = 1,\ldots,Q_1$, ..., $q_N = 1,\ldots,Q_N$. Then, a single row of $\mathbf{R}^{[g](2)}$ is given by:

$$[\mathbf{R}^{[g](2)}]_{q_1,\ldots,q_N,:} = \begin{bmatrix} \mathbf{0}_{N+1,1} \\ \mathbf{0}_{Q_2\ldots Q_N(q_1-1),1} \\ -u^{[n]}_{p_1,\ldots,p_N} \\ \mathbf{0}_{Q_2\ldots Q_N(Q_1-q_1),1} \\ -u^{[n]}_{p_1,\ldots,p_N} \\ \mathbf{0}_{Q_1Q_3\ldots Q_N(q_2-1),1} \\ -u^{[n]}_{p_1,\ldots,p_N} \\ \mathbf{0}_{Q_1Q_3\ldots Q_N(Q_2-q_2),1} \\ \vdots \\ -u^{[n]}_{p_1,\ldots,p_N} \\ \vdots \\ \mathbf{0}_{Q_1Q_2\ldots Q_{N-1}(Q_N-q_N),1} \end{bmatrix}^T \in \mathbb{R}^{Q_1\ldots Q_N\times(N+Q_1\ldots Q_N+1)}, \tag{5.46}$$

for $q_1 = 1,\ldots,Q_1$, ..., $q_N = 1,\ldots,Q_N$.

Algorithm 3: SVM-MTS Estimation Algorithm

Input: ε, C, $x_{n,p_1,p_2,...,p_N}$ and $y_{p_1,p_2,...,p_N}$, for $p_n = 1,...,P_n$ and $n = 1,...,N$.
Outputs: $\hat{\mathbf{w}}, \hat{b}, \hat{\mathscr{E}}$
Initialization: $\beta = 0$, $i = 0$, $\hat{\mathbf{A}}_0^{(1)}$, $\hat{\mathbf{A}}_0^{(2)}, \ldots, \hat{\mathbf{A}}_0^{(N)}$ and $\hat{\mathscr{G}}_0$ are random
while $\beta = 0$ **do**
  $i = i + 1$
  for $n = 1 : N$ **do**
    Construct $\mathbf{P}$, $\mathbf{q}$, $\mathbf{R}$ and $\mathbf{s}$ from (5.38)-(5.40) using $\hat{\mathbf{A}}_i^{(1)}$,..., $\hat{\mathbf{A}}_i^{(n-1)}$, $\hat{\mathbf{A}}_{i-1}^{(n+1)}$, ...,$\hat{\mathbf{A}}_{i-1}^{(N)}$ and $\hat{\mathscr{G}}_{i-1}$.
    Using any QP method, find $\hat{\mathbf{w}}_i, \hat{b}_i, \hat{\mathbf{A}}_i^{(n)}$
  end
  Construct $\mathbf{P}^{[g]}$, $\mathbf{q}^{[g]}$, $\mathbf{R}^{[g]}$ and $\mathbf{s}^{[g]}$ from (5.42)-(5.40) using $\hat{\mathbf{A}}_i^{(1)}$,...,$\hat{\mathbf{A}}_i^{(N)}$.
  Using any QP method, find $\hat{\mathbf{w}}_i, \hat{b}_i, \hat{\mathscr{G}}_i$
  Build $\hat{\mathscr{E}}_i$ using $\hat{\mathbf{A}}_i^{(1)}, \ldots, \hat{\mathbf{A}}_i^{(N)}$ and $\hat{\mathscr{G}}_i$ using (5.5)
  Unfold $\hat{\mathscr{E}}_i$ into $\hat{\mathbf{E}}_i^{[1]}$ using (2.18)
  if $|e_i| < \varepsilon$ **then**
    $\beta = 1$
    $\hat{\mathbf{w}} = \hat{\mathbf{w}}_i, \hat{b} = \hat{b}_i, \hat{\mathscr{E}} = \hat{\mathscr{E}}_i$
  end
end

After expressing the problems (5.27)-(5.29) and (5.32)-(5.34) as QP, it can be solved using any QP method.

5.4.3 Estimation Algorithm

The estimation algorithm of the SVM-MTS is shown in Algorithm 3. The algorithm is composed of N+1 stages. First the estimation of the factor matrices is done, and then, after, the estimation of the core tensor is performed. Each iteration is composed of N+1 stages, in each stage, one factor matrix of the Tucker decomposition of the slack tensor $\mathscr{E}$ is estimated using the previous estimation of the other factor matrices. When the N stages end, the (N+1)-th stage begins, where the core tensor is estimated. After that, the process starts over and these steps repeat runs until convergence.

These estimations are carried out using any QP method with the formulation presented given by (5.38)-(5.21). In each stage of each iteration, the weight vector $\mathbf{w}$ and the bias scalar b are also estimated.

As Algorithm 3 shows, the matrices $\mathbf{P}$, $\mathbf{q}$, $\mathbf{R}$ and $\mathbf{s}$ are used to estimate the factor matrices $\hat{\mathbf{A}}_i^{(1)}, \ldots, \hat{\mathbf{A}}_i^{(N)}$ similarly as in Algorithm 2, however, one more step is needed, hence

$\mathbf{P}^{(g)}$, $\mathbf{q}^{(g)}$, $\mathbf{R}^{(g)}$ and $\mathbf{s}^{(g)}$ are used to estimate the core tensor $\hat{\mathscr{G}}_i$. Next, Algorithm 3 proceeds similar as in Algorithm 2, with tensor $\hat{\xi}$ being mounted and unfolded at the end of each iteration.

Convergence is obtained when $e_i < \varepsilon$. The error e_i of the reconstructed tensor $\hat{\xi}_i$ at the $i-th$ iteration is given by (5.22), where ε is a small scalar constant. If the error e_i is smaller than the threshold ε, the binary flag variable β finish the estimations procedures. After the algorithm ends, the last obtained values of $\hat{\mathbf{w}}$ and $\hat{b}$ are saved to be used as final parameters to build the SVM model with the discriminant hyperplane:

$$f(\mathbf{x}(p_1, p_2..., p_N)) = \hat{\mathbf{w}}^T \mathbf{x}(p_1, p_2, ..., p_N) + \hat{b}. \tag{5.47}$$

The same is done for the SVM-MPS case. Also, a formulation for the SVR with multidimensional sampling is presented in Appendix A. Unfortunately, due to the short time, this approach is not tested in chis Chapter.

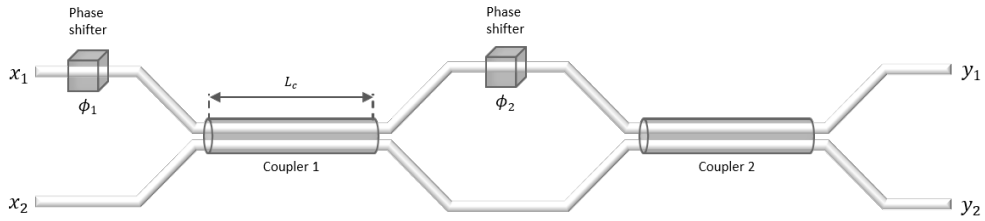
5.5 Photonic Database Description

All-optical processing is essential in systems and networks that avoid optoelectronic conversions and need high-speed data rates. The idea of designing logic gates based on optic devices, such as optical couplers, resonators and interferometers is intendend to solve this problem.

One of the devices that has been exploited for this application is a fiber-optic interferometer known as MZI. Numerical studies have used the solution of the nonlinear Schrodinger equation to design MZIs capable of obtaining logic functions (KUMAR *et al.*, 2014; ARAÚJO *et al.*, 2015). Hence, optical devices such as the MZI have been used in many computer and engineering applications, such as optical sensors, optical modulators and others (SOUZA *et al.*, 2018; GAYEN *et al.*, 2012).

Optical logic gates are a very important part in the development of all-optical communication and optical signal processing networks (ALIPOUR-BANAEI *et al.*, 2017). Indeed, they represent the basic building block of optical devices and networks, where all-optical processing is, in general, essential in systems and networks that want to avoid optoelectronic conversions and exploit high-speed data reception and transmission. Several researches have been done for designing logic gates based on optic devices, such as optical couplers, resonators, interferometers etc.

Figure 24 – Mach-Zehnder interferometer block scheme.



Source: Author.

The database used to test and validate the SVM-MPS and SVM-MTS was obtained from a Mach-Zehnder interferometer, which works performing power switching by associating two cascaded optical couplers, as illustrated in Fig 24. The first coupler splits the input optical signal into two parts, which are recombined when passing through the second coupler. The signals obtained at the MZI outputs are the result of the interference of the propagating signals in both arms of the interferometer. In this way, the output powers depend on the relative phase shifts suffered by the signals before each coupling procedure.

The phase shift is a very important step and may be implemented through several ways. Once the signals are guided physically separated in a MZI, a phase shift circuit can be implemented by means of fibers with different doping or simply with a difference between the lengths of the fibers. In the scheme of Fig. 24, two phase shifters are considered along one of the arms, implementing the relative phase shifts ϕ_1 and ϕ_2 in order to control the levels of interference in the respective couplers.

Hence, by considering a pulse amplitude modulation (PAM) of the signals at the input of the MZI, all the input combinations (00, 01, 10 and 11) were tested. The phase deviations of the MZI, ϕ_1 and ϕ_2, are altered in order to produce two outputs. In Table 20, parameters of the used MZI are shown. The number of tested values of ϕ_1 and ϕ_2, in the range 0 to 2π, is set to $M = 40$ and the number of tested values N for each input is set to 2 (0 and 1). Thus, for each pair (ϕ_1, ϕ_2), the MZI output associated with each input combination is obtained.

Therefore, the optic database corresponding to the inputs and phase deviation of the MZI has 6400 samples ($2 \times 2 \times 40 \times 40$), resulting in a matrix with dimensions 4×6400. The four columns of the matrix denote, in order, to the following attributes: I_1, I_2, ϕ_1 and ϕ_2, where I_1 and I_2 are the inputs of the interferometer, with values of 0 and 1 and ϕ_1 and ϕ_2 the phase deviations.

Moreover, we have y_1 and y_2, which are the two different outputs of the MZI, where these outputs are used as logic levels (high or low), to train a SVM model for classification of

Table 20 – MZI setup parameters

Description	Value
Modulation	PAM
Input power	4.56 kW
Wavelength	1.55 μm
Coupler length	1.8 cm
Attenuation	0
Input I_1	0,1
Input I_2	0,1
Output y_1	ranged form -51.07 to 79.52
Output y_2	ranged form -99.01 to 132.16
ϕ_1, ϕ_2	ranged from 0 to 2π
Number M of tested values of ϕ_1, ϕ_2	40
Number N of tested values of I_1, I_2	2

Source: Author.

optic data. However, these outputs shows values ranging from -99 to 132, therefore, these output values must undergo a binarization process before the classification step. The binarization sets the output as a logic 1 if the intensity value of the output is higher than zero, whereas the logic zero is set if the intensity values are lower than zero.

To summarize, the used data set consists in I_1, I_2, ϕ_1 and ϕ_2 as attributes and the binarization of y_2 as the output class tag, which were the output values chosen to be used in the classification model and provided the best results. After the binarization of y_2, the class tags were arranged as 3200 ones (Class 1) and 3200 zeroes (Class 0). Hence, for this application, it is considered a tensor sampling approach for the samples, where a fourth order tensor is constructed with the samples for attributes I_1, I_2, ϕ_1 and ϕ_2. Therefore, the constructed tensor has dimensions $2 \times 2 \times 40 \times 40$. The next step is to train the proposed modified SVM model with y_2 as class tag.

5.6 Results

This section presents the obtained results of this chapter. To validate the model, the K-fold cross-validation method was used, with K = 10. The constant C values were fixed at C = 100. Also, the results were averaged 100 times in order to mitigate fluctuation and the convergence parameter was set as $\varepsilon = 10^{-6}$.

It was considered the sampling modes as $p_1 = 1,...,P_1$, $p_2 = 1,...,P_2$, $p_3 = 1,...,P_3$ and $p_4 = 1,...,P_4$, with these indexes corresponding to I_1, I_2, ϕ_1 and ϕ_2 values. Hence, P_1 = 2, P_2 = 2, P_3 = 40 and P_4 = 40. In addition, the core tensor of the SVM-MTS has four rank values, Q_1, Q_2, Q_3 and Q_4. Data is structured as a tensor of dimensions $2 \times 2 \times 40 \times 40$ and fed into

Table 21 – Accuracy and execution time results for the proposed modifications and the SVM for various rank values.

