Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Multilinear Sampling Framework: A New Approach for Machine Learning with Sampling in Multidimensional Grids

C. Alexandre R. Fernandes
Universidade Federal do Ceará
Antonio A. T. Peixoto
Federal Institute of Ceará
Dnilson C. de Sousa
dnilsonsousa@alu.ufc.br
Universidade Federal do Ceará
Glendo de F. Guimarães
Instituto Federal do Ceará
Darlan A. Barroso
Universidade Federal do Ceará

Research Article

Keywords: Machine learning, multidimensional grid, sampling, support vector machine, tensor, photonics

Posted Date: May 12th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9635705/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Multilinear Sampling Framework: A New Approach for Machine Learning with Sampling in Multidimensional Grids

C. Alexandre R. Fernandes[1,4], Antonio A. T. Peixoto[2], Dnilson C. de Sousa[1*], Glendo de F. Guimarães[3], Darlan A. Barroso[4]

[1*]Department of Computer Engineering, Federal University of Ceará, Rua Coronel Estanislau Frota, 563, Sobral, 62010-560, Ceará, Brazil.
[2]Campus Jaguaribe, Federal Institute of Ceará, R. Pedro Bezerra de Menezes, Jaguaribe, 63475-000, Ceará, Brazil.
[3]Postgraduate Program in Telecommunications Engineering, Federal Institute of Ceará, Av. Treze de Maio, 2081, Fortaleza, 60410-426, Ceará, Brazil.
[4]Postgraduate Program in Teleinformatics Engineering, Federal University of Ceará, Campus do Pici - Bloco 725, Fortaleza, 60455-970, Ceará, Brazil.

*Corresponding author(s). E-mail(s): dnilsonsousa@alu.ufc.br;
Contributing authors: alexandrefernandes@ufc.br;
antonio.peixoto@ifce.edu.br; glendofreitas@ifce.edu.br;
darlan.engemec@alu.ufc.br;

Abstract

Many Machine Learning (ML) applications exhibit a natural multidimensional sampling structure that conventional vector-based methods discard. In this work, we propose multilinear sampling (MS), a framework in which the training set is organized as a high-order tensor whose modes correspond to the indices of the sampling grid, rather than concatenated into a single sample index. We instantiate the framework by modifying the support vector machine (SVM) cost function so that the multi-index structure of the slack variables is preserved, and propose two classifiers based on the PARAFAC and Tucker tensor decompositions, denoted SVM-MSP and SVM-MST. On two photonic datasets—a Mach-Zehnder

interferometer and light scattering in a gold cylindrical nanostructure—the proposed techniques reach up to 85.6% and 86.3% accuracy, respectively, providing competitive results with respect to standard vector-based baselines. Rather than establishing superiority over existing tensor learning methods, our goal is to highlight the multilinear-sampling perspective as a research direction in which the geometry of the sampling design is treated as a first-class component of the learning algorithm.

Keywords: Machine learning, multidimensional grid, sampling, support vector machine, tensor, photonics

1 Introduction

Machine learning (ML) encompasses a broad range of algorithms and computational tools used in a wide variety of applications (Jordan & Mitchell, 2015). Usually, in ML applications, the input data are represented by matrices with dimensions associated with the features and the samples. However, in many problems, the data are naturally sampled on structured multidimensional grids. In such cases, the sampling indices are often vectorized to be processed by standard ML methods. Nevertheless, this vectorization disregards the underlying grid structure of the data, which may lead to performance degradation and a loss of relevant structural information.

Multidimensional sampling structures arise in a wide range of applications, such as the design of logic gates using photonic crystal fibers (Martins et al., 2018) and Mach-Zehnder interferometers (MZI) (Fernandes, de Oliveira, Rocha, de Alexandria, & Guimarães, 2022), the design of photonic structures (W. Ma et al., 2021; Maia et al., 2025), optical devices (Xie et al., 2020), metasurfaces (Yeung et al., 2021), and other modeling problems (Dong et al., 2025; Tanaka, Tomiya, & Hashimoto, 2021). In particular, when ML methods model dynamic systems (in either direct or reverse design), researchers commonly build a training database by sweeping each input variable over a prescribed range, varying one input at a time.

In the present work, a new framework for ML, denoted by multilinear sampling (MS), is proposed assuming that the samples are obtained from a multidimensional grid. The basic assumption of this approach is that the input samples are obtained by adhering to a multidimensional sampling structure, in which the samples are generated from a sweep (scan) of the input variables over a certain range of values, with the samples generated by changing one element of the input vector at a time. It is considered that there are multiple indices associated with the sampling process, rather than only one index related to the samples, which is usually considered. Multidimensional grid-based representations can be viewed as an alternative to conventional vector-based ML approaches, with the objective of preserving the inherent structure induced by the sampling process. By explicitly exploiting the grid organization of the samples, these representations allow ML models to better capture the dependencies across different sampling dimensions, while avoiding the structural degradation caused by vectorization.

The multidimensional structure exploited by MS relies on how the samples were obtained, not on the structure of the input vector itself, as in standard tensor learning methods. In conventional tensor learning, a multidimensional structure is assumed for the attributes, which are modeled as tensors (Deng, Shi, & Yao, 2023). In the MS approach, the multidimensional structure relies on the sampling process. Representative examples of the conventional tensor learning paradigm include support tensor machines and their variants, which extend the classical support vector machine (SVM) formulation to operate directly on tensor-valued inputs by exploiting multilinear algebraic models (Chen, Batselier, Yu, & Wong, 2022; Kotsia & Patras, 2011; Kour, Dolgov, Stoll, & Benner, 2023; L. Ma, Hu, & Zhang, 2017).

In the proposed framework, each input sample is a vector, as in vector-based ML techniques. As a consequence, in the MS framework, the training database can be organized in a high-order tensor, with all but one of the dimensions corresponding to the sampling modes. In other words, each mode of the database array can be used to store a predetermined sample set. The main concept of MS is to take advantage of a multidimensional sampling structure to accommodate input data. In contrast, in a classical ML approach, all the samples would be concatenated into a single sampling index; however, this would break down the multidimensional structure of the samples, leading to a loss of structural information.

The multidimensional structure of the samples is exploited in the task of statistical classification. In particular, the MS framework is applied to the SVM, which is one of the most popular classifiers in the literature, used in a wide variety of applications. The cost function of the SVM is modified to cope with the multiple indices of the MS framework without the vectorization of the sampling modes. Two modifications of the SVM are proposed by modeling the slack variables using the Parallel Factor Analysis (PARAFAC) decomposition (Carroll & Chang, 1970; Harshman, 1970; Hitchcock, 1927), also known as Canonical decomposition (CANDECOMP) or Canonical Polyadic (CP), and the Tucker decomposition (Tucker, 1966). This gives rise to two new classifiers, denoted respectively by SVM with MS using PARAFAC (SVM-MSP) and SVM with MS using Tucker decomposition (SVM-MST).

To evaluate the performance of the proposed classifiers, two different databases are used. The first one is a photonic database. In particular, the methods are used to classify the binary outputs of an MZI. Optical devices, such as the MZI, have been used in many computer and engineering applications, such as optical sensors, modulators, logic gates, and others (de Souza et al., 2018). Moreover, the modeling of photonic devices or structures using ML techniques has been of great interest for researchers in the past years (Fernandes et al., 2022; W. Ma et al., 2021; Maia et al., 2025; Yeung et al., 2021).

Due to the high number of parameters of the MZI, we sweep some parameters over a given range while keeping the others fixed. The parameters chosen to vary in the database are the two binary inputs and two phase shifts. The input data are then organized in a five-way array with dimensions $4 \times 2 \times 2 \times 40 \times 40$, where the first mode corresponds to the four input attributes and the other modes correspond to the sampling modes. Simulation results show that the proposed classifiers reach 85.6%

accuracy on this dataset, providing a meaningful gain over standard vector-based baselines (the strongest of which, an MLP, reaches 77.6%).

The proposed techniques are also tested in a second dataset, built from light scattering in a gold cylindrical nanostructure. The modeling of light scattering in gold nanostructures using ML is essential for many applications (H. Zhang, Ijaz, & Blaikie, 2023). Indeed, ML offers a powerful alternative by enabling rapid predictions of the scattering behavior. In this database, the parameters radius, height, and incident wavelength are systematically varied, with one parameter varying at a time while the others are held constant. This leads to an input tensor of dimensions $3 \times 11 \times 21 \times 201$, where the first mode corresponds to the three input attributes and the other modes correspond to the sampling modes.

The aim of the present work is not to establish the proposed classifiers as superior to existing tensor learning methods, but to highlight the multilinear structure of the sampling process as a feature that can be directly incorporated into the SVM cost function, and to provide empirical evidence on two photonic datasets that supports the viability of this perspective. A more thorough comparison with kernelized tensor SVMs and other tensor learning paradigms defines a natural continuation of this research direction and is left for future work.

The remainder of this work is structured as follows. Section 3 recalls the tensor notation and decompositions used throughout the paper. Section 4 presents the proposed MS framework. Sections 5 and 6 introduce the SVM-MSP and SVM-MST classifiers. Section 7 describes the datasets, Section 8 reports the results, and Section 9 concludes the paper.

2 Related Work

2.1 Tensor Learning and Support Tensor Machines

Tensor learning has been developed as an alternative to vector-based ML, avoiding the destruction of multidimensional structure caused by data vectorization (Deng et al., 2023). Tensor-based formulations generalize conventional ML techniques to operate directly on tensor-valued inputs, preserving the multidimensional information of the data. Representative examples include image processing (L. Ma et al., 2017), hyperspectral image analysis (Liu, Long, Zheng, & Xiao, 2025), multi-view clustering (X. Zhang et al., 2024), infrared target detection (Luo, Li, Yan, & Xia, 2023), and deep network priors (Saragadam, Balestriero, Veeraraghavan, & Baraniuk, 2024). Within this context, the family of support tensor machines has gained considerable popularity, extending the SVM formulation to tensor-valued inputs through multilinear models (Chen et al., 2022; Kotsia & Patras, 2011; Kour et al., 2023; Tao, Li, Wu, Hu, & Maybank, 2007).