Ranks	Classifier	Accuracy	Execution Time (s)
Q=5	SVM-MPS	79.2%	**271.2**
Q=10	SVM-MPS	80.1%	301.1
Q=15	SVM-MPS	**83.2**%	344.6
Q=20	SVM-MPS	82.9%	369.7
Q_1=Q_2=2, Q_3=Q_4=5	SVM-MTS	78.5%	**299.4**
Q_1=Q_2=2, Q_3=Q_4=10	SVM-MTS	81.1%	322.7
Q_1=Q_2=2, Q_3=Q_4=15	SVM-MTS	85.3%	366.0
Q_1=Q_2=2, Q_3=Q_4=20	SVM-MTS	**85.6**%	417.5
-	SVM	71.7%	**155.1**

Source: Author.

both SVM-MPS and SVM-MTS, whereas for the conventional SVM with linear kernel, the data is matricized with dimensions 4×6400, breaking the multilinear sampling structure proposed. The performance of the proposed techniques is initially tested against the simple SVM technique and later with other classifiers of the literature.

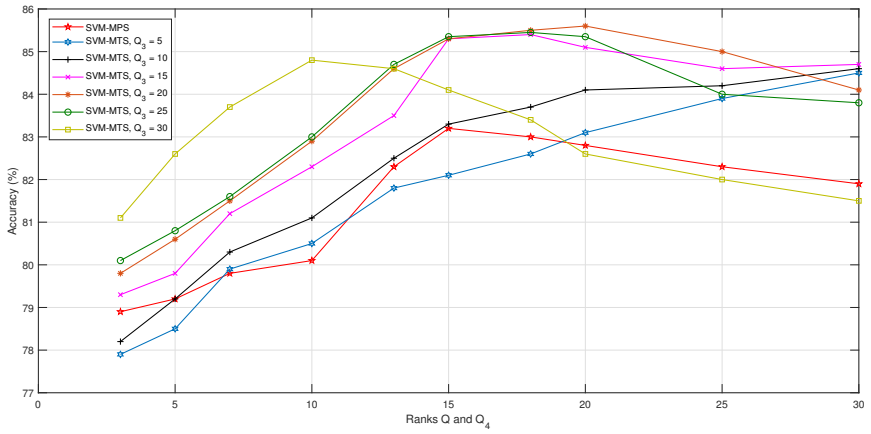
The results are presented in the form of accuracy in %, which is defined as the number of correctly classified samples over the total number of samples. Also, the execution time, in seconds, of the classifiers, is shown in order to measure computational cost. The processing times were obtained with an 9^{th} generation Intel Core i3 processor, running MATLAB version 2017b. The functions used to model the SVM classifier are *fitcsvm*, *predict*, and *quadrprog* was used for the SVM-MPS and SVM-MTS. The *quadrprog* uses three possible algorithms: the Goldfarb and Idnani dual algorithm, which is a numerically stable dual method for solving strictly convex quadratic programs (GOLDFARB; IDNANI, 1983), the Newton method of (COLEMAN; LI, 1996) and a projection method, similar to the one described in (GILL *et al.*, 2019).

In the sequel, the impact of ranks Q, Q_1, Q_2, Q_3 and Q_4 on the accuracy is analyzed. Later, the FLOPS and the number of iterations needed for convergence of the proposed techniques are analyzed.

5.6.1 Rank Impact on Accuracy and Execution Time

The first accuracy results are depicted in Table 21, which shows the impact of the rank values on classification rates, where the rank of the PARAFAC decomposition is tested for the following values, $Q = 5$, $Q = 10$, $Q = 15$ and $Q = 20$, whereas the Tucker ranks were chosen as $Q_1 = Q_2 = 2$, $Q_3 = Q_4 = 5$, $Q_3 = Q_4 = 10$, $Q_3 = Q_4 = 15$ and $Q_3 = Q_4 = 20$, which were the values that provided good accuracy on preliminary tests. The execution time, in seconds, of each scenario is also shown.

Figure 25 – Accuracy of the proposed SVM-MPS and SVM-MTS when varying ranks Q and Q_4, with $Q_1 = Q_2 = 2$.



Source: Author.

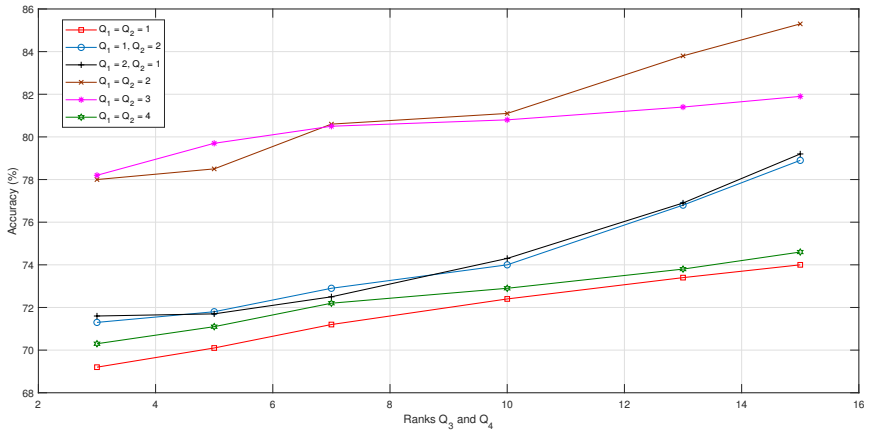
As Table 21 shows, both the proposed classifiers achieved the highest accuracy in comparison with the conventional SVM, which corroborates the fact that multilinear sampling structures can bring advantages in classification. In terms of rank, the PARAFAC modification performed better when its rank was increased, with the same behavior for the Tucker case. This is due the fact that with a higher rank, better tensor subspace representation is achieved, which facilitates the classification problem. However, when the rank Q value goes above 15, the accuracy of the SVM-MPS diminished, whereas for the SVM-MTS ranks $Q_3 = Q_4 = 20$ achieved the best accuracy value of 85.6%.

The higher accuracy was achieved by the proposed modifications, however, at the expense of longer execution times, which is explained by the fact that both SVM-MPS and SVM-MTS approaches need more steps to estimate all the factor matrices and the core tensor for the Tucker case. These extra steps add more time, making the proposed modifications to run slower in comparison to the conventional SVM, which runs almost two times faster.

In Figure 25, the accuracy of the proposed modifications versus the ranks Q (for the SVM-MPS) and Q_1, Q_2, Q_3 and Q_4 (for the SVM-MTS) variations is depicted. This result shows a better picture of the rank impact on accuracy than Table 21. As can be seen in Fig. 25, the accuracy of both SVM-MPS and SVM-MTS increases as the ranks Q and Q_4 increase, however, for values of Q higher than 15, the accuracy of the SVM-MPS starts to drop, which is the same behavior exhibited in Table 21. As for the SVM-MTS, the best case scenario is when $Q_3 = Q_4 = 20$, which achieves the higher accuracy, such as in Table 21.

Moreover, Fig. 25 shows that, with the rank Q_3 fixed, the accuracy of the SVM-MTS

Figure 26 – Accuracy of the proposed SVM-MTS when varying ranks Q_3 and Q_4, for fixed values of Q_1 and $Q_2 = 2$.



Source: Author.

tends to increase with Q_4 until a certain threshold, then begins to drop. For instance, when $Q_3 = 20$, the accuracy of the SVM-MTS starts to decrease when $Q_4 = 25$, then drops again when $Q_4 = 30$. This behavior of the algorithms in Figure 25 can be explained as follows. As the rank value start to rise, it start capturing most of the input data variation until a certain threshold, which is $Q = 15$ for the SVM-MPS and $Q_3 = Q_4 = 20$. When the rank value surpass these thresholds, redundancy is added, making the classification more difficult, with lower accuracy.

In addition, in Figure. 26 the impact of ranks Q_1 and Q_2 on the accuracy of the SVM-MTS is shown. It can be seen that the best accuracy is obtained when $Q_1 = Q_2 = 2$, whereas the other combinations achieved less success rates, with the worst case being with $Q_1 = Q_2 = 1$. When $Q_1 = Q_2 = 3$, the accuracy is higher than the one obtained with $Q_1 = Q_2 = 2$, but only for lower values of Q_3 and Q_4. The result obtained with $Q_1 = Q_2 = 4$ is the second worse, with accuracy only higher than when $Q_1 = Q_2 = 1$.

The final result of this subsection, presented in Table 22, depicts accuracy and execution times for the best case scenario of both SVM-MPS ($Q = 15$) and SVM-MTS ($Q_1 = Q_2 = 2$, $Q_3 = Q_4 = 20$) against other classifiers of the literature: Naive-Bayes (YANG, 2018), Logistic Regression (NICK; CAMPBELL, 2007), k-NN (JIANG *et al.*, 2007) and ANN (LIAO; WEN, 2007). The k-NN was performed with $k = 3$ and the tested ANN implements forward propagation with two fully connected layers, Rectified linear unit (ReLU) activation functions and a softmax function to the final fully connected layer.

As Table 22 shows, the proposed techniques surpassed, in terms of accuracy, well-known classifiers, such as the Naive-Bayes and ANN, thus showing the efficacy of the SVM-MPS

Table 22 – Accuracy and execution time results for the proposed modifications versus other classifiers.

Classifier	Accuracy	Execution Time (s)
SVM-MPS	**83.2**%	344.6
SVM-MTS	**85.6**%	417.5
SVM	71.7%	155.1
Naive-Bayes	70.2%	144.3
Logistic Regression	68.1%	132.7
k-NN	65.3%	**111.9**
ANN	77.6%	296.6

Source: Author.

Table 23 – Time-complexity of the proposed and tested techniques, in big-O notation.

SVM-MPS	$\mathcal{O}(QP^2[P_1+P_2+P_3+P_4]N_{it})$
SVM-MTS	$\mathcal{O}(P^2[P_1Q_1+P_2Q_2+P_3Q_3+P_4Q_4+Q_1Q_2Q_3Q_4]N_{it})$
SVM	$\mathcal{O}(DP^2)$
Naive-Bayes	$\mathcal{O}(DP)$
Logistic Regression	$\mathcal{O}(2NDP)$
***k*-NN**	$\mathcal{O}(kDP)$
ANN	$\mathcal{O}(D^4P)$

Source: Author.

and SVM-MTS and also the adopted multilinear sampling structure. As already explained, the multilinear sampling approach permits the SVM-MPS and SVM-MTS to take advantage of the structure of the data, thus allowing a multidimensional perspective, clearly improving the accuracy, as Tables 21 and 22 showed.

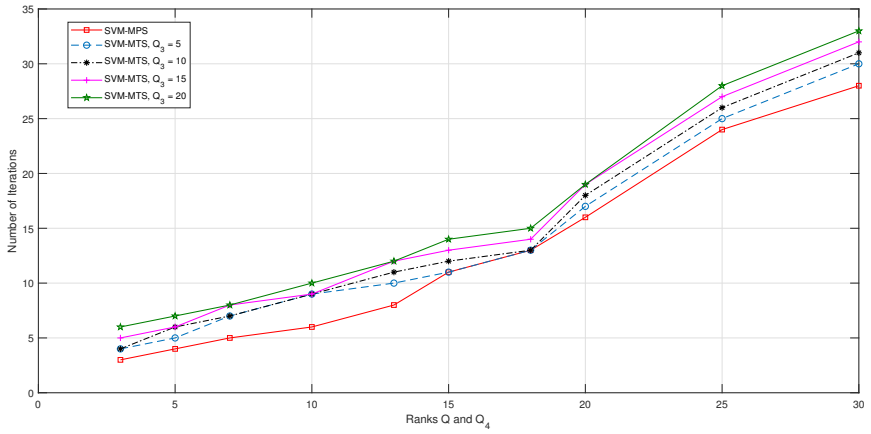
Furthermore, Table 22 shows that although the proposed techniques demand more running times in comparison to the literature alternatives, they also achieve higher success rates in classification, which is a clear trade-off between execution time and accuracy. Also, the only technique that achieved a classification rate close to the proposed algorithms was the ANN, which also has high execution time in comparison to the other classifiers.

5.6.2 Computational Cost Analysis

The computational cost of the proposed SVM modifications is analyzed, in terms of time-complexity, FLOP count and number of iterations required for convergence. Then first, Table 23 shows the time-complexity for tested techniques in this chapter: SVM-MPS, SVM-MTS, conventional linear SVM, Naive-Bayes, Logistic Regression, k-NN and the ANN, where D denotes the total dimension of the array used with the technique, as defined in Chapter 3.

It can be seen from Table 23 that the proposed SVM-MPS and SVM-MTS have high

Figure 27 – Iterations of the proposed SVM-MPS and SVM-MTS when varying ranks Q and Q_4, for fixed values of Q_1, Q_2 and Q_3.



Source: Author.

time-complexity in comparison to techniques such as the conventional SVM, Naive-Bayes, Logistic Regression and k-NN, which explain the higher execution times shown in Table 22. Moreover, the ANN technique showed high time-complexity, which corroborates with its execution time being closer to the SVM-MPS and SVM-MTS ones.

Next, in Figure 27 we have the number iterations needed for convergence, of the SVM-MPS and SVM-MTS estimations, described in Algorithms 2 and 3, in function of the ranks Q and Q_4. As Fig. 27 shows, the increase in Q and Q_4 demands more iterations for the estimations to be completed, which is expected since the the ranks Q, Q_3 and Q_4 impact directly on array sizes and in the time-complexity of the algorithms.

In addition, the best case scenario of accuracy for the SVM-MPS, with $Q = 15$, achieved convergence with around 10 iterations, therefore, the SVM-MPS needed a small number of iterations, resulting in lower execution times, to get its best accuracy result. On the other hand, the SVM-MTS best accuracy scenario was achieved with $Q_1 = Q_2 = 2$ and $Q_3 = Q_4 = 20$, which results in less than 20 iterations, therefore, the best case of the SVM-MPS was obtained with around 10 iterations whereas the SVM-MTS achieved it with double this number. Hence, from Table 27 it can be concluded that the SVM-MTS achieves higher accuracy than the SVM-MPS at the cost of more iterations, when estimating the factor matrices and core tensor, and thus, higher execution times, corroborating even more to the results obtained in Table 22.

The last result, depicted in Table 24, shows the FLOPS of the tested classification techniques, with two rank configurations for the SVM-MPS ($Q = 10$ and $Q = 15$) and SVM-MTS ($Q_1 = Q_2 = 2$, $Q_3 = Q_4 = 15$ and $Q_1 = Q_2 = 2$, $Q_3 = Q_4 = 20$). As can be seen, both SVM-MPS and

Table 24 – Floating Point Operations per Second (FLOPS) counts for the tested classifiers, with two different rank configurations.

Classifier	**FLOPS**
SVM-MPS ($Q = 10$)	3.3×10^7
SVM-MTS ($Q_1 = Q_2 = 2$, $Q_3 = Q_4 = 15$)	4.2×10^7
SVM-MPS ($Q = 15$)	4.7×10^7
SVM-MTS ($Q_1 = Q_2 = 2$, $Q_3 = Q_4 = 20$)	7.5×10^7
SVM	2.5×10^6
Naive-Bayes	1.2×10^4
Logistic Regression	1.7×10^5
k-NN	1.9×10^4
ANN	1.3×10^7

Source: Author.

SVM-MTS demanded similar FLOPS, in the 10^7 range, whereas the other classifiers demanded much less FLOPS, in the $10^4 - 10^6$ range, with the conventional SVM requiring 2.5×10^6. Such high FLOP count of the proposed techniques when comparing to the others is explained by the high time-complexity of the techniques, which were shown earlier in Table 23. Also, depicted in Table 24, an increase in the rank values also increases the FLOPS of both SVM-MPS and SVM-MTS.

5.7 Conclusions

In this chapter, the concept of multilinear sampling was proposed, aiming the proposition of classification algorithms that exploit a multidimensional structure of the samples. Such approach is based on tensor decompositions PARAFAC and Tucker, where the goal is to adapt the primal solution of the SVM to accommodate a multilinear sampling structure of the input data.

The proposed modifications of the SVM are used to classify the binary outputs obtained from a MZI. To accomplish that, a four-way array is considered to store four input sample sets, corresponding to the two interferometer inputs and two phase deviation variables.

The results showed better performance of the proposed classifiers in comparison to other techniques of the literature, which is expected as these techniques does not use the multilinear sampling approach, losing accuracy. In addition, computational cost and execution time tests were carried out, showing that the proposed SVM-MPS and SVM-MTS showed slight higher complexity in comparison to the other tested techniques. Moreover, convergence testes showed the proposed algorithms achieve convergence and finish estimations with ow iterations.

6 CONCLUSIONS

In this thesis, theoretical and applied contributions on tensor learning methods were presented. Our contributions have addressed the following main research axes:

- Tensor learning;
- Tensor decompositions;
- Multilinear dimensionality reduction;
- Feature transformation;
- Seismic event calssification;
- Multilinear sampling;
- Photonic data classification.

More specifically, by chapter order, a fully tensorial framework for seismic event classification was proposed in Chapter 3, which jointly performs feature extraction using the MDFT, dimensionality reduction using the MPCA, and classification with the SPM and the STuM. In this chapter, the tensorial framework, which can be fed with multidimensional data, was used for classifying volcano-seismic signals into five different classes. The database used in this work consists in three-dimensional data samples recorded during a period of great activity of the Ubinas volcano, Peru, in 2009. The tensor structure of the patterns, organized as *stations* $\times$ *channels* $\times$ *features*, is built by exploring the use of multiple multichannel triaxial sensors, operating simultaneously in two seismic stations. The framework itself, is an original contribution, as no other work in the literature has proposed similar approach for seismic event classification.

The results showed the very significant gain in performance provided by the tensorial classifiers, as well as by the MPCA, when compared with their vector-based counterparts. The best result was obtained with the STuM classifier along with the MPCA-DR. The best accuracy provided by the tensor-based configurations is due to the fact they preserve the multidimensional structure of the data, avoiding the drawbacks of tensor vectorization.

Then, in Chapter 4, a new dimensionality reduction technique called LC-MDR was proposed, based on a new tensor decomposition, the EONPD, which successfully reduced the correlation and dimensionality of seismic data, improving classification rates. Also, a generalization of the EONPD, that assumes the nesting of PARAFAC tensors of generic orders, called HONPD is formulated.

In this chapter, the presented method optimizes a cost function that takes into account the data correlation, generating variables with much less correlation than the MPCA. In particular,

the LC-MDR fits the input data correlation into the EONPD. Moreover, the LC-MDR is based on the ALS for estimating the factor matrices. A complete description of the EONPD is given, with both analytical and recursive expressions being derived. In addition, the HONPD formulation is also presented.

The results showed very significant gains in accuracy of the LC-MDR over the concurrent PCA and MPCA, in seismic event classification. The best accuracy (90.3%) was achieved with the 3D LC-MDR coupled with the STuM classifier, using dimensionality reduction.

In addition, the results showed the LC-MDR provides output data with much less correlation than the MPCA, which explains the better accuracy of the LC-MDR. Moreover, the LC-MDR showed a faster execution time in comparison with the MPCA. Besides, the proposed technique also provides smaller execution times than the PCA if dimensionality reduction is carried out.

And finally, in Chapter 5, a multilinear sampling approach for tensor learning and data structuring is proposed, which is employed within the SVM technique, joining the concept of multilinear sampling structures and tensor decompositions. This yielded two modifications for the SVM, denoted by SVM-MPS and SVM-MTS, which were both used in photonic data classification.

The proposed modification of the SVM is used to classify the photonic output obtained from a MZI. To accomplish that, a four-way array is considered to store four input sample sets, corresponding to the two interferometer inputs and two phase deviation variables, with each mode of the tensor storing a sample set.

Moreover, results showed better accuracy of the proposed SVM modifications in classifying the MZI output, comparing to the standard SVM. The proposed multidimensional sampling structure proved being effective in classification when compared to the conventional vectorization processes. Also computational cost and running times were taken as parameters to validate the proposed approach.

This thesis contributions may be extended by considering the following. The fully tensorial framework of Chapter 3 can be expanded by using other multidimensional feature extraction techniques, such as the multidimensional EMD, multidimensional ICA, and, other multilinear dimensionality reduction techniques, such as the MDA. Other tensor-based classifiers could be tested, such as the STTM, or kernelized versions of the STMs.

Regarding the contents of Chapter 4, the LC-MDR can be tested, instead of seismic

events, with other kinds of multidimensional data and other applications. The seismic data was arranged in 3D tensors, however, the LC-MDR is capable of performing dimensionality reduction and feature transformation even in N-th order arrays, thus any multidimensional data can be processed by the LC-MDR.

As for the contributions of Chapter 5, the multilinear sampling approach can be adopted for other kinds of data besides photonic. For instance, gait data, which comprises of multiple movement frames, can be arranged in multidimensional arrays and therefore exploit the proposed multilinear sampling approach. Regarding the proposed SVM modifications, the SVM-MPS and SVM-MTS, they can be used with multiple multidimensional dataset, thus being very versatile classifiers.

Additionally, the Dual formulation of the SVM and SVR can be explored in the context of multilinear sampling and tested against the primal solution presented in this thesis. Finally, the multilinear sampling SVR formulation presented Appendix A may be tested in regression applications, exploiting the multilinear sampling approach.

BIBLIOGRAPHY

ALIPOUR-BANAEI, H.; SERAJMOHAMMADI, S.; MEHDIZADEH, F. All optical nand gate based on nonlinear photonic crystal ring resonators. **Optik**, Elsevier, v. 130, p. 1214–1221, 2017.

ALMEIDA, A. D. **Tensor modeling and signal processing for wireless communication systems**. Tese (Doutorado) — Université de Nice Sophia Antipolis, 2007.

ALMEIDA, A. L. F. de; FAVIER, G. Double khatri–rao space-time-frequency coding using semi-blind parafac based receiver. **IEEE Signal Processing Letters**, IEEE, v. 20, n. 5, p. 471–474, 2013.

ALMEIDA, A. L. F. de; FAVIER, G.; COSTA, J. da; MOTA, J. C. M. Overview of tensor decompositions with applications to communications. **Signals and images: advances and results in speech, estimation, compression, recognition, filtering, and processing**, CRC Press Boca Raton, FL, USA, p. 325–356, 2016.

ALPAYDIN, E. **Introduction to machine learning**. [S.l.]: MIT press, 2020.

ARAÚJO, A.; OLIVEIRA, A.; MARTINS, F.; COELHO, A.; FRAGA, W.; NASCIMENTO, J. Two all-optical logic gates in a single photonic interferometer. **Optics Communications**, Elsevier, v. 355, p. 485–491, 2015.

BEGG, R. K.; PALANISWAMI, M.; OWEN, B. Support vector machines for automated gait classification. **IEEE transactions on Biomedical Engineering**, IEEE, v. 52, n. 5, p. 828–838, 2005.

BELARBI, M. A.; MAHMOUDI, S.; BELALEM, G. Pca as dimensionality reduction for large-scale image retrieval systems. **International Journal of Ambient Computing and Intelligence (IJACI)**, IGI Global, v. 8, n. 4, p. 45–58, 2017.

BERGE, J. M. F. T.; SIDIROPOULOS, N. D. On uniqueness in candecomp/parafac. **Psychometrika**, Springer, v. 67, n. 3, p. 399–409, 2002.

BRO, R. Parafac. tutorial and applications. **Chemometrics and intelligent laboratory systems**, Elsevier, v. 38, n. 2, p. 149–171, 1997.

BURGES, C. J. A tutorial on support vector machines for pattern recognition. **Data mining and knowledge discovery**, Springer, v. 2, n. 2, p. 121–167, 1998.

CAI, D.; HE, X.; WEN, J.-R.; HAN, J.; MA, W.-Y. **Support tensor machines for text categorization**. [S.l.], 2006.

CALVI, G. G.; LUCIC, V.; MANDIC, D. P. Support tensor machine for financial forecasting. In: IEEE. **ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.], 2019. p. 8152–8156.

CARLEO, G.; CIRAC, I.; CRANMER, K.; DAUDET, L.; SCHULD, M.; TISHBY, N.; VOGT-MARANTO, L.; ZDEBOROVÁ, L. Machine learning and the physical sciences. **Reviews of Modern Physics**, APS, v. 91, n. 4, p. 045002, 2019.

CARROLL, J. D.; CHANG, J.-J. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. **Psychometrika**, Springer, v. 35, n. 3, p. 283–319, 1970.

CERVANTES, J.; GARCIA-LAMONT, F.; RODRÍGUEZ-MAZAHUA, L.; LOPEZ, A. A comprehensive survey on support vector machine classification: Applications, challenges and trends. **Neurocomputing**, Elsevier, v. 408, p. 189–215, 2020.

CHEN, C.; BATSELIER, K.; KO, C.-Y.; WONG, N. A support tensor train machine. In: IEEE. **2019 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2019. p. 1–8.

CHOI, S.; CICHOCKI, A.; PARK, H.-M.; LEE, S.-Y. Blind source separation and independent component analysis: A review. **Neural Information Processing-Letters and Reviews**, v. 6, n. 1, p. 1–57, 2005.

CHOUET, B. A. Long-period volcano seismicity: its source and use in eruption forecasting. **Nature**, Nature Publishing Group, v. 380, n. 6572, p. 309–316, 1996.

CHOWDHARY, K. Natural language processing. **Fundamentals of artificial intelligence**, Springer, p. 603–649, 2020.