2.2 Multivariate Approximation on Structured Grids

The use of input samples generated according to a structured multidimensional sampling scheme is well established in multivariate approximation theory, where individual variables are swept over prescribed ranges (Bungartz & Griebel, 2004; Cohen,

DeVore, & Schwab, 2011; Hackbusch, 2012). Such assumptions naturally arise in the approximation of multivariate functions defined on tensor-product grids, where function evaluations are collected by varying one coordinate at a time. This setting is central to a wide class of approximation methods, including sparse-grid techniques, tensor-product approximations, and low-rank representations, which exploit the separable and structured nature of multidimensional sampling to mitigate the curse of dimensionality.

2.3 Machine Learning for Photonic Modeling

ML has been increasingly used to model photonic devices and structures, including MZIs (Fernandes et al., 2022), plasmonic nanostructures (Maia et al., 2025), photonic structures more generally (W. Ma et al., 2021), optical splitters (Xie et al., 2020), and metasurfaces (Dong et al., 2025; Yeung et al., 2021). The motivation is largely computational: the design of such devices typically relies on solving partial differential equations or coupled-mode equations whose evaluation, although well understood, is expensive enough that mapping a continuous design space requires a structured exploration strategy. As a consequence, training databases in this domain are most commonly built by sweeping each geometric or optical parameter over a prescribed range, varying one parameter at a time, which produces precisely the kind of grid-organized data assumed by the MS framework proposed here.

2.4 Position of This Work

The position of this work is best understood by separating two aspects of the proposed framework. The first is the assumption of a product-grid sampling structure, which, as noted in Section 2.2, is well-established in multivariate approximation theory and is not in itself a contribution of this work. The second is how this sampling structure is exploited inside the SVM cost function: rather than imposing a low-rank tensor decomposition on the input data, as in conventional support tensor machines (Chen et al., 2022; Kotsia & Patras, 2011; Kour et al., 2023; L. Ma et al., 2017), the proposed framework imposes a PARAFAC or Tucker decomposition on the slack tensor $\mathcal{E}$ of the SVM, while keeping each input sample as a vector and the separating hyperplane linear in the original feature space.

The motivation for imposing a low-rank structure on $\mathcal{E}$ is that the slack variables inherit the topology of the sampling grid. Two samples whose multi-indices $(p_1, \ldots, p_N)$ differ in a single coordinate by one unit correspond to neighboring points in the input space; whenever the underlying decision function is smooth and the grid resolution is sufficiently fine to resolve its variations, the corresponding slack values vary smoothly across the grid. A PARAFAC or Tucker decomposition with small ranks is consistent with this kind of regularity, acting as an inductive bias on $\mathcal{E}$ adapted to the geometry of the sweep. We note that low-rank tensor structure and grid smoothness are not equivalent in general, and we adopt this perspective as a working motivation for the framework presented in the following sections.

Our goal in this work is to highlight that the sampling process itself carries a multilinear structure that can be directly incorporated into the SVM cost function. The

broader idea of exploiting sampling geometry in supervised learning has antecedents in manifold regularization (Belkin, Niyogi, & Sindhwani, 2006), structured Gaussian processes on grids (Gilboa, Saatçi, & Cunningham, 2015), sparse-grid methods (Bungartz & Griebel, 2004), and, more recently, geometric deep learning (Bronstein, Bruna, Cohen, & Veličković, 2021). The contrast with kernelized tensor SVMs (Chen et al., 2022; Kour et al., 2023) is sharpest in the choice of where multilinearity is exploited: those works use it in the input representation and lift the decision boundary to a reproducing kernel Hilbert space, whereas in our setting the decision boundary remains linear in the original input and the structural information of the sampling acts on the optimization itself. We view the MS framework as one instance within this broader programme, and as a starting point for further developments in which the geometry of the sampling design is treated as a first-class component of the learning algorithm. A systematic empirical comparison against kernelized tensor SVMs is a natural next step within this programme.

3 Tensor Background

In this section, the notation adopted in the present work is introduced, as well as some basic definitions and tensor decompositions.

3.1 Notation

The notation used in the present work is presented as follows. Scalars are denoted by lowercase letters ($a, b, \dots$), vectors as lower-case boldface letters ($\mathbf{a}, \mathbf{b}, \dots$), matrices as upper-case boldface letters ($\mathbf{A}, \mathbf{B}, \dots$) and tensors as calligraphic letters ($\mathcal{A}$,$\mathcal{B}$,$\dots$). The i^{th} element of the vector $\mathbf{a}$ is denoted by a_i or $[\mathbf{a}]_i$, the $(i,j)^{th}$ entry of the matrix $\mathbf{A}$ is denoted by $a_{i,j}$ or $[\mathbf{A}]_{i,j}$, and the $(i_1, \dots, i_N)^{th}$ element of the N^{th} order tensor $\mathcal{A}$ is denoted by $a_{i_1,\dots,i_N}$ or $[\mathcal{A}]_{i_1,\dots,i_N}$.

$\mathbf{I}_N$ is the identity matrix of dimension N, $\mathbf{1}_{N,M}$ is a $N \times M$ matrix composed of 1's and $\mathbf{0}_{N,M}$ is a $N \times M$ matrix composed of 0's. Considering an arbitrary matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$, its transpose, Frobenius norm and j^{th} row are denoted respectively by $\mathbf{A}^T$, $\|\mathbf{A}\|_F$ and $[\mathbf{A}]_{j,:} \in \mathbb{R}^{1 \times J}$. Given two vectors $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^I$, $\langle \mathbf{a}, \mathbf{b} \rangle$ denotes the inner product. Given two matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$, $\mathbf{A} \odot \mathbf{B} \in \mathbb{R}^{IJ \times K}$ denotes the Khatri-Rao product.

A matrix slice of a tensor is a two-dimensional section of the tensor, obtained by fixing all indices but two. For instance, the third-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ has mode-1, mode-2 and mode-3 slice matrices denoted respectively by $[\mathcal{X}]_{i_1,:,:} \in \mathbb{R}^{I_2 \times I_3}$, $[\mathcal{X}]_{:,i_2,:} \in \mathbb{R}^{I_1 \times I_3}$ and $[\mathcal{X}]_{:,:,i_3} \in \mathbb{R}^{I_1 \times I_2}$.

The mode-n unfolded matrix of a N^{th}-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ is a matrix $\mathbf{X}^{[n]} \in \mathbb{R}^{I_1 \dots I_{n-1} I_{n+1} \dots I_N \times I_n}$ whose elements are obtained from the tensor $\mathcal{X}$ in the following way:

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

3.2 Parallel Factor Analysis (PARAFAC)

The PARAFAC decomposition was first presented by Hitchcock (1927) and further developed by Harshman (1970) and Carroll and Chang (1970). It was referred to in Carroll & Chang's work as Canonical Decomposition, abbreviated as CANDECOMP, but it can also be referred to by the acronym CP.

The PARAFAC decomposition of an arbitrary third-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ can be expressed by:

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

where Q is the tensor rank and $a_{i_1,q}^{(1)}, a_{i_2,q}^{(2)}$ and $a_{i_3,q}^{(3)}$ are the factors of the PARAFAC decomposition, which can be organized in the 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 PARAFAC decomposition expresses a tensor as a sum of rank-1 tensors.

3.3 Tucker Decomposition

The Tucker decomposition, proposed by 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. For a third-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the Tucker decomposition can be expressed as:

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

where (Q_1, Q_2, Q_3) is the trilinear rank, $a_{i_1,q_1}^{(1)}$, $a_{i_2,q_2}^{(2)}$ and $a_{i_3,q_3}^{(3)}$ form the matrix factors of the Tucker decomposition $\mathbf{A}^{(1)} \in \mathbb{R}^{I_1 \times Q_1}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{I_2 \times Q_2}$ and $\mathbf{A}^{(3)} \in \mathbb{R}^{I_3 \times Q_3}$, and g_{q_1,q_2,q_3} forms the core tensor $\mathcal{G} \in \mathbb{R}^{Q_1 \times Q_2 \times Q_3}$.

4 Multilinear Sampling in Support Vector Machines

In this section, the MS framework is presented and, then, an adaptation of the primal formulation of the SVM to the MS is presented. Subsequently, two solutions for the proposed formulation are presented.

4.1 Multilinear Sampling Framework

Let us consider a training data set consisting of P input feature vectors denoted by $\mathbf{x}_p \in \mathbb{R}^N$, for $p = 1, \ldots, P$, where N is the number of attributes. The basic assumption of the MS approach is that the P samples follow a multidimensional sampling structure, i.e., the samples are generated from a scan (sweep) of the input variables over a range of values, with each sample computed by changing one element of the input vector at a time.

Let us assume that the training database is constructed by making the n^{th} element of the input vector assume the values $\{\beta_1^{(n)}, \ldots, \beta_{P_n}^{(n)}\}$, for $n = 1, \ldots, N$, with $\beta_{P_n}^{(n)} <$

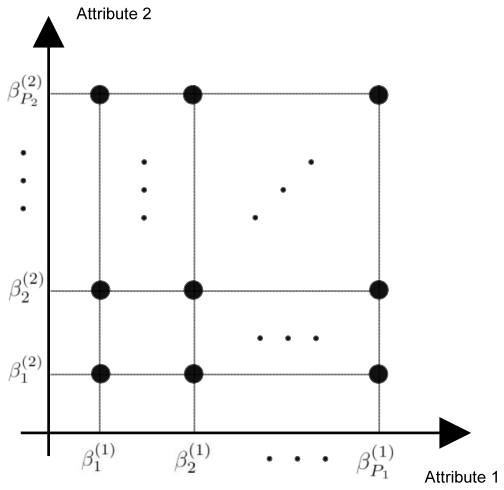


Fig. 1 2D database: sweep of the attributes over a rectangle

$\beta_{P_{n+1}}^{(n)}$, where P_n is the number of values that the n^{th} attribute assumes in the training database. The database is composed of $P = P_1 \cdots P_N$ samples, a generic sample being given by:

$$\mathbf{x}_p = \mathbf{x}_{p_1,\ldots,p_N} = \begin{bmatrix} \beta_{p_1}^{(1)} \\ \vdots \\ \beta_{p_N}^{(N)} \end{bmatrix} \in \mathbb{R}^{N\times 1}. \tag{4}$$

The whole database is composed of all the combinations of $\mathbf{x}_{p_1,\ldots,p_N}$ in (4) for $p_1 = 1,\ldots,P_1,\ \ldots,\ p_N = 1,\ldots,P_N$. Fig. 1 illustrates the construction of the input vectors for $N = 2$. This figure shows a 2D grid of attributes and a database consisting of a sweep of the input variables over a rectangle.