COLEMAN, T. F.; LI, Y. A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables. **SIAM Journal on Optimization**, SIAM, v. 6, n. 4, p. 1040–1058, 1996.

COMON, P. Tensors: a brief introduction. **IEEE Signal Processing Magazine**, v. 31, n. 3, p. 44–53, 2014.

COMON, P.; LUCIANI, X.; ALMEIDA, A. L. F. D. Tensor decompositions, alternating least squares and other tales. **Journal of Chemometrics: A Journal of the Chemometrics Society**, Wiley Online Library, v. 23, n. 7-8, p. 393–405, 2009.

CORREIA, D.; FRAGA, W. de; GUIMARÃES, G. *et al.* Obtaining optical logic gates–or, xor, and and logic functions using asymmetric mach-zehnder interferometer based on photonic crystal fiber. **Optics & Laser Technology**, Elsevier, v. 97, p. 370–378, 2017.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine learning**, Springer, v. 20, n. 3, p. 273–297, 1995.

CRESWELL, A.; WHITE, T.; DUMOULIN, V.; ARULKUMARAN, K.; SENGUPTA, B.; BHARATH, A. A. Generative adversarial networks: An overview. **IEEE Signal Processing Magazine**, IEEE, v. 35, n. 1, p. 53–65, 2018.

CURILEM, G.; VERGARA, J.; FUENTEALBA, G.; ACUÑA, G.; CHACÓN, M. Classification of seismic signals at villarrica volcano (chile) using neural networks and genetic algorithms. **Journal of volcanology and geothermal research**, Elsevier, v. 180, n. 1, p. 1–8, 2009.

CURILEM, M.; CANÁRIO, J. P.; FRANCO, L.; RIOS, R. A. Using cnn to classify spectrograms of seismic events from llaima volcano (chile). In: IEEE. **2018 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2018. p. 1–8.

DENG, L.; YU, D. Deep learning: methods and applications. **Foundations and trends in signal processing**, Now Publishers Inc. Hanover, MA, USA, v. 7, n. 3–4, p. 197–387, 2014.

EL-HASNONY, I. M.; BAKRY, H. M. E.; SALEH, A. A. Comparative study among data reduction techniques over classification accuracy. **International Journal of Computer Applications**, Foundation of Computer Science, v. 122, n. 2, 2015.

FAVIER, G.; FERNANDES, C. A. R.; ALMEIDA, A. L. F. de. Nested tucker tensor decomposition with application to mimo relay systems using tensor space–time coding (tstc). **Signal Processing**, Elsevier, v. 128, p. 318–331, 2016.

FOODY, G. M.; MATHUR, A. A relative evaluation of multiclass image classification by support vector machines. **IEEE Transactions on geoscience and remote sensing**, IEEE, v. 42, n. 6, p. 1335–1343, 2004.

FREITAS, W. d. C.; FAVIER, G.; ALMEIDA, A. L. F. de. Sequential closed-form semiblind receiver for space-time coded multihop relaying systems. **IEEE Signal Processing Letters**, IEEE, v. 24, n. 12, p. 1773–1777, 2017.

FU, X.; WANG, L. Data dimensionality reduction with application to simplifying rbf network structure and improving classification performance. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, IEEE, v. 33, n. 3, p. 399–409, 2003.

FU, Y.; HUANG, T. S. Image classification using correlation tensor analysis. **IEEE Transactions on Image Processing**, IEEE, v. 17, n. 2, p. 226–234, 2008.

GAYEN, D. K.; BHATTACHRYYA, A.; CHATTOPADHYAY, T.; ROY, J. N. Ultrafast all-optical half adder using quantum-dot semiconductor optical amplifier-based mach-zehnder interferometer. **Journal of Lightwave technology**, IEEE, v. 30, n. 21, p. 3387–3393, 2012.

GILL, P. E.; MURRAY, W.; WRIGHT, M. H. **Practical optimization**. [S.l.]: SIAM, 2019.

GOLDFARB, D.; IDNANI, A. A numerically stable dual method for solving strictly convex quadratic programs. **Mathematical programming**, Springer, v. 27, n. 1, p. 1–33, 1983.

GUESSOUM, A.; MERSEREAU, R. Fast algorithms for the multidimensional discrete fourier transform. **IEEE transactions on acoustics, speech, and signal processing**, IEEE, v. 34, n. 4, p. 937–943, 1986.

GUO, W.; KOTSIA, I.; PATRAS, I. Tensor learning for regression. **IEEE Transactions on Image Processing**, IEEE, v. 21, n. 2, p. 816–827, 2011.

GUO, X.; HUANG, X.; ZHANG, L.; ZHANG, L. Support tensor machine with local pixel neighborhood for hyperspectral image classification. In: IEEE. **2014 6th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)**. [S.l.], 2014. p. 1–4.

GUPTA, S.; KAR, A. K.; BAABDULLAH, A.; AL-KHOWAITER, W. A. Big data with cognitive computing: A review for the future. **International Journal of Information Management**, Elsevier, v. 42, p. 78–89, 2018.

HAN, X.; KWITT, R.; AYLWARD, S.; BAKAS, S.; MENZE, B.; ASTURIAS, A.; VESPA, P.; HORN, J. V.; NIETHAMMER, M. Brain extraction from normal and pathological images: A joint pca/image-reconstruction approach. **NeuroImage**, Elsevier, v. 176, p. 431–445, 2018.

HARDOON, D. R.; SZEDMAK, S.; SHAWE-TAYLOR, J. Canonical correlation analysis: An overview with application to learning methods. **Neural computation**, MIT Press, v. 16, n. 12, p. 2639–2664, 2004.

HARSHMAN, R. A. Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis. **UCLA Working Papers in Phonetics**, University of California at Los Angeles Los Angeles, CA, v. 16, p. 1–84, 1970.

HARSHMAN, R. A.; LUNDY, M. E. The parafac model for three-way factor analysis and multidimensional scaling. **Research methods for multimode data analysis**, New York: Praeger, v. 46, p. 122–215, 1984.

HART, P. E.; STORK, D. G.; DUDA, R. O. **Pattern classification**. [S.l.]: Wiley Hoboken, 2000.

HASHEMIZADEH, M.; LIU, M.; MILLER, J.; RABUSSEAU, G. Adaptive tensor learning with tensor networks. **arXiv preprint arXiv:2008.05437**, 2020.

HAVSKOV, J.; ALGUACIL, G. Correction for instrument response. In: **Instrumentation in Earthquake Seismology**. [S.l.]: Springer, 2016. p. 197–230.

HE, L.; LU, C.-T.; MA, G.; WANG, S.; SHEN, L.; PHILIP, S. Y.; RAGIN, A. B. Kernelized support tensor machines. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2017. p. 1442–1451.

HE, X.; CAI, D.; NIYOGI, P. Tensor subspace analysis. **Advances in neural information processing systems**, v. 18, 2005.

HITCHCOCK, F. L. The expression of a tensor or a polyadic as a sum of products. **Journal of Mathematics and Physics**, Wiley Online Library, v. 6, n. 1-4, p. 164–189, 1927.

HOLLAND, J. H. Genetic algorithms. **Scientific american**, JSTOR, v. 267, n. 1, p. 66–73, 1992.

INZA, L. A.; MARS, J. I.; MÉTAXIAN, J.-P.; O'BRIEN, G. S.; MACEDO, O. Seismo-volcano source localization with triaxial broad-band seismic array. **Geophysical Journal International**, Blackwell Publishing Ltd Oxford, UK, v. 187, n. 1, p. 371–384, 2011.

INZA, L. A.; MÉTAXIAN, J.-P.; MARS, J. I.; BEAN, C. J.; O'BRIEN, G. S.; MACEDO, O.; ZANDOMENEGHI, D. Analysis of dynamics of vulcanian activity of ubinas volcano, using multicomponent seismic antennas. **Journal of Volcanology and Geothermal Research**, Elsevier, v. 270, p. 35–52, 2014.

JI, Y.; WANG, Q.; LI, X.; LIU, J. A survey on tensor techniques and applications in machine learning. **IEEE Access**, IEEE, v. 7, p. 162950–162990, 2019.

JIA, J.; CAI, L.; LU, P.; LIU, X. Fingerprint matching based on weighting method and the svm. **Neurocomputing**, Elsevier, v. 70, n. 4-6, p. 849–858, 2007.

JIANG, L.; CAI, Z.; WANG, D.; JIANG, S. Survey of improving k-nearest-neighbor for classification. In: IEEE. **Fourth international conference on fuzzy systems and knowledge discovery (FSKD 2007)**. [S.l.], 2007. v. 1, p. 679–683.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. **Science**, American Association for the Advancement of Science, v. 349, n. 6245, p. 255–260, 2015.

KARMAKAR, D.; SARKAR, R.; DATTA, M. Spoofed replay attack detection by multidimensional fourier transform on facial micro-expression regions. **Signal Processing: Image Communication**, Elsevier, v. 93, p. 116164, 2021.

KHALID, S.; KHALIL, T.; NASREEN, S. A survey of feature selection and feature extraction techniques in machine learning. In: IEEE. **2014 science and information conference**. [S.l.], 2014. p. 372–378.

KOLDA, T. G.; BADER, B. W. Tensor decompositions and applications. **SIAM review**, SIAM, v. 51, n. 3, p. 455–500, 2009.

KORTSTRÖM, J.; USKI, M.; TIIRA, T. Automatic classification of seismic events within a regional seismograph network. **Computers & Geosciences**, Elsevier, v. 87, p. 22–30, 2016.

KOTSIA, I.; GUO, W.; PATRAS, I. Higher rank support tensor machines for visual recognition. **Pattern Recognition**, Elsevier, v. 45, n. 12, p. 4192–4203, 2012.

KOTSIA, I.; PATRAS, I. Support tucker machines. In: IEEE. **CVPR 2011**. [S.l.], 2011. p. 633–640.

KRUSKAL, J. B. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. **Linear algebra and its applications**, Elsevier, v. 18, n. 2, p. 95–138, 1977.

KULKARNI, S. R.; LUGOSI, G.; VENKATESH, S. S. Learning pattern classification-a survey. **IEEE Transactions on Information Theory**, IEEE, v. 44, n. 6, p. 2178–2206, 1998.

KUMAR, A.; KUMAR, S.; RAGHUWANSHI, S. K. Implementation of xor/xnor and and logic gates by using mach–zehnder interferometers. **Optik**, Elsevier, v. 125, n. 19, p. 5764–5767, 2014.

LARA, P. E. E.; FERNANDES, C. A. R.; INZA, A.; MARS, J. I.; MÉTAXIAN, J.-P.; MURA, M. D.; MALFANTE, M. Automatic multichannel volcano-seismic classification using machine learning and emd. **IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing**, IEEE, v. 13, p. 1322–1331, 2020.

LATHAUWER, L. D. A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. **SIAM journal on Matrix Analysis and Applications**, SIAM, v. 28, n. 3, p. 642–666, 2006.

LATHAUWER, L. D.; MOOR, B. D.; VANDEWALLE, J. Blind source separation by higher-order singular value decomposition. In: **Proc. EUSIPCO**. [S.l.: s.n.], 1994. v. 1, p. 175–178.

LATHAUWER, L. D.; MOOR, B. D.; VANDEWALLE, J. An introduction to independent component analysis. **Journal of chemometrics**, v. 14, n. 3, p. 123–149, 2000.

LATHAUWER, L. D.; MOOR, B. D.; VANDEWALLE, J. A multilinear singular value decomposition. **SIAM journal on Matrix Analysis and Applications**, SIAM, v. 21, n. 4, p. 1253–1278, 2000.

LATHAUWER, L. D.; VANDEWALLE, J. Dimensionality reduction in higher-order signal processing and rank-(r1, r2,..., rn) reduction in multilinear algebra. **Linear Algebra and its Applications**, Elsevier, v. 391, p. 31–55, 2004.

LATTIN, J. M.; CARROLL, J. D.; GREEN, P. E. **Analyzing multivariate data**. [S.l.]: Thomson Brooks/Cole Pacific Grove, CA, 2003.

LI, J.; ALLINSON, N.; TAO, D.; LI, X. Multitraining support vector machine for image retrieval. **IEEE Transactions on Image Processing**, IEEE, v. 15, n. 11, p. 3597–3601, 2006.

LIANG, G.; HONG, H.; XIE, W.; ZHENG, L. Combining convolutional neural network with recursive neural network for blood cell image classification. **IEEE access**, IEEE, v. 6, p. 36188–36197, 2018.

LIAO, S.-H.; WEN, C.-H. Artificial neural networks classification and clustering of methodologies and applications–literature analysis from 1995 to 2005. **Expert Systems with applications**, Elsevier, v. 32, n. 1, p. 1–11, 2007.

LIU, J.; MUSIALSKI, P.; WONKA, P.; YE, J. Tensor completion for estimating missing values in visual data. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 35, n. 1, p. 208–220, 2012.