For instance, let us consider the case with $N = 2$, where the first and second elements of the input vector assume, respectively, the values $\{1, 2\}$ and $\{-1, 0, 1\}$ in the training database. In this case, $P_1 = 2$, $P_2 = 3$, $\beta_1^{(1)} = 1$, $\beta_2^{(1)} = 2$, $\beta_1^{(2)} = -1$, $\beta_2^{(2)} = 0$ and $\beta_3^{(2)} = 1$. The whole input database can be organized in the 3^{rd} order tensor $\mathcal{X} \in \mathbb{R}^{2\times 2\times 3}$ whose mode-2 matrix slices are given by:

$$[\mathcal{X}]_{:,1,:} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 3}, \tag{5}$$

$$[\mathcal{X}]_{:,2,:} = \begin{bmatrix} 2 & 2 & 2 \\ -1 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 3}. \tag{6}$$

In the general case, the input database can be organized in a $(N+1)^{th}$ order tensor $\mathcal{X} \in \mathbb{R}^{N\times P_1\times\cdots\times P_N}$, with the last N dimensions corresponding to sampling dimensions denoted by $P_1,\ldots,P_N$, where $P = P_1 \cdots P_N$. The indices associated with the N sampling dimensions are denoted by $p_1,\ldots,p_N$, where $p = p_N + (p_{N-1} -$

$1)P_N + \cdots + (p_1 - 1)P_N \cdots P_2$, for $1 \leq p_n \leq P_n$ and $n = 1, \ldots, N$. In other words, index p corresponds to the concatenation of the indices $p_1, \ldots, p_N$. The output data (classes) is assumed to follow a similar structure, being organized in a N^{th} order tensor $\mathcal{Y} \in \mathbb{R}^{P_1 \times \cdots \times P_N}$.

The above structure for the multidimensional sampling occurs in a broad scope of applications, such as photonic structures (W. Ma et al., 2021; Maia et al., 2025) and optical devices (Fernandes et al., 2022; Martins et al., 2018; Xie et al., 2020). In particular, when ML methods model dynamic systems, the training database is usually built by sweeping each input variable over a range while recording the corresponding outputs. This sweep typically follows the MS structure in both direct and reverse design.

The MS exploits the multidimensional structure of the samples. This structure relies on the way the samples were obtained and not on the structure of the input vector, since each input sample is itself a vector. In a classical ML approach, the input data would be organized in a matrix $\mathbf{X} \in \mathbb{R}^{N \times P}$ and the output in a vector $\mathbf{y} \in \mathbb{R}^{P}$; i.e., all the samples would be concatenated into a single sampling index p, breaking the multidimensional structure of the samples and discarding the structural information of the sweep.

In the following section, the MS 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. In particular, the primal cost function of the SVM is modified to cope with the multiple indices of the MS framework, aiming to exploit the multilinear structure of the sampling.

4.2 Primal Formulation of the SVM with MS

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

$$\min_{\mathbf{w}, b, \xi_1, \ldots, \xi_P} \quad \langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{p=1}^{P} \xi_p, \tag{7}$$

subject to

$$y_p \left(\langle \mathbf{w}, \mathbf{x}_p \rangle + b \right) \geq 1 - \xi_p, \tag{8}$$

$$\xi_p \geq 0, \tag{9}$$

for $p = 1, \ldots, P$, where $\mathbf{w} \in \mathbb{R}^N$ is the weight vector, C is the relaxing constant, ξ_p is a slack variable, y_p is the binary output, and b is the bias term.

Considering the data samples follow the earlier described MS, the SVM problem can be stated as:

$$\min_{\mathbf{w}, b, \mathcal{E}} \quad \langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{p_1=1}^{P_1} \cdots \sum_{p_N=1}^{P_N} \xi_{p_1, \ldots, p_N}, \tag{10}$$

subject to

$$y_{p_1, \ldots, p_N} \left(\langle \mathbf{w}, \mathbf{x}_{p_1, \ldots, p_N} \rangle + b \right) \geq 1 - \xi_{p_1, \ldots, p_N}, \tag{11}$$

$$\xi_{p_1,\ldots,p_N} \geq 0, \tag{12}$$

for $p_1 = 1, \ldots, P_1, \ldots, p_N = 1, \ldots, P_N$, where $\mathcal{E} \in \mathbb{R}^{P_1 \times \cdots \times P_N}$ is a N^{th} order tensor that contains the slack variables. The only difference between these two SVM formulations is that the index p in (7)-(9) was replaced by the multiple sampling indices $p_1, \ldots, p_N$ in (10)-(12). In the next two sections, two methods for solving the formulation (10)-(12) are proposed, considering PARAFAC and Tucker decompositions for the slack tensor $\mathcal{E}$.

5 SVM with MS using PARAFAC (SVM-MSP)

In this section, a solution for the problem formulated in (10)-(12) is presented by considering a PARAFAC decomposition for the tensor $\mathcal{E}$. Subsequently, the problem is expressed as a Quadratic Programming (QP) problem and, then, the algorithm of the SVM classifier with MS using the PARAFAC decomposition (SVM-MSP) is presented.

5.1 Problem Formulation

A solution to (10)-(12) can be obtained by considering a PARAFAC decomposition for the slack tensor $\mathcal{E}$ and using multiple conventional SVMs. The PARAFAC decomposition of slack tensor $\mathcal{E}$ can be expressed as:

$$\xi_{p_1,\ldots,p_N} = \sum_{q=1}^{Q} a_{p_1,q}^{(1)} \cdots a_{p_N,q}^{(N)}, \tag{13}$$

where Q is the tensor rank and $a_{p_1,q}^{(1)}, \ldots, a_{p_N,q}^{(N)}$ 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 (13), eqs. (10)-(12) can be rewritten as:

$$\min_{\mathbf{w},b,\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}} \quad \langle \mathbf{w}, \mathbf{w} \rangle + C \left(\sum_{p_1=1}^{P_1} \cdots \sum_{p_N=1}^{P_N} \sum_{q=1}^{Q} a_{p_1,q}^{(1)} \cdots a_{p_N,q}^{(N)} \right), \tag{14}$$

subject to

$$y_{p_1,\ldots,p_N} \left(\langle \mathbf{w}, \mathbf{x}_{p_1,\ldots,p_N} \rangle + b \right) \geq 1 - \sum_{q=1}^{Q} a_{p_1,q}^{(1)} \cdots a_{p_N,q}^{(N)}, \tag{15}$$

$$\sum_{q=1}^{Q} a_{p_1,q}^{(1)} \cdots a_{p_N,q}^{(N)} \geq 0, \tag{16}$$

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

The proposed classifier is an iterative algorithm with multiple alternated steps at each iteration, similarly to the Alternating Least Squares (ALS) algorithm (Comon,

Luciani, & de Almeida, 2009). At each step of each iteration, one of the factor matrices of the slack tensor $\mathcal{E}$ 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 re-estimated in each step of each iteration.

Assuming that all the factor matrices are known, except for $\mathbf{A}^{(n)}$, eqs. (14)-(16) can be rewritten as follows:

$$\min_{\mathbf{w},b,\mathbf{A}^{(n)}} \quad \langle \mathbf{w}, \mathbf{w} \rangle + C \left(\sum_{q=1}^{Q} C_q^{[n]} \sum_{p_n=1}^{P_n} a_{p_n,q}^{(n)} \right), \tag{17}$$

subject to

$$y_{p_1,\ldots,p_N} \left(\langle \mathbf{w}, \mathbf{x}_{p_1,\ldots,p_N} \rangle + b \right) \geq 1 - \sum_{q=1}^{Q} a_{p_n,q}^{(n)} u_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,q}^{[n]}, \tag{18}$$

$$\sum_{q=1}^{Q} a_{p_n,q}^{(n)} u_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,q}^{[n]} \geq 0, \tag{19}$$

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

$$C_q^{[n]} = \sum_{p_1=1}^{P_1} \cdots \sum_{p_{n-1}=1}^{P_{n-1}} \sum_{p_{n+1}=1}^{P_{n+1}} \cdots \sum_{p_N=1}^{P_N} a_{p_1,q}^{(1)} \cdots a_{p_{n-1},q}^{(n-1)} a_{p_{n+1},q}^{(n+1)} \cdots a_{p_N,q}^{(N)} \tag{20}$$

and

$$u_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,q}^{[n]} = a_{p_1,q}^{(1)} \cdots a_{p_{n-1},q}^{(n-1)} a_{p_{n+1},q}^{(n+1)} \cdots a_{p_N,q}^{(N)} \tag{21}$$

are assumed to be known. Despite being different from the standard SVM formulation (7)-(9), the optimization problem in (17)-(19) can also be expressed as QP, as it will be shown in the next subsection.

5.2 Quadratic Programming Formulation

QP is a class of nonlinear programming for solving optimization problems with quadratic objective subject to linear constraints. Many well-established QP methods exist, including interior-point, active-set, augmented Lagrangian, and conjugate-gradient methods (Nocedal & Wright, 1999). A general formulation for a QP is given by:

$$\min_{\mathbf{x}} \quad \mathbf{x}^T \mathbf{P} \mathbf{x} + \mathbf{x}^T \mathbf{q}, \tag{22}$$

subject to

$$\mathbf{R}\mathbf{x} \leq \mathbf{s}. \tag{23}$$

The cost function (17) can be expressed in the general form (22) with the following correspondences:

$$\mathbf{x} = \left[\mathbf{w}^T \; b \; a_{1,1}^{(n)} \cdots a_{1,Q}^{(n)} \cdots a_{P_n,1}^{(n)} \cdots a_{P_n,Q}^{(n)} \right]^T \in \mathbb{R}^{(N+P_nQ+1)\times 1}, \tag{24}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_N & \mathbf{0}_{N,(P_nQ+1)} \\ \mathbf{0}_{(P_nQ+1),N} & \mathbf{0}_{(P_nQ+1),(P_nQ+1)} \end{bmatrix} \in \mathbb{R}^{(N+P_nQ+1)\times(N+P_nQ+1)}, \tag{25}$$

$$\mathbf{q} = \begin{bmatrix} \mathbf{0}_{(N+1),1} \\ \\ \mathbf{1}_{P_n,1} \odot \mathbf{c}^{[n]} \end{bmatrix} \in \mathbb{R}^{(N+P_nQ+1)\times 1}, \tag{26}$$

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

Moreover, the constraints (18) and (19) can be expressed as in the general form (23) with the following correspondences:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}^{(1)} \\ \\ \mathbf{R}^{(2)} \end{bmatrix} \in \mathbb{R}^{2P\times(N+P_nQ+1)}, \tag{27}$$

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}^{(1)} \\ \\ \mathbf{s}^{(2)} \end{bmatrix} \in \mathbb{R}^{2P\times 1}, \tag{28}$$

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 $p = 1,\ldots,P$, is given by:

$$[\mathbf{R}^{(1)}]_{p,:} = -\begin{bmatrix} y_{p_1,\ldots,p_N}\mathbf{x}_{p_1,\ldots,p_N} \\ y_{p_1,\ldots,p_N} \\ \mathbf{0}_{Q(p_n-1),1} \\ u^{[n]}_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,1} \\ \vdots \\ u^{[n]}_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,Q} \\ \mathbf{0}_{Q(P_n-p_n),1} \end{bmatrix}^T \in \mathbb{R}^{1\times(N+P_nQ+1)}, \tag{29}$$

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

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 $p = 1,\ldots,P$ and $P = P_1\cdots P_N$, is given by:

$$[\mathbf{R}^{(2)}]_{p,:} = -\begin{bmatrix} \mathbf{0}_{N+1+Q(p_n-1),1} \\ u^{[n]}_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,1} \\ \vdots \\ u^{[n]}_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,Q} \\ \mathbf{0}_{Q(P_n-p_n),1} \end{bmatrix}^T, \tag{30}$$

with p corresponding to the concatenation of the indices $p_1,\ldots,p_N$, as in (29). Once the problem (17)-(19) of the SVM-MSP can be expressed as QP, it can be solved by the earlier mentioned QP methods.

Algorithm 1 SVM-MSP Estimation Algorithm

Inputs: ϵ, C, $\mathcal{X}$, $\mathcal{Y}$
Outputs: $\hat{\mathbf{w}}, \hat{b}, \hat{\mathcal{E}}$
Initialization: $\beta = 0$, $i = 0$, $\hat{\mathbf{A}}_0^{(2)}, \ldots, \hat{\mathbf{A}}_0^{(N)}$ randomly
while $\beta = 0$ **do**
    $i = i + 1$
    for $n = 1 : N$ **do**
        Obtain $\mathbf{P}$, $\mathbf{q}$, $\mathbf{R}$, $\mathbf{s}$ from (25)-(30) using $\hat{\mathbf{A}}_i^{(1)}, \ldots, \hat{\mathbf{A}}_i^{(n-1)}, \hat{\mathbf{A}}_{i-1}^{(n+1)}, \ldots, \hat{\mathbf{A}}_{i-1}^{(N)}$.
        Find $\hat{\mathbf{w}}_i, \hat{b}_i, \hat{\mathbf{A}}_i^{(n)}$ using any QP method.
    end for
    Build $\hat{\mathcal{E}}_i$ using $\hat{\mathbf{A}}_i^{(1)}, \ldots, \hat{\mathbf{A}}_i^{(N)}$ from (13).
    Unfold $\hat{\mathcal{E}}_i$ into $\hat{\mathbf{E}}_i^{[1]}$ using (1).
    if $|e_i| < \epsilon$ **then**
        $\beta = 1$
        $\hat{\mathbf{w}} = \hat{\mathbf{w}}_i, \hat{b} = \hat{b}_i, \hat{\mathcal{E}} = \hat{\mathcal{E}}_i$
    end if
end while

5.3 Estimation Algorithm

The estimation algorithm of the SVM-MSP is shown in Algorithm 1. 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 $\mathcal{E}$ 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 (25)-(30). 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 $\hat{\mathcal{E}}_i$ is constructed and unfolded, and a test for convergence is performed using the binary flag variable β, as follows:

$$e_i = \frac{\|\hat{\mathbf{E}}_i^{[1]} - \hat{\mathbf{E}}_{i-1}^{[1]}\|_F^2}{\|\hat{\mathbf{E}}_{i-1}^{[1]}\|_F^2}, \tag{31}$$

where ϵ is a small scalar constant. 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.

6 SVM with MS using Tucker (SVM-MST)

In this section, a solution for the problem formulated in (10)-(12), denoted by SVM-MST, is presented by considering a Tucker decomposition for the slack tensor $\mathcal{E}$, since the Tucker decomposition is more generic and flexible than the PARAFAC decomposition.

6.1 Problem Formulation

Let us assume that the N-th order slack tensor $\mathcal{E} \in \mathbb{R}^{P_1 \times \cdots \times P_N}$ follows a Tucker decomposition:

$$\xi_{p_1,\ldots,p_N} = \sum_{q_1=1}^{Q_1} \cdots \sum_{q_N=1}^{Q_N} g_{q_1,\ldots,q_N} a^{(1)}_{p_1,q_1} \cdots a^{(N)}_{p_N,q_N}, \tag{32}$$

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

The optimization problem of the SVM may be restated as:

$$\min_{\mathbf{w},b,\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)},\mathcal{G}} \quad \langle \mathbf{w}, \mathbf{w} \rangle + C \left(\sum_{p_1=1}^{P_1} \cdots \sum_{p_N=1}^{P_N} \sum_{q_1=1}^{Q_1} \cdots \sum_{q_N=1}^{Q_N} g_{q_1,\ldots,q_N} a^{(1)}_{p_1,q_1} \cdots a^{(N)}_{p_N,q_N} \right), \tag{33}$$

subject to

$$y_{p_1,\ldots,p_N} \left(\langle \mathbf{w}, \mathbf{x}_{p_1,\ldots,p_N} \rangle + b \right) \geq 1 - \sum_{q_1=1}^{Q_1} \cdots \sum_{q_N=1}^{Q_N} g_{q_1,\ldots,q_N} a^{(1)}_{p_1,q_1} \cdots a^{(N)}_{p_N,q_N}, \tag{34}$$

$$\sum_{q_1=1}^{Q_1} \cdots \sum_{q_N=1}^{Q_N} g_{q_1,\ldots,q_N} a^{(1)}_{p_1,q_1} \cdots a^{(N)}_{p_N,q_N} \geq 0, \tag{35}$$

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

The variables $a^{(1)}_{p_1,q_1}$, ..., $a^{(N)}_{p_N,q_N}$ and $g_{q_1,\ldots,q_N}$ are estimated iteratively and alternately. At each iteration, each of these Tucker factors is estimated assuming that the other factors are known, using the previous estimates of these factors, as in the SVM-MSP Algorithm.

Assuming that the core tensor $\mathcal{G}$ and all the factor matrices are known, except for $\mathbf{A}^{(n)}$, eqs. (33)-(35) can be re-expressed as in (17)-(18), substituting Q by Q_n and q by q_n, with $C^{[n]}_{q_n}$ and $u^{[n]}_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,q_n}$ given by:

$$\begin{aligned} C^{[n]}_{q_n} = \sum_{p_1=1}^{P_1} \cdots \sum_{p_{n-1}=1}^{P_{n-1}} \sum_{p_{n+1}=1}^{P_{n+1}} \cdots \sum_{p_N=1}^{P_N} \sum_{q_1=1}^{Q_1} \cdots \sum_{q_{n-1}=1}^{Q_{n-1}} \sum_{q_{n+1}=1}^{Q_{n+1}} \cdots \sum_{q_N=1}^{Q_N} g_{q_1,\ldots,q_N} \times \\ \times \; a^{(1)}_{p_1,q_1} \cdots a^{(n-1)}_{p_{n-1},q_{n-1}} a^{(n+1)}_{p_{n+1},q_{n+1}} \cdots a^{(N)}_{p_N,q_N}, \end{aligned} \tag{36}$$

$$u^{[n]}_{p_1,\ldots,p_{n-1},p_{n+1},\ldots,p_N,q_n} = \sum_{q_1=1}^{Q_1} \cdots \sum_{q_{n-1}=1}^{Q_{n-1}} \sum_{q_{n+1}=1}^{Q_{n+1}} \cdots \sum_{q_N=1}^{Q_N} g_{q_1,\ldots,q_N} \times \\ \times\ a^{(1)}_{p_1,q_1} \cdots a^{(n-1)}_{p_{n-1},q_{n-1}} a^{(n+1)}_{p_{n+1},q_{n+1}} \cdots a^{(N)}_{p_N,q_N}. \tag{37}$$

One additional step is necessary for estimating $g_{q_1,\ldots,q_N}$ assuming that $a^{(1)}_{p_1,q_1}, \ldots, a^{(N)}_{p_N,q_N}$ are known. Let us consider that the term $C^{[g]}_{q_1,\ldots,q_N}$ is known and given by:

$$C^{[g]}_{q_1,\ldots,q_N} = \sum_{p_1=1}^{P_1} \cdots \sum_{p_N=1}^{P_N} a^{(1)}_{p_1,q_1} \cdots a^{(N)}_{p_N,q_N}. \tag{38}$$

Eq. (33) can be rewritten as follows:

$$\min_{\mathbf{w},b,\mathcal{G}} \quad \langle \mathbf{w}, \mathbf{w} \rangle + C \left(\sum_{q_1=1}^{Q_1} \cdots \sum_{q_N=1}^{Q_N} C^{[g]}_{q_1,\ldots,q_N} g_{q_1,\ldots,q_N} \right), \tag{39}$$

subject to

$$y_{p_1,\ldots,p_N} \left(\langle \mathbf{w}, \mathbf{x}_{p_1,\ldots,p_N} \rangle + b \right) \geq 1 - \sum_{q_1=1}^{Q_1} \cdots \sum_{q_N=1}^{Q_N} g_{q_1,\ldots,q_N} u^{[g]}_{p_1,\ldots,p_N,q_1,\ldots,q_N}, \tag{40}$$

$$\sum_{q_1=1}^{Q_1} \cdots \sum_{q_N=1}^{Q_N} g_{q_1,\ldots,q_N} u^{[g]}_{p_1,\ldots,p_N,q_1,\ldots,q_N} \geq 0, \tag{41}$$

with $p_1 = 1,\ldots,P_1, \ldots, p_N = 1,\ldots,P_N$ and $u^{[g]}_{p_1,\ldots,p_N,q_1,\ldots,q_N}$ is given by:

$$u^{[g]}_{p_1,\ldots,p_N,q_1,\ldots,q_N} = a^{(1)}_{p_1,q_1} \cdots a^{(N)}_{p_N,q_N}. \tag{42}$$

6.2 QP formulation for the Tucker case

The QP formulation for the Tucker case is similar to the PARAFAC case. In fact, (24)-(30) can be used in the Tucker case, substituting Q by the respective Q_n. However, one extra step is performed to estimate the core tensor $\mathcal{G}$. In this extra step, the input vector is defined as:

$$\mathbf{x} = \left[\mathbf{w}^T \ b \ g_{1,\ldots,1} \cdots g_{Q_1,\ldots,Q_N} \right]^T \in \mathbb{R}^{(N+\prod_{n=1}^{N} Q_n + 1) \times 1}. \tag{43}$$

Without loss of generalization, it is assumed that, in (43), the index q_N runs faster, while the index q_1 runs slower.