LIU, Z.; ZHU, D.; RAJU, L.; CAI, W. Tackling photonic inverse design with machine learning. **Advanced Science**, Wiley Online Library, v. 8, n. 5, p. 2002923, 2021.

LU, H.; PLATANIOTIS, K. N.; VENETSANOPOULOS, A. N. Multilinear principal component analysis of tensor objects for recognition. In: IEEE. **18th International Conference on Pattern Recognition (ICPR'06)**. [S.l.], 2006. v. 2, p. 776–779.

LU, H.; PLATANIOTIS, K. N.; VENETSANOPOULOS, A. N. Mpca: Multilinear principal component analysis of tensor objects. **IEEE transactions on Neural Networks**, IEEE, v. 19, n. 1, p. 18–39, 2008.

LU, H.; PLATANIOTIS, K. N.; VENETSANOPOULOS, A. N. A survey of multilinear subspace learning for tensor data. **Pattern Recognition**, Elsevier, v. 44, n. 7, p. 1540–1551, 2011.

MA, L.; HU, Y.; ZHANG, Y. Support tucker machines based bubble defect detection of lithium-ion polymer cell sheets. **Engineering Letters**, v. 25, n. 1, 2017.

MA, W.; LIU, Z.; KUDYSHEV, Z. A.; BOLTASSEVA, A.; CAI, W.; LIU, Y. Deep learning for the design of photonic structures. **Nature Photonics**, Nature Publishing Group, v. 15, n. 2, p. 77–90, 2021.

MACEDO, O.; MÉTAXIAN, J.; TAIPE, E.; RAMOS, D.; INZA, L. Seismicity associated with the 2006–2008 eruption, ubinas volcano. **The VOLUME Project, edited by: Bean, CJ, Braiden, AK, Lokmer, I., Martini, F., O'Brien, GS**, v. 1, p. 262–270, 2009.

MALFANTE, M.; MURA, M. D.; MÉTAXIAN, J.-P.; MARS, J. I.; MACEDO, O.; INZA, A. Machine learning for volcano-seismic signals: Challenges and perspectives. **IEEE Signal Processing Magazine**, IEEE, v. 35, n. 2, p. 20–30, 2018.

MATHUR, A.; FOODY, G. M. Multiclass and binary svm classification: Implications for training and classification users. **IEEE Geoscience and remote sensing letters**, IEEE, v. 5, n. 2, p. 241–245, 2008.

MCNUTT, S. R. Volcanic seismology. **Annu. Rev. Earth Planet. Sci.**, Annual Reviews, v. 32, p. 461–491, 2005.

MENGU, D.; RAHMAN, M. S. S.; LUO, Y.; LI, J.; KULCE, O.; OZCAN, A. At the intersection of optics and deep learning: statistical inference, computing, and inverse design. **Advances in Optics and Photonics**, Optica Publishing Group, v. 14, n. 2, p. 209–290, 2022.

MITCHELL, T. M. Artificial neural networks. **Machine learning**, McGraw-Hill New York, v. 45, p. 81–127, 1997.

MYLES, A. J.; FEUDALE, R. N.; LIU, Y.; WOODY, N. A.; BROWN, S. D. An introduction to decision tree modeling. **Journal of Chemometrics: A Journal of the Chemometrics Society**, Wiley Online Library, v. 18, n. 6, p. 275–285, 2004.

NICK, T. G.; CAMPBELL, K. M. Logistic regression. **Topics in biostatistics**, Springer, p. 273–301, 2007.

NIU, G.; MA, Z. Tensor dimensionality reduction via mode product and hsic. **IET Image Processing**, Wiley Online Library, v. 15, n. 12, p. 2986–3002, 2021.

NOCEDAL, J.; WRIGHT, S. J. Quadratic programming. **Numerical optimization**, Springer, p. 448–492, 2006.

OSELEDETS, I.; TYRTYSHNIKOV, E. Tt-cross approximation for multidimensional arrays. **Linear Algebra and its Applications**, Elsevier, v. 432, n. 1, p. 70–88, 2010.

OSELEDETS, I. V. Tensor-train decomposition. **SIAM Journal on Scientific Computing**, SIAM, v. 33, n. 5, p. 2295–2317, 2011.

PANG, B.; NIJKAMP, E.; WU, Y. N. Deep learning with tensorflow: A review. **Journal of Educational and Behavioral Statistics**, SAGE Publications Sage CA: Los Angeles, CA, v. 45, n. 2, p. 227–248, 2020.

PARDO, M.; SBERVEGLIERI, G. Learning from data: A tutorial with emphasis on modern pattern recognition methods. **IEEE Sensors Journal**, IEEE, v. 2, n. 3, p. 203–217, 2002.

PAUCA, P.; PIPER, J.; PLEMMONS, R. J. Nonnegative matrix factorization for spectral data analysis. **Linear algebra and its applications**, Elsevier, v. 416, n. 1, p. 29–47, 2006.

PAULUS, C.; MARS, J. I. New multicomponent filters for geophysical data processing. **IEEE transactions on geoscience and remote sensing**, IEEE, v. 44, n. 8, p. 2260–2270, 2006.

PEIXOTO, A. A. T. **Detecção multiusuário baseada em tensores para sistemas de comunicação sem fio cooperativos**. Dissertação — Universidade Federal do Ceará, 2017.

PEIXOTO, A. A. T.; FERNANDES, C. A. R. Tensor-based multiuser detection in cooperative multirelay uplink. **Journal of Communication and Information Systems**, v. 34, n. 1, p. 36–49, 2019.

PEIXOTO, A. A. T.; FERNANDES, C. A. R.; LARA, P. E. E.; INZA, A.; MARS, J. I.; METAXIAN, J.-P.; MURA, M. D.; MALFANTE, M. Tensor-based learning framework for automatic multichannel volcano-seismic classification. **IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing**, IEEE, v. 14, p. 4517–4529, 2021.

PORGES, T.; FAVIER, G. Automatic target classification in sar images using mpca. In: IEEE. **2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.], 2011. p. 1225–1228.

RAHMAN, M. B.; NURHASANAH, I. S.; NUGROHO, S. P. Community resilience: learning from mt merapi eruption 2010. **Procedia-Social and Behavioral Sciences**, Elsevier, v. 227, p. 387–394, 2016.

REYNEN, A.; AUDET, P. Supervised machine learning on a network scale: Application to seismic event classification and detection. **Geophysical Journal International**, Oxford University Press, v. 210, n. 3, p. 1394–1409, 2017.

ROCHA, D. S.; FAVIER, G.; FERNANDES, C. A. R. Closed-form receiver for multi-hop mimo relay systems with tensor space-time coding. **Journal of Communication and Information Systems**, v. 34, n. 1, p. 50–54, 2019.

RODARMEL, C.; SHAN, J. Principal component analysis for hyperspectral image classification. **Surveying and Land Information Science**, American Congress on Surveying and Mapping, v. 62, n. 2, p. 115–122, 2002.

SAVITZKY, A.; GOLAY, M. J. Smoothing and differentiation of data by simplified least squares procedures. **Analytical chemistry**, ACS Publications, v. 36, n. 8, p. 1627–1639, 1964.

SCARPETTA, S.; GIUDICEPIETRO, F.; EZIN, E. C.; PETROSINO, S.; PEZZO, E. D.; MARTINI, M.; MARINARO, M. Automatic classification of seismic signals at mt. vesuvius volcano, italy, using neural networks. **Bulletin of the Seismological Society of America**, Seismological Society of America, v. 95, n. 1, p. 185–196, 2005.

SHAKHNAROVICH, G.; MOGHADDAM, B. Face recognition in subspaces. In: **Handbook of face recognition**. [S.l.]: Springer, 2005. p. 141–168.

SHIMSHONI, Y.; INTRATOR, N. Classification of seismic signals by integrating ensembles of neural networks. **IEEE transactions on signal processing**, IEEE, v. 46, n. 5, p. 1194–1201, 1998.

SHLENS, J. A tutorial on principal component analysis. **arXiv preprint arXiv:1404.1100**, 2014.

SIDIROPOULOS, N. D.; BRO, R. On the uniqueness of multilinear decomposition of n-way arrays. **Journal of chemometrics**, v. 14, n. 3, p. 229–239, 2000.

SIDIROPOULOS, N. D.; GIANNAKIS, G. B.; BRO, R. Blind parafac receivers for ds-cdma systems. **IEEE Transactions on Signal Processing**, IEEE, v. 48, n. 3, p. 810–823, 2000.

SIDIROPOULOS, N. D.; LATHAUWER, L. D.; FU, X.; HUANG, K.; PAPALEXAKIS, E. E.; FALOUTSOS, C. Tensor decomposition for signal processing and machine learning. **IEEE Transactions on Signal Processing**, IEEE, v. 65, n. 13, p. 3551–3582, 2017.

SOUSA, J.; FERREIRA, A.; BATISTA, G.; SOBRINHO, C.; BASTOS, A.; LYRA, M.; SOMBRA, A. *et al.* Generation of logic gates based on a photonic crystal fiber michelson interferometer. **Optics Communications**, Elsevier, v. 322, p. 143–149, 2014.

SOUZA, F. C. d. N. de; MAIA, L. S. P.; MEDEIROS, G. M. de; MIRANDA, M. A. R.; SASAKI, J. M.; GUIMARAES, G. F. Optical current and magnetic field sensor using mach-zehnder interferometer with nanoparticles. **IEEE Sensors Journal**, IEEE, v. 18, n. 19, p. 7998–8004, 2018.

STEGEMAN, A.; SIDIROPOULOS, N. D. On kruskal's uniqueness condition for the candecomp/parafac decomposition. **Linear Algebra and its applications**, Elsevier, v. 420, n. 2-3, p. 540–552, 2007.

STEINWART, I. Consistency of support vector machines and other regularized kernel classifiers. **IEEE transactions on information theory**, IEEE, v. 51, n. 1, p. 128–142, 2005.

SWIFT, A. J.; LU, H.; UTHOFF, J.; GARG, P.; COGLIANO, M.; TAYLOR, J.; METHERALL, P.; ZHOU, S.; JOHNS, C. S.; ALABED, S. *et al.* A machine learning cardiac magnetic resonance approach to extract disease features and automate pulmonary arterial hypertension diagnosis. **European Heart Journal-Cardiovascular Imaging**, Oxford University Press, v. 22, n. 2, p. 236–245, 2021.

TAHERSIMA, M. H.; KOJIMA, K.; KOIKE-AKINO, T.; JHA, D.; WANG, B.; LIN, C.; PARSONS, K. Deep neural network inverse design of integrated photonic power splitters. **Scientific reports**, Nature Publishing Group, v. 9, n. 1, p. 1–9, 2019.

TANG, K.-S.; MAN, K.-F.; KWONG, S.; HE, Q. Genetic algorithms and their applications. **IEEE signal processing magazine**, IEEE, v. 13, n. 6, p. 22–37, 1996.

TAO, D.; LI, X.; HU, W.; MAYBANK, S.; WU, X. Supervised tensor learning. In: IEEE. **Fifth IEEE International Conference on Data Mining (ICDM'05)**. [S.l.], 2005. p. 8–pp.

TAO, D.; LI, X.; WU, X.; MAYBANK, S. J. General tensor discriminant analysis and gabor features for gait recognition. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 29, n. 10, p. 1700–1715, 2007.

TAO, D.; LI, X.; WU, X.; HU, W.; MAYBANK, S. J. Supervised tensor learning. **Knowledge and Information Systems**, v. 1, n. 13, p. 1–42, 2007.

THARWAT, A.; GABER, T.; IBRAHIM, A.; HASSANIEN, A. E. Linear discriminant analysis: A detailed tutorial. **AI communications**, IOS Press, v. 30, n. 2, p. 169–190, 2017.

TOLIMIERI, R.; AN, M.; LU, C. **Mathematics of multidimensional Fourier transform algorithms**. [S.l.]: Springer Science & Business Media, 2012.

TRAVERSA, P.; LENGLINÉ, O.; MACEDO, O.; MÉTAXIAN, J.-P.; GRASSO, J.-R.; INZA, A.; TAIPE, E. Short term forecasting of explosions at ubinas volcano, perú. **Journal of Geophysical Research: Solid Earth**, Wiley Online Library, v. 116, n. B11, 2011.

TSUI, T. K.; ZHANG, X.-P.; ANDROUTSOS, D. Color image watermarking using multidimensional fourier transforms. **IEEE Transactions on Information Forensics and security**, IEEE, v. 3, n. 1, p. 16–28, 2008.

TUCKER, L. R. Some mathematical notes on three-mode factor analysis. **Psychometrika**, Springer, v. 31, n. 3, p. 279–311, 1966.