The matrix $\mathbf{P}$ is built as in (25), substituting $P_n Q$ by $\prod_{n=1}^{N} Q_n$, whereas the vector $\mathbf{q}$ is given by:

$$\mathbf{q} = \begin{bmatrix} \mathbf{0}_{(N+1),1} \\ \\ \mathbf{c}^{[g]} \end{bmatrix} \in \mathbb{R}^{(N+\prod_{n=1}^{N} Q_n + 1) \times 1}, \tag{44}$$

where $\mathbf{c}^{[g]} = C[C^{[g]}_{1,\ldots,1} \cdots C^{[g]}_{Q_1,\ldots,Q_N}]^T \in \mathbb{R}^{\prod_{n=1}^{N} Q_n \times 1}$.

Moreover, the constraints (40) and (41) can be expressed as in (23), with $\mathbf{R}$ and $\mathbf{s}$ defined in (27) and (28), where $\mathbf{s}^{(1)} = -\mathbf{1}_{P,1} \in \mathbb{R}^{P\times 1}$, $\mathbf{s}^{(2)} = \mathbf{0}_{P,1} \in \mathbb{R}^{P\times 1}$ and the p^{th} row of $\mathbf{R}^{(1)} \in \mathbb{R}^{P\times(N+\prod_{n=1}^{N} Q_n+1)}$ and $\mathbf{R}^{(2)} \in \mathbb{R}^{P\times(N+\prod_{n=1}^{N} Q_n+1)}$, for $p = 1, \ldots, P$, are given respectively by:

$$[\mathbf{R}^{(1)}]_{p,:} = -\begin{bmatrix} y_{p_1,\ldots,p_N}\mathbf{x}_{p_1,\ldots,p_N} \\ y_{p_1,\ldots,p_N} \\ u^{[g]}_{p_1,\ldots,p_N,1,\ldots,1} \\ \vdots \\ u^{[g]}_{p_1,\ldots,p_N,Q_1,\ldots,Q_N} \end{bmatrix}, \tag{45}$$

$$[\mathbf{R}^{(2)}]_{p,:} = -\begin{bmatrix} \mathbf{0}_{N+1,1} \\ u^{[g]}_{p_1,\ldots,p_N,1,\ldots,1} \\ \vdots \\ u^{[g]}_{p_1,\ldots,p_N,Q_1,\ldots,Q_N} \end{bmatrix}. \tag{46}$$

6.3 Estimation Algorithm

The estimation algorithm of the SVM-MST is shown in Algorithm 2. The main difference between the two proposed methods is the extra steps needed to calculate the core matrix in the SVM-MST and the way the variables $\mathbf{P}$, $\mathbf{q}$, $\mathbf{R}$ and $\mathbf{s}$ are constructed. In Algorithm 2, the extra steps are performed after the estimation of the matrices $\hat{\mathbf{A}}_i^{(1)}, \ldots, \hat{\mathbf{A}}_i^{(N)}$.

Table 1 shows the number of parameters to be estimated by the two proposed methods. For comparison purposes, the number of parameters to be estimated by the conventional (vector-based) SVM technique is also reported.

Table 1 Number of parameters to be estimated

Model	N. parameters
SVM (vector-based)	$P_1 \cdots P_N + N + 1$
SVM-MSP	$Q(P_1 + \cdots + P_N) + N + 1$
SVM-MST	$P_1Q_1 + \cdots + P_NQ_N + Q_1 \cdots Q_N + N + 1$

7 Database Descriptions

7.1 MZI Database

All-optical devices, such as optical sensors, amplifiers and modulators, are of great importance in communications and computer systems. In particular, the design of

Algorithm 2 SVM-MST Estimation Algorithm

Inputs: ϵ, C, $\mathcal{X}$, $\mathcal{Y}$
Outputs: $\hat{\mathbf{w}}, \hat{b}, \hat{\mathcal{E}}$
Initialization: $\beta = 0$, $i = 0$, $\hat{\mathbf{A}}_0^{(2)}, \ldots, \hat{\mathbf{A}}_0^{(N)}$, $\hat{\mathcal{G}}_0 \rightarrow$ random
while $\beta = 0$ **do**
    $i = i + 1$
    for $n = 1 : N$ **do**
        Obtain $\mathbf{P}$, $\mathbf{q}$, $\mathbf{R}$, $\mathbf{s}$ from (25)-(30) using $\hat{\mathbf{A}}_i^{(1)}$, $\ldots$, $\hat{\mathbf{A}}_i^{(n-1)}$, $\hat{\mathbf{A}}_{i-1}^{(n+1)}$, $\ldots$, $\hat{\mathbf{A}}_{i-1}^{(N)}$, $\hat{\mathcal{G}}_{i-1}$.
        Find $\hat{\mathbf{w}}_i, \hat{b}_i, \hat{\mathbf{A}}_i^{(n)}$ using any QP method.
    end for
    Obtain $\mathbf{P}$, $\mathbf{q}$, $\mathbf{R}$, $\mathbf{s}$ from (25) and (44)-(46) using $\hat{\mathbf{A}}_i^{(1)}$, $\ldots$, $\hat{\mathbf{A}}_i^{(N)}$.
    Find $\hat{\mathbf{w}}_i, \hat{b}_i, \hat{\mathcal{G}}_i$ using any QP method.
    Build $\hat{\mathcal{E}}_i$ using $\hat{\mathbf{A}}_i^{(1)}, \ldots, \hat{\mathbf{A}}_i^{(N)}, \hat{\mathcal{G}}_i$ from (32).
    Unfold $\hat{\mathcal{E}}_i$ into $\hat{\mathbf{E}}_i^{[1]}$ using (1).
    if $|e_i| < \epsilon$ **then**
        $\beta = 1$
        $\hat{\mathbf{w}} = \hat{\mathbf{w}}_i, \hat{b} = \hat{b}_i, \hat{\mathcal{E}} = \hat{\mathcal{E}}_i$
    end if
end while

logic gates based on all-optical devices, such as optical couplers, resonators and interferometers, has attracted the attention of many scientists (Fernandes et al., 2022; Martins et al., 2018). Indeed, optical logic gates are essential in the development of all-optical communication and signal processing networks that avoid optoelectronic conversions to obtain high-speed data. In many cases, the output of these photonic devices undergoes a binarization process to be used as logic levels.

In this context, the use of ML methods to model photonic devices or structures has been of great interest in the past years (Fernandes et al., 2022; W. Ma et al., 2021; Yeung et al., 2021). In particular, one of the devices that has been exploited for the design of logic gates is a fiber-optic interferometer known as MZI. In this work, the database used to test the performance of the proposed techniques is obtained from an MZI, whose simplified block scheme is shown in Fig. 2. The MZI performs power switching by means of two optical couplers. The first coupler splits the input signal into two parts, which are recombined later, at the second coupler, and the interference of the propagating signals in both arms determines the signals at the output of the MZI.

Phase shifters with phase deviations ϕ_1 and ϕ_2 are placed along one of the arms to control the two outputs. In addition to the phase shifts, other linear and nonlinear propagation effects are considered: second- and third-order dispersion, path attenuation, and nonlinear phase shifts due to self-phase modulation. The mode coupling and the resulting power switching between the guides of the coupler are also modeled. See Agrawal (2008); Fernandes et al. (2022) for more details on the propagation effects and the database, respectively.

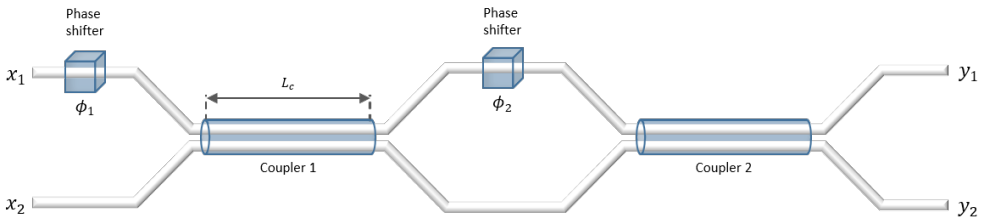


Fig. 2 Simplified block scheme of the MZI

The output of the MZI described above is obtained by solving the coupled-mode equations for each optical coupler, derived from the nonlinear Schrödinger equation (NLSE) (Agrawal, 2008), after adding the phase shifts ϕ_1 and ϕ_2 to the input signals before each coupler. Since these equations have no analytical solution except in specific cases, a numerical treatment was carried out. Runge-Kutta (RK) based methods are frequently used by researchers (Araújo et al., 2015; Bezerra Filho et al., 2017). The database was then constructed by solving the nonlinear Schrödinger equations numerically with the 4th-order RK method together with the Fourier transform (Bezerra Filho et al., 2017).

Table 2 shows the configuration used to simulate the MZI and construct the database. Due to the high number of parameters and the range of their possible values, we sweep some parameters over a prescribed range while keeping the others fixed. The parameters chosen to vary are the phase shifts (ϕ_1, ϕ_2) and the binary inputs, denoted I_1 and I_2. The number of tested values of ϕ_1 and ϕ_2 in the range 0 to 2π is set to $M = 40$, i.e., $\phi_1, \phi_2 \in \{2\pi i/40\}$ for $i = 0, \ldots, 39$. With binary pulse amplitude modulation (PAM-2), all input combinations (00, 01, 10, and 11) were tested. Hence, the optical database has $P = 6400$ samples ($6400 = 2 \cdot 2 \cdot 40 \cdot 40$) and the input vector has four attributes: I_1, I_2, ϕ_1, and ϕ_2.

Table 2 Configuration used to generate the MZI dataset. *Increment* is the step size when sweeping each variable.

Parameter	Range of Values	Increment	N. of values
Modulation	PAM-2	–	1
Input power	4.56 kW	–	1
Wavelength	1.55 μm	–	1
Coupler length	1.8 cm	–	1
Attenuation	0 dB	–	1
Input I_1	0 – 1	1	2
Input I_2	0 – 1	1	2
ϕ_1	0 – 2π rad	0.05π rad	40
ϕ_2	0 – 2π rad	0.05π rad	40

Regarding the output signals of the MZI, either of the two outputs can be used in the numerical simulations. In our study, output 2 was chosen because it presented slightly better results. The outputs of the MZI underwent a binarization process before entering the classifier. The logic output is set to 1 if the output is higher than zero; otherwise, the logic output is set to 0.

In brief, the dataset consists of I_1, I_2, ϕ_1 and ϕ_2 as attributes and the binarized version of y_2 as the output. Following the MS approach, the input data were organized in a 5^{th} order tensor $\mathcal{X} \in \mathbb{R}^{4\times2\times2\times40\times40}$, whereas the outputs form a 4^{th} order tensor $\mathcal{Y} \in \mathbb{R}^{2\times2\times40\times40}$, with $N = 4$, $P_1 = P_2 = 2$ and $P_3 = P_4 = 40$. The MZI database is available in the repository MZI.

7.2 Light Scattering in a Gold Nanostructure

Light scattering in metallic nanostructures plays a pivotal role in numerous photonic applications, including optical sensing, plasmonic resonance, and nanoscale light control (H. Zhang et al., 2023). Modeling light scattering in gold nanostructures with ML is essential for many of these applications. Traditional computational methods, while accurate, are time-consuming and resource-intensive, especially for complex geometries. ML offers a powerful alternative by enabling rapid predictions of the scattering behavior, accelerating design and optimization while revealing relationships between structural parameters and light scattering.

The present work uses a dataset generated from numerical simulations of light scattering in gold cylindrical nanostructures, where three physical parameters were systematically varied: radius (r), height (h), and incident wavelength (λ). The data collection followed a structured parametric sweep, where one parameter was varied at a time while the others were held constant, as detailed in Section 4. Table 3 summarizes the simulation setup. This procedure produces a complete set of sample combinations, organized in a fourth-order input tensor $\mathcal{X} \in \mathbb{R}^{3\times11\times21\times201}$, where the first mode corresponds to the three input attributes and the other modes correspond to the sampling modes. The second, third, and fourth modes of $\mathcal{X}$ are the sampling modes associated with the radius, height, and wavelength, respectively. The associated output tensor $\mathcal{Y} \in \mathbb{R}^{11\times21\times201}$ contains binarized light-scattering intensity values for each configuration, obtained by thresholding against the global median. Prior to modeling, the data were normalized using min-max scaling, mapping all values to the range $[0, 1]$. The gold nanostructure dataset is available in the repository Gold.

Table 3 Configuration used to generate the gold cylindrical nanostructure dataset. *Increment* is the step size when sweeping each variable.

Parameter	Range of Values	Increment	N. of values
Radius (r)	50 nm – 150 nm	10 nm	11
Height (h)	100 nm – 300 nm	10 nm	21
Wavelength (λ)	0.4 μm – 1.2 μm	0.004 μm	201

8 Results

This section evaluates the proposed methods on the two photonic datasets. The experiments are designed to support three claims: (i) preserving the multidimensional sampling structure improves classification accuracy over vector-based baselines, (ii) the Tucker variant (SVM-MST) reaches higher accuracy than the PARAFAC variant (SVM-MSP) at the cost of a higher computational time, and (iii) both proposed algorithms converge within a modest number of iterations. We first report results on the MZI dataset and then on the gold cylindrical nanostructure dataset.

Accuracy, defined as the number of correctly classified samples over the total number of samples, is the main figure of merit. The processing time, in seconds, measures the time complexity of the techniques. Processing times were obtained on a 9th-generation Intel Core i3 processor running MATLAB version 2017b. The number of iterations needed for convergence is also analyzed. The QP problems were solved with the MATLAB *quadprog* function, which uses three algorithms: the Goldfarb-Idnani dual method (Goldfarb & Idnani, 1983), the reflective Newton method of Coleman and Li (1996), and a projection method (Gill, Murray, & Wright, 2019).

8.1 MZI Database

After preliminary tests (omitted to save space), the relaxing constant was set to $C = 100$ and the convergence parameter to $\epsilon = 10^{-6}$. All results are averages over 100 runs with K-fold cross-validation at $K = 10$. The proposed techniques are compared with the conventional SVM classifier and other baselines. When standard vector-based classifiers are used, the inputs are organized in a matrix $\mathbf{X} \in \mathbb{R}^{4\times 6400}$ and the outputs form a vector $\mathbf{y} \in \mathbb{R}^{6400}$.

8.1.1 Impact of the Ranks on the Accuracy

The next results analyze the impact of the multilinear ranks on the accuracy of the proposed classifiers. Fig. 3 shows the accuracy provided by the SVM-MST for several combinations of the multilinear ranks Q_1 and Q_2, as a function of Q_3, with $Q_3 = Q_4$. The highest accuracy is obtained with $Q_1 = Q_2 = 2$, followed by $Q_1 = Q_2 = 3$. The accuracy increases with the ranks Q_3 and Q_4 because larger ranks add degrees of freedom to the model and improve its fitting capacity.

Fig. 4 shows the accuracy of the SVM-MSP versus the tensor rank Q, together with the accuracy of the SVM-MST versus the mode-4 rank Q_4, for $Q_1 = Q_2 = 2$ and several values of the mode-3 rank Q_3. The best accuracy is obtained by the SVM-MST with $Q_3 = Q_4 = 20$, reaching 85.6%, while the highest accuracy of the SVM-MSP is 83.2% at $Q = 15$. The SVM-MST outperforms the SVM-MSP because the Tucker decomposition is more general and flexible than PARAFAC. Unless stated otherwise, the SVM-MST uses $Q_1 = Q_2 = 2$ and $Q_3 = Q_4 = 20$ in the remainder of this section, and the SVM-MSP uses $Q = 15$.

As in Fig. 3, the accuracy initially increases with the ranks Q and Q_4; however, beyond a certain point, the accuracy decreases for most of the curves. This happens because, for very high ranks, the PARAFAC and Tucker decompositions are no longer able to model the slack tensor efficiently, reducing the classifier accuracy.

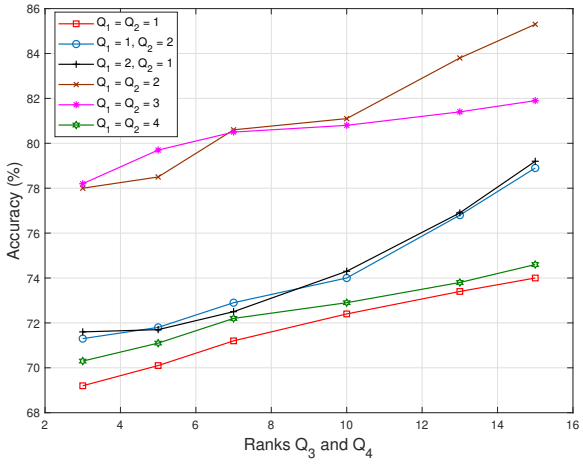


Fig. 3 Accuracy of the SVM-MST on the MZI dataset, varying Q_1 and Q_2 with $Q_3 = Q_4$

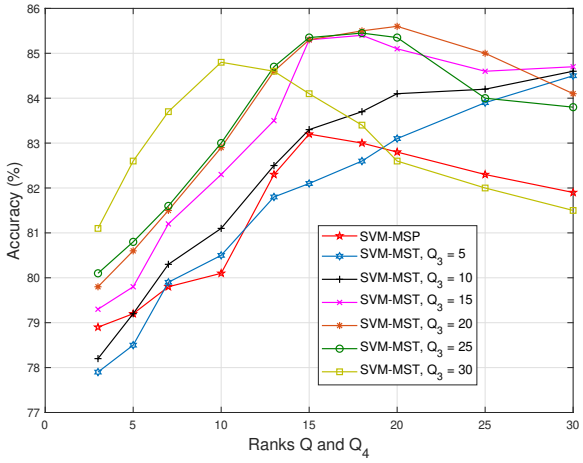


Fig. 4 Accuracy of the SVM-MSP and SVM-MST on the MZI dataset, varying Q and Q_4 with $Q_1 = Q_2 = 2$ and $Q_3 = Q_4$

8.1.2 Comparison with Other Classifiers

The proposed classifiers are compared with the following baselines from the literature: the standard SVM, naive Bayes (Yang, 2018), logistic regression (Nick & Campbell, 2007), k-nearest neighbors (k-NN) (Jiang, Cai, Wang, & Jiang, 2007), and multilayer perceptron (MLP) (Liao & Wen, 2007). We omit preliminary hyperparameter tuning to save space. The k-NN classifier uses $k = 3$, the MLP uses two fully connected layers with rectified linear unit (ReLU) activations and a softmax output, and the logistic regression is fitted linearly without regularization.

Table 4 Accuracy and processing time for several classifiers on the MZI dataset. Best value per column in **bold**; ties are broken by the lowest processing time.