VAPNIK, V. **The Nature of Statistical Learning Theory**. [S.l.]: Springer Science & Business Media, 2013.

VAPNIK, V. N. An overview of statistical learning theory. **IEEE transactions on neural networks**, IEEE, v. 10, n. 5, p. 988–999, 1999.

VRABIE, V. D.; BIHAN, N. L.; MARS, J. I. Multicomponent wave separation using hosvd/unimodal-ica subspace method. **Geophysics**, Society of Exploration Geophysicists, v. 71, n. 5, p. V133–V143, 2006.

WANG, X.; JING, X.; ZHU, X.; SUN, S.; HONG, L. A novel approach of fingerprint recognition based on multilinear ica. In: IEEE. **2009 IEEE International Conference on Network Infrastructure and Digital Content**. [S.l.], 2009. p. 740–744.

WOLD, S.; ESBENSEN, K.; GELADI, P. Principal component analysis. **Chemometrics and intelligent laboratory systems**, Elsevier, v. 2, n. 1-3, p. 37–52, 1987.

XIANG, Y.; JIANG, Q.; HE, J.; JIN, X.; WU, L.; YAO, S. The advance of support tensor machine. In: IEEE. **2018 IEEE 16th International Conference on Software Engineering Research, Management and Applications (SERA)**. [S.l.], 2018. p. 121–128.

XIE, Y.; HUANG, T.; JI, Q.; YANG, M.; WANG, J.; TU, X.; CHENG, Z.; XU, G.; WEI, Q.; WU, Y. *et al.* Design of an arbitrary ratio optical power splitter based on a discrete differential multiobjective evolutionary algorithm. **Applied Optics**, Optica Publishing Group, v. 59, n. 6, p. 1780–1785, 2020.

YAN, S.; XU, D.; YANG, Q.; ZHANG, L.; TANG, X.; ZHANG, H.-J. Discriminant analysis with tensor representation. In: IEEE. **2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**. [S.l.], 2005. v. 1, p. 526–532.

YAN, S.; XU, D.; YANG, Q.; ZHANG, L.; TANG, X.; ZHANG, H.-J. Multilinear discriminant analysis for face recognition. **IEEE Transactions on image processing**, IEEE, v. 16, n. 1, p. 212–220, 2006.

YANG, F.-J. An implementation of naive bayes classifier. In: IEEE. **2018 International conference on computational science and computational intelligence (CSCI)**. [S.l.], 2018. p. 301–306.

YUNQI, L.; DONGJIE, C.; MEILING, Y.; QINGMIN, L.; ZHENXIANG, S. 3d face recognition by surface classification image and pca. In: IEEE. **2009 Second International Conference on Machine Vision**. [S.l.], 2009. p. 145–149.

ZANDOMENEGHI, D.; INZA, A.; METAXIAN, J.-P.; MACEDO, O. Long-period seismic events at ubinas volcano (peru): their implications and potentiality as monitoring tool. In: **EGU General Assembly Conference Abstracts**. [S.l.: s.n.], 2012. p. 9359.

ZETIE, K.; ADAMS, S.; TOCKNELL, R. How does a mach-zehnder interferometer work? **Physics Education**, IOP Publishing, v. 35, n. 1, p. 46, 2000.

ZHOU, H.; LI, L.; ZHU, H. Tensor regression with applications in neuroimaging data analysis. **Journal of the American Statistical Association**, Taylor & Francis, v. 108, n. 502, p. 540–552, 2013.

ZHOU, Q.; TONG, G.; XIE, D.; LI, B.; YUAN, X. A seismic-based feature extraction algorithm for robust ground target classification. **IEEE Signal Processing Letters**, IEEE, v. 19, n. 10, p. 639–642, 2012.

ZHU, X.; GOLDBERG, A. B. Introduction to semi-supervised learning. **Synthesis lectures on artificial intelligence and machine learning**, Morgan & Claypool Publishers, v. 3, n. 1, p. 1–130, 2009.

APPENDIX A – SUPPORT VECTOR REGRESSION WITH MULTILINEAR SAMPLING FORMULATION

A.1 Primal Formulation of Support Vector Regression (SVR)

The SVM concepts presented in the previous subsection can be generalized to become applicable to regression problems. As in classification, support vector regression (SVR) is characterized by the use of kernels, sparse solution, control of the margin and the number of support vectors.

The primal formulation of the SVR cost function is given by:

$$\min_{\mathbf{w}} \ \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{p=1}^{P} \left(\xi_p + \tilde{\xi}_p \right), \tag{A.1}$$

subject to

$$y_p - (\langle \mathbf{w}, \mathbf{x}_p \rangle + b) \leq \varepsilon + \xi_p, \tag{A.2}$$

$$(\langle \mathbf{w}, \mathbf{x}_p \rangle + b) - y_p \leq \varepsilon + \tilde{\xi}_p, \tag{A.3}$$

$$\xi_p \geq 0, \ \tilde{\xi}_p \geq 0, \tag{A.4}$$

for $p = 1, ..., P$, where ξ_p and $\tilde{\xi}_p$ are the slack variables, and ε is the scalar margin.

A.2 Primal Formulation of SVR with multilinear sampling using PARAFAC Decomposition

The primal formulation of the SVR cost function with multilinear sampling is given by

$$\min_{\mathbf{w}} \ \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{p_1=1}^{P_1} \sum_{p_2=1}^{P_2} \sum_{p_3=1}^{P_3} \sum_{p_4=1}^{P_4} \left(\xi_{p_1,p_2,p_3,p_4} + \tilde{\xi}_{p_1,p_2,p_3,p_4} \right), \tag{A.5}$$

subject to

$$y_{p_1,p_2,p_3,p_4} - (\langle \mathbf{w}, \mathbf{x}_{p_1,p_2,p_3,p_4} \rangle + b) \leq \varepsilon + \xi_{p_1,p_2,p_3,p_4}, \tag{A.6}$$

$$(\langle \mathbf{w}, \mathbf{x}_{p_1,p_2,p_3,p_4} \rangle + b) - y_{p_1,p_2,p_3,p_4} \leq \varepsilon + \tilde{\xi}_{p_1,p_2,p_3,p_4}, \tag{A.7}$$

$$\xi_{p_1,p_2,p_3,p_4} \geq 0, \ \tilde{\xi}_{p_1,p_2,p_3,p_4} \geq 0, \tag{A.8}$$

for $p_1 = 1,...,P_1$, $p_2 = 1,...,P_2$, $p_3 = 1,...,P_3$ and $p_4 = 1,...,P_4$.

Let $\mathscr{E} \in \mathbb{R}^{P_1 \times P_2 \times P_3 \times P_4}$ and $\tilde{\mathscr{E}} \in \mathbb{R}^{P_1 \times P_2 \times P_3 \times P_4}$ be two fourth order tensors formed from ξ_{p_1,p_2,p_3,p_4} and $\tilde{\xi}_{p_1,p_2,p_3,p_4}$, respectively. Let us assume that the slack tensors $\mathscr{E}$ and $\tilde{\mathscr{E}}$ follow a PARAFAC decomposition:

$$\xi_{p_1,p_2,p_3,p_4} = \sum_{q=1}^{Q} a^{(1)}_{p_1,q} a^{(2)}_{p_2,q} a^{(3)}_{p_3,q} a^{(4)}_{p_4,q}, \tag{A.9}$$

$$\tilde{\xi}_{p_1,p_2,p_3,p_4} = \sum_{\tilde{q}=1}^{\tilde{Q}} \tilde{a}^{(1)}_{p_1,\tilde{q}} \tilde{a}^{(2)}_{p_2,\tilde{q}} \tilde{a}^{(3)}_{p_3,\tilde{q}} \tilde{a}^{(4)}_{p_4,\tilde{q}}, \tag{A.10}$$

where Q and $\tilde{Q}$ are the tensor ranks, $a^{(1)}_{p_1,q}$, $a^{(2)}_{p_2,q}$, $a^{(3)}_{p_3,q}$, $a^{(4)}_{p_4,q}$, $\tilde{a}^{(1)}_{p_1,\tilde{q}}$, $\tilde{a}^{(2)}_{p_2,\tilde{q}}$, $\tilde{a}^{(3)}_{p_3,\tilde{q}}$ and $\tilde{a}^{(4)}_{p_4,\tilde{q}}$ form the matrix factors of the PARAFAC decompositions $\mathbf{A}^{(1)} \in \mathbb{R}^{P_1 \times Q}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{P_2 \times Q}$, $\mathbf{A}^{(3)} \in \mathbb{R}^{P_3 \times Q}$, $\mathbf{A}^{(4)} \in \mathbb{R}^{P_4 \times Q}$, $\tilde{\mathbf{A}}^{(1)} \in \mathbb{R}^{P_1 \times \tilde{Q}}$, $\tilde{\mathbf{A}}^{(2)} \in \mathbb{R}^{P_2 \times \tilde{Q}}$, $\tilde{\mathbf{A}}^{(3)} \in \mathbb{R}^{P_3 \times \tilde{Q}}$ and $\tilde{\mathbf{A}}^{(4)} \in \mathbb{R}^{P_4 \times \tilde{Q}}$.

The optimization problem of the SVR may be stated as follows:

$$\min_{\mathbf{w}} \ \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{p_1=1}^{P_1} \sum_{p_2=1}^{P_2} \sum_{p_3=1}^{P_3} \sum_{p_4=1}^{P_4} \left(\sum_{q=1}^{Q} a^{(1)}_{p_1,q} a^{(2)}_{p_2,q} a^{(3)}_{p_3,q} a^{(4)}_{p_4,q} + \sum_{\tilde{q}=1}^{\tilde{Q}} \tilde{a}^{(1)}_{p_1,\tilde{q}} \tilde{a}^{(2)}_{p_2,\tilde{q}} \tilde{a}^{(3)}_{p_3,\tilde{q}} \tilde{a}^{(4)}_{p_4,\tilde{q}} \right), \tag{A.11}$$

subject to

$$y_{p_1,p_2,p_3,p_4} - (\langle \mathbf{w}, \mathbf{x}_{p_1,p_2,p_3,p_4} \rangle + b) \leq \varepsilon + \sum_{q=1}^{Q} a^{(1)}_{p_1,q} a^{(2)}_{p_2,q} a^{(3)}_{p_3,q} a^{(4)}_{p_4,q}, \tag{A.12}$$

$$(\langle \mathbf{w}, \mathbf{x}_{p_1,p_2,p_3,p_4} \rangle + b) - y_{p_1,p_2,p_3,p_4} \leq \varepsilon + \sum_{\tilde{q}=1}^{\tilde{Q}} \tilde{a}^{(1)}_{p_1,\tilde{q}} \tilde{a}^{(2)}_{p_2,\tilde{q}} \tilde{a}^{(3)}_{p_3,\tilde{q}} \tilde{a}^{(4)}_{p_4,\tilde{q}}, \tag{A.13}$$

$$\sum_{q=1}^{Q} a^{(1)}_{p_1,q} a^{(2)}_{p_2,q} a^{(3)}_{p_3,q} a^{(4)}_{p_4,q} \geq 0, \tag{A.14}$$

$$\sum_{\tilde{q}=1}^{\tilde{Q}} \tilde{a}^{(1)}_{p_1,\tilde{q}} \tilde{a}^{(2)}_{p_2,\tilde{q}} \tilde{a}^{(3)}_{p_3,\tilde{q}} \tilde{a}^{(4)}_{p_4,\tilde{q}} \geq 0, \tag{A.15}$$

for $p_1 = 1,...,P_1$, $p_2 = 1,...,P_2$, $p_3 = 1,...,P_3$ and $p_4 = 1,...,P_4$.