Classifier	Accuracy	Processing Time (s)
SVM-MSP	83.2%	344.6
SVM-MST	**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**
MLP	77.6%	296.6

Table 4 reports the accuracy and processing time of the proposed methods and the baselines. The proposed techniques achieve the highest accuracy among all tested classifiers, with at least 5.6 percentage points of advantage over the strongest baseline (MLP, 77.6%). Preserving the multidimensional sampling structure thus leads to substantial accuracy gains over vector-based baselines.

The proposed techniques also require more processing time than the baselines, since the tensor-based approach used by the SVM-MSP and SVM-MST requires multiple steps to estimate all the factors. The SVM-MSP requires less processing time than the SVM-MST because the Tucker decomposition has more parameters than PARAFAC.

Smaller ranks reduce the processing time of the proposed classifiers, as illustrated in Table 5, which reports accuracy and processing time for several values of Q (for the SVM-MSP) and Q_3, Q_4 (for the SVM-MST), with $Q_1 = Q_2 = 2$ and $Q_3 = Q_4$. Both the SVM-MSP and SVM-MST achieve a substantial reduction in processing time while maintaining accuracy higher than all baselines in Table 4.

Table 5 Accuracy and processing time for several rank values on the MZI dataset. Best value per column in **bold**; ties are broken by the lowest processing time.

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

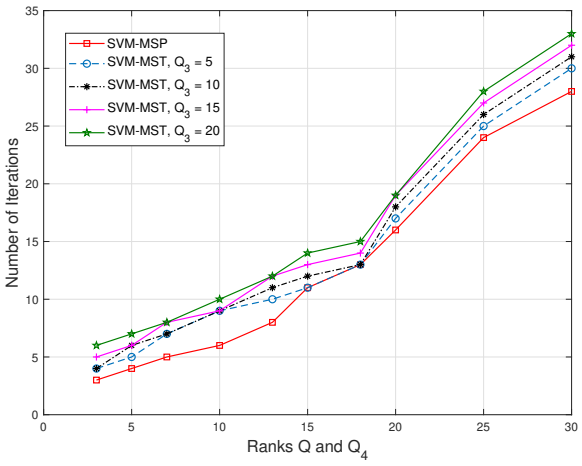


Fig. 5 Iterations needed for convergence versus Q and Q_4 for the MZI dataset

8.1.3 Number of Iterations for Convergence

Fig. 5 shows the number of iterations needed for the SVM-MSP and SVM-MST to converge, as a function of the ranks Q and Q_4, for $Q_1 = Q_2 = 2$ and $Q_3 = Q_4$. The number of iterations increases with Q, Q_3, and Q_4, since these ranks directly affect the number of parameters to be estimated. The maximum number of iterations is approximately 34. In the cases that achieve the highest accuracy ($Q = 15$ for the SVM-MSP and $Q_3 = Q_4 = 20$ for the SVM-MST), roughly 7.5 and 18 iterations are required, respectively. These results show that the proposed algorithms converge within a modest number of iterations.

8.2 Gold Nanostructure Database

Having validated the methods on the MZI dataset, we now turn to the higher-dimensional gold nanostructure problem. The dataset was split into training (80%) and testing (20%) sets. Accuracy and processing time are the main figures of merit, as in the previous analysis. For this dataset, the matrix $\mathbf{P}$ from (25) was regularized to improve stability: $\mathbf{P} \leftarrow \mathbf{P} + \delta \mathbf{I}_{N+P_n Q+1}$, with $\delta = 0.01$.

8.2.1 Impact of the Ranks on the Accuracy

We next analyze the impact of the multilinear ranks on the accuracy of the proposed classifiers. Fig. 6 shows the accuracy of the SVM-MSP and SVM-MST as a function of the ranks Q and Q_3, for $Q_1 = 4$ and $Q_2 = 2$. These values of Q_1 and Q_2 were chosen based on preliminary experiments (omitted for brevity). For the SVM-MSP, the accuracy grows by roughly 7 percentage points as Q increases from 2 to 8, peaking at 85.7% for $Q = 8$ and $Q = 9$, and then slightly decreasing.

For the SVM-MST, the accuracy steadily improves with Q_3, reaching its highest value of 86.3% for $Q_3 \in \{7, 8, 9\}$. As in the MZI dataset, larger ranks add degrees

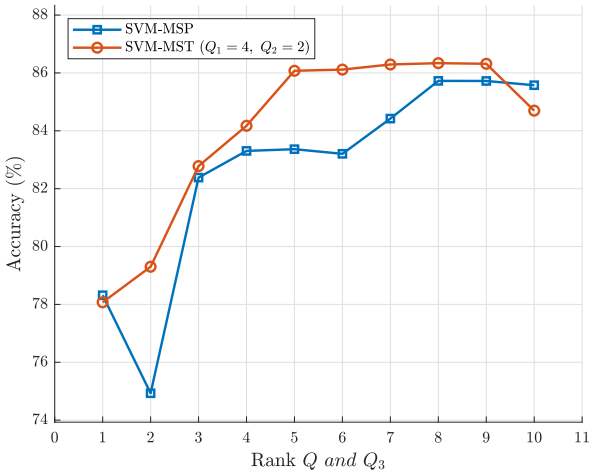


Fig. 6 Accuracy of the SVM-MSP and SVM-MST for the gold nanostructure dataset, varying Q and Q_3 with $Q_1 = 4$ and $Q_2 = 2$

of freedom and improve fitting capacity, but accuracy degrades at very high ranks, suggesting overfitting or loss of decomposition efficiency.

As in the MZI dataset, the SVM-MST reaches slightly higher accuracy than the SVM-MSP across most configurations, owing to the greater flexibility of the Tucker decomposition. In what follows, the SVM-MST uses $Q_1 = 4$, $Q_2 = 2$, $Q_3 = 7$, and the SVM-MSP uses $Q = 8$.

8.2.2 Comparison with Other Classifiers

The proposed classifiers are compared with the same baselines tested in Subsection 8.1.2, after omitted preliminary hyperparameter tuning. Table 6 reports the accuracy and processing time of the proposed classifiers and the baselines. The proposed classifiers achieve higher accuracy than the baselines, supporting the claim that preserving the multidimensional sampling structure benefits classification tasks with complex and structured sampling information.

As observed on the MZI dataset, the proposed methods require more processing time than the baselines, which is expected because tensor-based classifiers involve additional optimization steps. The accuracy gain indicates that preserving the multidimensional structure of the data is worth the additional cost.

Smaller ranks again reduce the processing time of the proposed classifiers. Table 7 reports accuracy and processing time for several rank configurations. The processing time decreases markedly as the ranks decrease, for both classifiers.

8.2.3 Number of Iterations for Convergence

Fig. 7 shows the number of iterations required for convergence. The SVM-MSP requires more iterations, with a peak of approximately 13 at $Q = 5$. The SVM-MST converges

Table 6 Accuracy and processing time for several classifiers on the gold nanostructure dataset. Best value per column in **bold**; ties are broken by the lowest processing time.

Classifier	Accuracy	Processing Time (s)
SVM-MSP	85.7%	17.46
SVM-MST	**86.3**%	24.52
SVM	80.0%	**0.01**
Naive Bayes	59.7%	0.60
Logistic regression	65.6%	0.03
k-NN	85.4%	0.23
MLP	78.9%	0.59

Table 7 Accuracy and processing time for several rank values on the gold nanostructure dataset. Best value per column in **bold**; ties are broken by the lowest processing time.

Ranks	Classifier	Accuracy	Processing Time (s)
Q=1	SVM-MSP	78.3%	**1.00**
Q=2	SVM-MSP	74.9%	2.98
Q=3	SVM-MSP	82.4%	2.30
Q=4	SVM-MSP	83.3%	6.91
Q=5	SVM-MSP	83.4%	22.99
Q=6	SVM-MSP	83.2%	21.72
Q=7	SVM-MSP	84.4%	16.40
Q=8	SVM-MSP	85.7%	17.46
Q=9	SVM-MSP	85.7%	21.95
Q=10	SVM-MSP	85.6%	16.05
Q_1=4, Q_2=2, Q_3=1	SVM-MST	78.1%	7.05
Q_1=4, Q_2=2, Q_3=2	SVM-MST	79.3%	12.02
Q_1=4, Q_2=2, Q_3=3	SVM-MST	82.8%	15.87
Q_1=4, Q_2=2, Q_3=4	SVM-MST	84.2%	13.48
Q_1=4, Q_2=2, Q_3=5	SVM-MST	86.1%	19.06
Q_1=4, Q_2=2, Q_3=6	SVM-MST	86.1%	15.68
Q_1=4, Q_2=2, Q_3=7	SVM-MST	**86.3**%	24.52
Q_1=4, Q_2=2, Q_3=8	SVM-MST	86.3%	30.65
Q_1=4, Q_2=2, Q_3=9	SVM-MST	86.3%	34.87
Q_1=4, Q_2=2, Q_3=10	SVM-MST	84.7%	20.27

with markedly fewer iterations, typically 4 to 5 across different ranks, which indicates an advantage of the SVM-MST over the SVM-MSP in convergence speed. As in the MZI dataset, both proposed algorithms converge within a modest number of iterations.

9 Conclusion and Future Work

We have introduced the multilinear sampling (MS) framework, a perspective on supervised learning in which the multidimensional structure induced by a product-grid

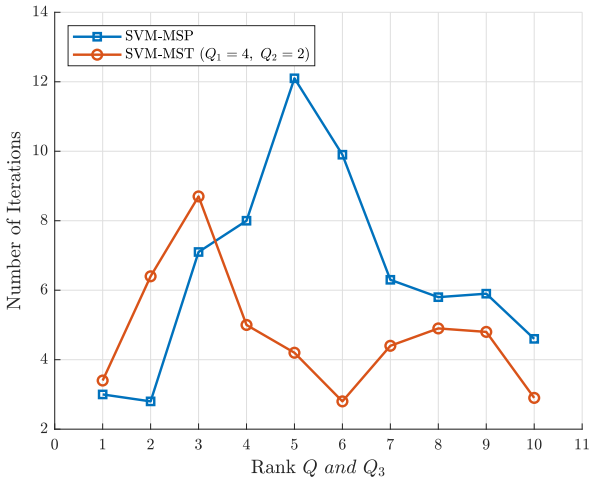


Fig. 7 Iterations needed for convergence versus Q and Q_3 for the gold nanostructure dataset

sampling design is exploited directly inside the cost function of the classifier. Two concrete instances of this perspective were developed for the support vector machine, denoted SVM-MSP and SVM-MST, which encode the slack tensor via PARAFAC and Tucker decompositions respectively, while keeping each input sample as a vector and the separating hyperplane linear in the original input space.