In order to find $a^{(1)}_{p_1,q}$, $a^{(2)}_{p_2,q}$, $a^{(3)}_{p_3,q}$, $a^{(4)}_{p_4,q}$, $\tilde{a}^{(1)}_{p_1,\tilde{q}}$, $\tilde{a}^{(2)}_{p_2,\tilde{q}}$, $\tilde{a}^{(3)}_{p_3,\tilde{q}}$ and $\tilde{a}^{(4)}_{p_4,\tilde{q}}$, we must estimate them iteratively. Assuming that the terms $C^{[1]}_q$ and $\tilde{C}^{[1]}_q$ are known and given by:

$$C^{[1]}_q = \sum_{p_2=1}^{P_2} \sum_{p_3=1}^{P_3} \sum_{p_4=1}^{P_4} a^{(2)}_{p_2,q} a^{(3)}_{p_3,q} a^{(4)}_{p_4,q} \tag{A.16}$$

and

$$\tilde{C}^{[1]}_{\tilde{q}} = \sum_{p_2=1}^{P_2} \sum_{p_3=1}^{P_3} \sum_{p_4=1}^{P_4} \tilde{a}^{(2)}_{p_2,\tilde{q}} \tilde{a}^{(3)}_{p_3,\tilde{q}} \tilde{a}^{(4)}_{p_4,\tilde{q}}, \tag{A.17}$$

Then, we can rewrite (A.11) as follows:

$$\min_{\mathbf{w}} \ \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \left(\sum_{q=1}^{Q} C_q^{[1]} \sum_{p_1=1}^{P_1} a_{p_1,q}^{(1)} + \sum_{\tilde{q}=1}^{\tilde{Q}} \tilde{C}_q^{[1]} \sum_{p_1=1}^{P_1} \tilde{a}_{p_1,\tilde{q}}^{(1)} \right) \tag{A.18}$$

subject to

$$y_{p_1,p_2,p_3,p_4} - \left(\langle \mathbf{w}, \mathbf{x}_{p_1,p_2,p_3,p_4} \rangle + b \right) \leq \varepsilon + \sum_{q=1}^{Q} a_{p_1,q}^{(1)} u_{p2,p3,p_4,q}^{[1]}, \tag{A.19}$$

$$\left(\langle \mathbf{w}, \mathbf{x}_{p_1,p_2,p_3,p_4} \rangle + b \right) - y_{p_1,p_2,p_3,p_4} \leq \varepsilon + \sum_{\tilde{q}=1}^{\tilde{Q}} \tilde{a}_{p_1,\tilde{q}}^{(1)} \tilde{u}_{p2,p3,p_4,\tilde{q}}^{[1]}, \tag{A.20}$$

$$\sum_{q=1}^{Q} a_{p_1,q}^{(1)} u_{p2,p3,p_4,q}^{[1]} \geq 0, \tag{A.21}$$

$$\sum_{\tilde{q}=1}^{\tilde{Q}} \tilde{a}_{p_1,\tilde{q}}^{(1)} \tilde{u}_{p2,p3,p_4,\tilde{q}}^{[1]} \geq 0, \tag{A.22}$$

for $p_1 = 1,...,P_1$, $p_2 = 1,...,P_2$, $p_3 = 1,...,P_3$ and $p_4 = 1,...,P_4$, $u_{p2,p3,p_4,q}^{[1]} = a_{p_2,q}^{(2)} a_{p_3,q}^{(3)} a_{p_4,q}^{(4)}$ and $\tilde{u}_{p2,p3,p_4,\tilde{q}}^{[1]} = \tilde{a}_{p_2,\tilde{q}}^{(2)} \tilde{a}_{p_3,\tilde{q}}^{(3)} \tilde{a}_{p_4,\tilde{q}}^{(4)}$.

APPENDIX B – *PACKET CLASSIFICATION USING SUPPORT TENSOR MACHINES*

Submitted to XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais, Santa Rita do Sapucaí, Minas Gerais, 2022.

Packet Classification using Support Tensor Machines

Antonio Augusto Teixeira Peixoto, Laise Santos e Carlos Alexandre Rolim Fernandes

***Abstract*— A wide variety of packet classification algorithms exist in the research literature and commercial market. The existing solutions exploit various design tradeoffs, providing high search rates, power and space efficiency and the ability to scale to large numbers of filters. However, still remains a need for techniques that achieve a favorable balance among these tradeoffs and scale to support classification. Based on this motivations, this paper presents a tensor approach for the classification of TCP and UDP packets. By using a multidimensional structure, more specifically a 4-th order tensor, to store the packet data, a tensorial algorithm known as Support Tensor Machines (STM) is used to perform classification. Results showed good performance of the approach in comparison to other classifiers such as the Support Vector Machines and Naive-Bayes.**

***Keywords*— packet classification, tensor, support tensor machines.**

I. Introduction

The process of classifying information is directly related to categorization, where ideas, objects or data are recognized, differentiated, understood and then, separated in different tags [1]. Moreover, data classification can be achieved with methods aimed to determine whether or not the data contains some specific information, feature, or behavior, then, identifying the correct class of the data [2]. This method is an important branch of computer vision, machine learning and computational intelligence, being used in many fields such as geophysics (seismic recognition, seismic swarms) [3], recognition of fingerprint images [4], face [5], handwritten digits [6], gait [7], electrocardiogram signals [8], and even identification of specific vehicles [9].

The most common methods of classification are based on supervised learning, which is the task of learning a function that maps an input to an output based on example input-output pairs [10]. In supervised learning, each example is a pair consisting of an input object, typically a vector, and a desired output value, also called the supervisory signal. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. A common learning algorithm that has been reported used on various classification applications such as the mentioned earlier, is the the support vector machine (SVM) [11], mostly employed to solve two-class problems, but also used on multi-class solutions [12].

Augusto Peixoto, Redes de Computadores, IFCE, Jaguaribe-CE, e-mail: antonio.peixoto@ifce.edu.br; Laise Santos, Redes de Computadores, IFCE, Jaguaribe-CE, e-mail: layse0877@gmail.com; Carlos Alexandre R. Fernandes, Engenharia da Computação, UFC, Sobral-CE, e-mail: alexandrefernandes@ufc.br

More specifically, a standard SVM model is based on vector inputs and cannot directly deal with matrices or higher dimensional data structures, namely, tensors, which are very common in real-life applications [15]. The SVM realization on such high dimensional inputs is by reshaping each sample into a vector. However, when the training data sample size is relatively small compared to the feature vector dimension, it may easily result in poor classification performance, known as curse of dimensionality [15], [16].

In order to avoid the data destruction by converting tensors into vectors, the supervised tensor learning method is proposed [17]. This technique has been extensively studied in recent years, and, proposes a supervised tensor learning framework, which extends the standard linear SVM framework to tensor patterns by constructing multilinear models, hence, called Support Tensor Machines (STM). Also, the technique utilizes a rank-one tensor to capture the data structure, thereby alleviating the overfitting and curse of dimensionality problems in the conventional SVM [17], [18]. Moreover, in the context of supervised tensor learning, preserving the structural information and exploiting the discriminating nonlinear relationships of tensor data are crucial for improving the performance of learning tasks [19].

On the other hand, classification of data packets in computer networks is a very demanding task [20]. Packet classification is important for applications such as firewalls, intrusion detection, and differentiated services. Existing algorithms for packet classification reported in the literature scale poorly in either time or space as filter databases grow in size [21]. Also, existing solutions may require high computational cost. In the work of [22], an overview of packet classification algorithms is presented. As explained [22], researchers have proposed a variety of algorithms which, broadly speaking, can be categorized as basic search algorithms, geometric algorithms, heuristic algorithms, or hardware-specific search algorithms.

Moreover, the use of tensors and tensor-based classifiers in packet classification is not common in the literature, with few works addressing the topic. We may cite [23], which utilizes a multidimensional approach for multi-scale feature attention approach to network traffic classification, by using convolutional neural networks (CNN) as the building block of the deep packet analysis model. With so few tensor-based works in covering this topic, it is desirable the development of multilinear algorithms that could be used in packet classification.

In this paper, a tensor-based approach for the classification of TCP and UDP packets is presented. By using a multidimensional structure, to store the packet data, a tensorial algorithm

known as STM is used to perform classification. The first step is a creation of a 4-th order tensor from two packet databases, with information from TCP and UDP segments obtained through a packet capture software. After obtaining the tensor, the data is classified using the STM algorithm, achieving interesting results. For comparison purposes, both linear SVM and Naive-Bayes were tested.

A. Organization

The rest of this paper is organized as follows. Section II presents methods used, while Section III describes the Support Tensor Machine formulation. Section IV introduces the STM algorithm, Section V presents the results and, finally, Section VI ends this paper with the conclusions.

B. Notation

Scalars are denoted by Roman lower-case letters (a,b,...), vectors as lower-case boldface letters (**a**,**b**,...), matrices as upper-case boldface letters (**A**,**B**,...) and tensors as calligraphic letters ($\mathcal{A}$,$\mathcal{B}$,...). To retrieve the element (i,j) of an arbitrary matrix $\mathbf{A} \in \mathbb{C}^{I\times R}$, we use $a_{i,j}$ (the same for tensors). $\mathbf{A}^T$ stands for the transpose. Also, $\otimes$ denotes the Kronecker product between $\mathbf{A} \in \mathbb{R}^{I\times K}$ and $\mathbf{B} \in \mathbb{R}^{J\times L}$, resulting in $\mathbf{A}\otimes\mathbf{B} \in \mathbb{C}^{IJ\times KL}$. A matrix unfold of a 4-th order tensor $\mathcal{X} \in \mathbb{R}^{K\times M\times N\times P}$ is given by $\mathbf{X}^{[n]}$, i.e. $\mathbf{X}^{[1]} \in \mathbb{R}^{KMN\times N}$.

II. Methods

The methodology of this work was developed as follows. Initially, TCP and UDP segment data were collected using the Wireshark software. This software is a network protocol analyzer, allowing the visualization of data from a given network, allowing the user to interact by browsing through the captured data and seeing the details of each packet [24]. Two networks were analyzed for packets, one wired and the other one wireless, where packets were obtained during conventional web browsing, video calls and streaming services.

The data collected from the TCP and UDP packets were as follows:

- Source port;
- Destination port;
- Header length;
- Window size;
- Payload (in bytes);
- Flag;

The attributes described above were chosen because they facilitate the distinction between TCP and UDP segments, thus facilitating the classification of these later on. After collecting 100 samples of TCP and UDP segments from the wired network, another 100 samples were obtained from the wireless network. Once the samples were properly tabulated, a 3rd order tensor containing the database was assembled, with the tensor dimensions being $2 \times 3 \times 6 \times 100$, with a total of 600 packets. The first mode of the tensor denotes the two types of network: wired and wireless, the second mode refers to how the packets were obtained (web browsing, video call or streaming), the third mode refers to the number of attributes whereas the fourth mode indicates the number of samples. The class tags are two: a packet of the dataset is either TCP or UDP. In Figure 1 we can see the flowchart of the methodology.

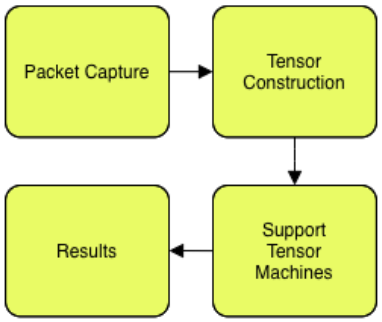


Fig. 1. Flowchart of the presented classification method.

TABLE I
DATABASE DESCRIPTION.

Size: 2 × 3 × 6 × 100	**Number of Packets**
Wired Network	100 (60 TCP, 40 UDP)
Wireless Network	100 (60 TCP, 40 UDP)

Table I shows the dataset description, with tensor dimensions, number of TCP and UDP packets per type of network. In the following, the adopted classifier is presented.

III. Support Tensor Machines (STM) Formulation

Considered an extension to the conventional SVM, the STM works in the following way. Let $\mathcal{X} \in \mathbb{R}^{K\times M\times N\times P}$ be the training data set tensor split into P third order tensors and a vector **y** consisting of the class tags associated to each of the P tensors, with **y** $\in$ {-1,1}, thus, we need to find a tensor classifier such the two classes can be separated with maximum margin as the decision function:

$$f(\mathcal{X}_p) = \mathcal{X}_p(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \mathbf{v}^{(3)}) + b, \tag{1}$$

where $\mathcal{X}_p \in \mathbb{R}^{K\times M\times N}$, $\mathbf{v}^{(1)} \in \mathbb{R}^{1\times K}$, $\mathbf{v}^{(2)} \in \mathbb{R}^{1\times M}$ and $\mathbf{v}^{(3)} \in \mathbb{R}^{1\times N}$ are vectors orthogonal to the hyperplane. We can also define:

$$\mathcal{X}_p(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \mathbf{v}^{(3)}) = \sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N} x_{k,m,n}\mathbf{v}_k^{(1)}\mathbf{v}_m^{(2)}\mathbf{v}_n^{(3)}. \tag{2}$$

Then, (1) can be rewritten as follows:

$$f(\mathcal{X}_p) = \mathcal{X}_p \times_1 \mathbf{v}^{(1)} \times_2 \mathbf{v}^{(2)} \times_3 \mathbf{v}^{(3)} + b. \tag{3}$$

In matricial notation, we have:

$$f(\mathbf{X}_p) = (\mathbf{v}^{(2)} \otimes \mathbf{v}^{(3)})\mathbf{X}_p^{(1)}(\mathbf{v}^{(1)})^T + b, \tag{4}$$

where $\mathbf{X}^{[1]} \in \mathbb{R}^{MN\times K}$ is an unfolding of the tensor $\mathcal{X}_p$.

As stated earlier, the LR-STM method is an generalization of the SVM for higher order arrays. In order to use (3), it

is necessary to compute $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$. The algorithm is described as follows:

- Initialize $\mathbf{v}^{(2)}$ = {1,...,1} $\in \mathbb{R}^{1\times M}$ and $\mathbf{v}^{(3)}$ = {1,...,1} $\in \mathbb{R}^{1\times N}$;
- Let $\mathbf{x}_p = (\mathbf{X}_p^{[1]})^T(\mathbf{v}^{(2)} \otimes \mathbf{v}^{(3)})^T$;
- Compute $\mathbf{v}^{(1)}$ by solving the following optimization problem:

$$\min_{\mathbf{v}^{(1)}} \frac{1}{2}\langle \mathbf{v}^{(1)}, \mathbf{v}^{(1)}\rangle + C\sum_{p=1}^{P}\xi_p \tag{5}$$

subject to

$$\mathbf{y}_p(\langle \mathbf{v}^{(1)}, \mathbf{x}_p\rangle + b) \geq 1 - \xi_p, \tag{6}$$

where

$$\xi_p \geq 0.$$

- With $\mathbf{v}^{(1)}$ obtained, let $\mathbf{x}_p = (\mathbf{X}_p^{[2]})^T(\mathbf{v}^{(3)} \otimes \mathbf{v}^{(1)})^T$, then compute $\mathbf{v}^{(2)}$ with:

$$\min_{\mathbf{v}^{(2)}} \frac{1}{2}\langle \mathbf{v}^{(2)}, \mathbf{v}^{(2)}\rangle + C\sum_{p=1}^{P}\xi_p \tag{7}$$

subject to

$$\mathbf{y}_p(\langle \mathbf{v}^{(2)}, \mathbf{x}_p\rangle + b) \geq 1 - \xi_p, \tag{8}$$

where

$$\xi_p \geq 0.$$

- With $\mathbf{v}^{(1)}$ and $\mathbf{v}^{(2)}$ obtained, let $\mathbf{x}_p = (\mathbf{X}_p^{[3]})^T(\mathbf{v}^{(1)} \otimes \mathbf{v}^{(2)})^T$, then compute $\mathbf{v}^{(3)}$ with:

$$\min_{\mathbf{v}^{(3)}} \frac{1}{2}\langle \mathbf{v}^{(3)}, \mathbf{v}^{(3)}\rangle + C\sum_{p=1}^{P}\xi_p \tag{9}$$

subject to

$$\mathbf{y}_p(\langle \mathbf{v}^{(3)}, \mathbf{x}_p\rangle + b) \geq 1 - \xi_p, \tag{10}$$

where

$$\xi_p \geq 0.$$

- After the steps above, we can iteratively compute $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$ until they converge.

Now, with $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$ obtained, we can classify the test data $\mathcal{Z}_s \in \mathbb{R}^{K\times M\times N}$, with s = 1,...,S and $J = P + S$, where P and S denotes the training and test samples, respectively. Thus we have:

$$g(\mathcal{Z}_s) = sign(\mathcal{Z}_s \times_1 \mathbf{v}^{(1)} \times_2 \mathbf{v}^{(2)} \times_3 \mathbf{v}^{(3)} + b). \tag{11}$$

IV. STM Estimation Algorithm

The algorithm of the STM method, in pseudo-code format, is shown in Algorithm 1. The basic process of classification is illustrated in Figure 2, where, with the training samples, we build the LR-STM model by estimating $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$. Then, we use the model to classify the test samples with (11).

Convergence is achieved as follows. As we shown, the optimizations problems in (5,7,9) are the same as in the standard SVM algorithm, then we can use the computational

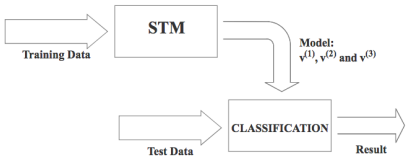


Fig. 2. Flowchart of the presented classification method.

Algorithm 1 STM Algorithm

For p = 1,...,P:
1)Initialization: Set i = 0; Initialize $\mathbf{v}_i^{(2)}$ and $\mathbf{v}_i^{(3)}$;
2)With $\mathbf{x}_p = (\mathbf{X}_p^{[1]})^T(\mathbf{v}_i^{(2)} \otimes \mathbf{v}_i^{(3)})^T$, estimate $\mathbf{v}_i^{(1)}$ using (5);
3)i = i + 1;
4)With $\mathbf{x}_p = (\mathbf{X}_p^{[2]})^T(\mathbf{v}_{i-1}^{(3)} \otimes \mathbf{v}_{i-1}^{(1)})^T$, estimate $\mathbf{v}_i^{(2)}$ using (7);
5)With $\mathbf{x}_p = (\mathbf{X}_p^{[3]})^T(\mathbf{v}_{i-1}^{(1)} \otimes \mathbf{v}_i^{(2)})^T$, estimate $\mathbf{v}_i^{(3)}$ using (9);
6)Repeat steps 2-5 until convergence;

methods for SVM to solve (5,7,9). As for the convergence of the algorithm, the iterative procedure to solve the optimization problems (5,7,9) will monotonically decreases the objective function values in (6,8,10), and hence the STM algorithm converges.

Let $\mathbf{v}_0^{(2)}$ and $\mathbf{v}_0^{(3)}$ be the initial values. Fixing $\mathbf{v}_0^{(2)}$ and $\mathbf{v}_0^{(3)}$, we get $\mathbf{v}_0^{(1)}$ by solving the optimization problem (5). Likewise, fixing $\mathbf{v}_0^{(1)}$ and $\mathbf{v}_0^{(3)}$, we get $\mathbf{v}_1^{(2)}$ by solving the optimization problem (7) and so on. Notice that the optimization problem of SVM is convex, so the solution of SVM is globally optimum [25]. Thus, we have:

$$f(\mathbf{v}_0^{(1)}, \mathbf{v}_0^{(2)}, \mathbf{v}_0^{(3)}) \geq f(\mathbf{v}_0^{(1)}, \mathbf{v}_1^{(2)}, \mathbf{v}_0^{(3)}). \tag{12}$$

And finally we get:

$$f(\mathbf{v}_0^{(1)}, \mathbf{v}_0^{(2)}, \mathbf{v}_0^{(3)}) \geq \ldots \geq f(\mathbf{v}_1^{(1)}, \mathbf{v}_1^{(2)}, \mathbf{v}_1^{(3)}) \geq \ldots. \tag{13}$$

V. Results

In this section, the obtained classification results are presented. The STM algorithm was implemented using MATLAB 2017b, in a 9-th generation core i3 processor. For comparison purposes, the conventional SVM with linear kernel and the Naive-bayes classifiers were tested, using a vectorized version of the data, in order to better evaluate the STM performance. Also, an Artificial Neural Network (ANN) implementing forward propagation with two fully connected layers, Rectified linear unit (ReLU) activation functions and a softmax function to the final fully connected layer, using vectorized data, was tested against the STM.

The data was classified using K-fold cross validation, with K = 10 and the relaxing constant was set C = 100. The results are presented in the form of accuracy, which is defined as the number of correctly classified samples over the total number of samples, execution time, in seconds, of the feature technique plus classifier, and, confusion tables. The results were averaged 100 times to eliminate fluctuation.

The first result, presented in Table II shows the accuracy and execution times of the tested methods. As can be seen,

the STM showed the highest accuracy, thus showing its good performance in packet classification. The second best classification rate was achieved by the ANN, however, at a cost of more execution time. The worst accuracy was obtained by the Naive-Bayes classifies. Moreover, although the STM demanded more running time than the SVM and Naive-Bayes techniques, it achieved higher accuracy, showing a trade-off between performance of classification and computational cost.

TABLE II
ACCURACY AND EXECUTION TIME RESULTS FOR THE TESTED TECHNIQUES

Classifier	Accuracy	Time (s)
STM	**88.3**%	201.11
SVM	80.3%	171.2
Naive-Bayes	75.4%	126.8
ANN	83.2%	244.67

Furthermore, in Table III, the obtained confusion matrix is shown, illustrating the performance of STM in the TCP and UDP packet classification process. As we can see, an accuracy of 88.3% was achieved in the classification of the data.

TABLE III
CONFUSION TABLE - ACCURACY OF THE STM.

88.3% Accuracy	TCP	UDP
TCP	330	40
UDP	30	200

VI. CONCLUSIONS

In this work, a tensor-based approach for packet classification was presented. By using a multidimensional structure, to store the TCP and UDP packets, a tensorial algorithm known as STM is used to perform classification. Results showed good performance of the proposed approach in comparison to other classifiers such as the SVM and Naive-Bayes techniques.

REFERENCES

[1] M. Pardo and G. Sberveglieri, “Learning from data: A tutorial with emphasis on modern pattern recognition methods," *IEEE Sensors Journal*, vol. 2, no. 3, pp. 203-217, 2009.

[2] S. R. Kulkarni, G. Lugosi and S. S. Venkatesh, “Learning pattern classification-a survey," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2178-2206, 1998.

[3] Rouet-Leduc, B., Hulbert, C., Lubbers, N., Barros, K., Humphreys, C. J., & Johnson, P. A. (2017). Machine learning predicts laboratory earthquakes. Geophysical Research Letters, 44(18), 9276-9282.

[4] Jia, J., Cai, L., Lu, P., & Liu, X. (2007). Fingerprint matching based on weighting method and the SVM. Neurocomputing, 70(4-6), 849-858.

[5] M. J. Lyons, J. Budynek, and S. Akamatsu, “Automatic classification of single facial images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 21, no. 12, pp. 1357-1362, 1999.

[6] B. Savas, and L. Eldén, “Handwritten digit classification using higher order singular value decomposition," *Pattern recognition*, vol. 40, no. 3, pp. 993-1003, Elsevier, 2007.

[7] R. K. Begg, M. Palaniswami and B. Owen, “Support vector machines for automated gait classificatio," *IEEE transactions on Biomedical Engineering*, vol. 52, no. 5, pp. 828-838, 2005.

[8] F. Melgani & Y. Bazi, “Classification of electrocardiogram signals with support vector machines and particle swarm optimization," *IEEE transactions on information technology in biomedicine*, vol. 12, vol. 5, pp. 667-677, 2008.

[9] S. Gupte, O. Masoud, R. F. Martin and N. P. Papanikolopoulos, Detection and classification of vehicles. *IEEE Transactions on intelligent transportation systems*, vol. 3, no. 1, pp. 37-47, 2002.

[10] Zhu, X., & Goldberg, A. B. (2009). Introduction to semi-supervised learning. Synthesis lectures on artificial intelligence and machine learning, 3(1), 1-130.

[11] Vapnik, V. (2013). The nature of statistical learning theory. Springer science & business media.

[12] Mathur, A., & Foody, G. M. (2008). Multiclass and binary SVM classification: Implications for training and classification users. IEEE Geoscience and remote sensing letters, 5(2), 241-245.

[13] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery, 2(2), 121-167.

[14] Ma, L., Hu, Y., & Zhang, Y. (2017). Support Tucker Machines Based Bubble Defect Detection of Lithium-ion Polymer Cell Sheets. Engineering Letters, 25(1).

[15] Chen, C., Batselier, K., Ko, C. Y., & Wong, N. (2019, July). A support tensor train machine. In 2019 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

[16] Li, J., Allinson, N., Tao, D., & Li, X. (2006). Multitraining support vector machine for image retrieval. IEEE Transactions on Image Processing, 15(11), 3597-3601.

[17] D. Tao, X. Li, X. Wu, W. Hu, and S. J. Maybank. Supervised tensor learning. Knowledge and Information Systems, 13(1):1?42, 2007.

[18] Cai, D., He, X., Wen, J. R., Han, J., & Ma, W. Y. (2006). Support tensor machines for text categorization.

[19] He, L., Lu, C. T., Ma, G., Wang, S., Shen, L., Yu, P. S., & Ragin, A. B. (2017, August). Kernelized support tensor machines. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 1442-1451). JMLR. org.

[20] Taylor, David E., and Jonathan S. Turner. "Scalable packet classification using distributed crossproducing of field labels." In Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., vol. 1, pp. 269-280. IEEE, 2005.

[21] Baboescu, Florin, and George Varghese. "Scalable packet classification." IEEE/ACM transactions on networking 13, no. 1 (2005): 2-14.

[22] Gupta, Pankaj, and Nick McKeown. "Algorithms for packet classification." IEEE Network 15, no. 2 (2001): 24-32.

[23] Wang, Yipeng, Xiaochun Yun, Yongzheng Zhang, Chen Zhao, and Xin Liu. "A Multi-scale Feature Attention Approach to Network Traffic Classification and Its Model Explanation." IEEE Transactions on Network and Service Management (2022).

[24] Lamping, Ulf, and Ed Warnicke. "Wireshark user’s guide." Interface 4, no. 6 (2004): 1.

[25] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20, no. 3 (1995): 273-297.