On the MZI dataset (a five-way tensor with dimensions $4 \times 2 \times 2 \times 40 \times 40$), the SVM-MST and SVM-MSP reached 85.6% and 83.2% accuracy respectively, providing a meaningful gain over the standard vector-based baselines we considered (the strongest of which, an MLP, reached 77.6%). On the gold nanostructure dataset (a fourth-order tensor with dimensions $3 \times 11 \times 21 \times 201$), the SVM-MST and SVM-MSP reached 86.3% and 85.7%, providing competitive accuracy that supports the viability of the proposed perspective. Beyond accuracy, the experiments revealed an accuracy/processing-time trade-off, the SVM-MST being more accurate than the SVM-MSP but more expensive, due to the larger number of parameters of the Tucker decomposition. Both algorithms converged within a modest number of iterations (at most ≈ 34 in our experiments).

The current implementation has two main limitations. First, the proposed classifiers cost roughly 2–3$\times$ more processing time than vector-based baselines on the tested datasets. Second, the alternating optimization is sensitive to the random initialization of the factor matrices, which can affect convergence speed.

These results suggest several natural directions for further investigation within the multilinear-sampling research programme: (i) the dual SVM formulation, (ii) regression tasks, (iii) non-uniform and irregular sampling grids, (iv) additional photonic datasets, including plasmonic inverse-design problems, and (v) a systematic empirical comparison with kernelized tensor SVMs, which we anticipated in Section 2 as a natural next step within the broader programme outlined in this work.

Statements and Declarations

- **Funding:** This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq/Brazil) under Grant 306871/2025-4 and Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP/Brazil) under Grant BP5-0197-00183.01.06/23.

- **Competing Interests:** The authors have no competing interests to declare that are relevant to the content of this article.
- **Ethics approval and consent to participate:** Not applicable.
- **Consent for publication:** Not applicable.
- **Data availability:** The MZI dataset is available here. The gold nanostructure dataset is available here.
- **Code availability:** Not applicable.
- **Author contribution:** C. Alexandre R. Fernandes: Conceptualization, Supervision, Funding acquisition, Writing – review & editing. Antonio A. T. Peixoto: Methodology, Formal analysis, Software, Writing – original draft. Dnilson C. de Sousa: Formal analysis, Software, Investigation, Writing – review & editing. Glendo de F. Guimarães: Data curation, Resources. Darlan A. Barroso: Data curation, Resources.

References

Agrawal, G.P. (2008). *Applications of nonlinear fiber optics* (2nd ed.). Elsevier.

Araújo, A., et al. (2015). Two all-optical logic gates in a single photonic interferometer. *Optics Communications*, *355*, 485–491, https://doi.org/10.1016/j.optcom.2015.07.013

Belkin, M., Niyogi, P., Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, *7*, 2399–2434,

Bezerra Filho, G.S., et al. (2017). Obtaining optical logic gates – OR, XOR, AND and logic functions using asymmetric Mach–Zehnder interferometer based on photonic crystal fiber. *Optics & Laser Technology*, *97*, 370–378, https://doi.org/10.1016/j.optlastec.2017.07.030

Bronstein, M.M., Bruna, J., Cohen, T., Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, ,

Bungartz, H.-J., & Griebel, M. (2004). Sparse grids. *Acta Numerica*, *13*, 147–269, https://doi.org/10.1017/S0962492904000182

Carroll, J.D., & Chang, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart–Young" decomposition. *Psychometrika*, *35*(3), 283–319, https://doi.org/10.1007/BF02310791

Chen, C., Batselier, K., Yu, W., Wong, N. (2022). Kernelized support tensor train machines. *Pattern Recognition*, *122*, ,

Cohen, A., DeVore, R., Schwab, C. (2011). Analytic regularity and polynomial approximation of parametric and stochastic elliptic PDEs. *Analysis and Applications*, *9*(1), 11–47, https://doi.org/10.1142/S0219530511001728

Coleman, T.F., & Li, Y. (1996). A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM Journal on Optimization*, *6*(4), 1040–1058, https://doi.org/10.1137/S1052623494240456

Comon, P., Luciani, X., de Almeida, A.L.F. (2009). Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics*, *23*(7-8), 393–405, https://doi.org/10.1002/cem.1236

Deng, X., Shi, Y., Yao, D. (2023). Theories, algorithms and applications in tensor learning. *Applied Intelligence*, *53*(17), 20514–20534, https://doi.org/10.1007/s10489-022-04153-5

de Souza, F.C.N., et al. (2018). Optical current and magnetic field sensor using Mach–Zehnder interferometer with nanoparticles. *IEEE Sensors Journal*, *18*(19), 7998–8004, https://doi.org/10.1109/JSEN.2018.2862393

Dong, Y., An, S., Jiang, H., Zheng, B., Tang, H., Huang, Y., ... Zhang, H. (2025). Advanced deep learning approaches in metasurface modeling and design: A review. *Progress in Quantum Electronics*, *99*, 100554, https://doi.org/10.1016/j.pquantelec.2025.100554

Fernandes, C.A.R., de Oliveira, M.E.N., Rocha, D.S., de Alexandria, A.R., Guimarães, G.F. (2022). Design of optical logic gates using Mach–Zehnder interferometers and machine learning. *Journal of Lightwave Technology*, *40*(18), 6240–6248, https://doi.org/10.1109/JLT.2022.3190140

Gilboa, E., Saatçi, Y., Cunningham, J.P. (2015). Scaling multidimensional inference for structured Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(2), 424–436,

Gill, P.E., Murray, W., Wright, M.H. (2019). *Practical optimization*. Philadelphia: SIAM.

Goldfarb, D., & Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, *27*(1), 1–33, https://doi.org/10.1007/BF02591962

Hackbusch, W. (2012). *Tensor spaces and numerical tensor calculus*. Springer.

Harshman, R.A. (1970). *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-mode factor analysis* (Tech. Rep. No. 16). University of California, Los Angeles. Retrieved from https://www.psychology.uwo.ca/faculty/harshman/wpppfac0.pdf

Hitchcock, F.L. (1927). The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, *6*(1-4), 164–189, https://doi.org/10.1002/sapm192761164

Jiang, L., Cai, Z., Wang, D., Jiang, S. (2007). Survey of improving k-nearest-neighbor for classification. *Fourth international conference on fuzzy systems and knowledge discovery (fskd 2007)* (Vol. 1, pp. 679–683). IEEE.

Jordan, M.I., & Mitchell, T.M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260, https://doi.org/10.1126/science.aaa8415

Kotsia, I., & Patras, I. (2011). Support Tucker machines. *Proceedings of the ieee conference on computer vision and pattern recognition (cvpr)* (pp. 633–640).

Kour, K., Dolgov, S., Stoll, M., Benner, P. (2023). Efficient structure-preserving support tensor train machine. *Journal of Machine Learning Research*, *24*(4), 1–22,

Liao, S.-H., & Wen, C.-H. (2007). Artificial neural networks classification and clustering of methodologies and applications – literature analysis from 1995 to 2005. *Expert Systems with Applications*, *32*(1), 1–11, https://doi.org/10.1016/j.eswa.2005.11.014

Liu, P., Long, H., Zheng, Z., Xiao, L. (2025). Hyperspectral image denoising and destriping via gradient tensor subspace low-rank learning and along-across stripe directional constraints. *IEEE Transactions on Geoscience and Remote Sensing*, *63*, 1–15, https://doi.org/10.1109/TGRS.2025.3583772

Luo, Y., Li, X., Yan, Y., Xia, C. (2023). Spatial-temporal tensor representation learning with priors for infrared small target detection. *IEEE Transactions on Aerospace and Electronic Systems*, *59*(6), 9598–9620, https://doi.org/10.1109/TAES.2023.3320635

Ma, L., Hu, Y., Zhang, Y. (2017). Support Tucker machines based bubble defect detection of lithium-ion polymer cell sheets. *Engineering Letters*, *25*(1), , https://doi.org/10.2174/1872212112666180522082329

Ma, W., et al. (2021). Deep learning for the design of photonic structures. *Nature Photonics*, *15*, 77–90, https://doi.org/10.1038/s41566-020-0685-y

Maia, L.S.P., Barroso, D.A., Silveira, A.B., Oliveira, W.F., Galembeck, A., Fernandes, C.A.R., ... Guimarães, G.F. (2025). Inverse design of plasmonic nanostructures using machine learning for optimized prediction of physical parameters. *Photonics*, *12*, 572, https://doi.org/10.3390/photonics12060572

Martins, F.L.B., et al. (2018). Two and three-input all-optical logic gates on a planar three-core photonic crystal fiber. *Optik*, *154*, 516–523, https://doi.org/10.1016/j.ijleo.2017.10.067

Nick, T.G., & Campbell, K.M. (2007). Logistic regression. *Methods in molecular biology* (Vol. 404, pp. 273–301). New York: Humana Press.

Nocedal, J., & Wright, S.J. (1999). Quadratic programming. *Numerical optimization* (pp. 438–486). New York: Springer.

Saragadam, V., Balestriero, R., Veeraraghavan, A., Baraniuk, R.G. (2024). DeepTensor: Low-rank tensor decomposition with deep network priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(12), 10337–10348, https://doi.org/10.1109/TPAMI.2024.3450575

Tanaka, A., Tomiya, A., Hashimoto, K. (2021). *Deep learning and physics* (1st ed.). New York, NY: Springer.

Tao, D., Li, X., Wu, X., Hu, W., Maybank, S.J. (2007). Supervised tensor learning. *Knowledge and Information Systems*, *13*(1), 1–42, https://doi.org/10.1007/s10115-006-0047-8

Tucker, L.R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31*(3), 279–311, https://doi.org/10.1007/BF02289464

Xie, Y., Huang, T., Ji, Q., Yang, M., Wang, J., Tu, X., ... others (2020). Design of an arbitrary ratio optical power splitter based on a discrete differential multi-objective evolutionary algorithm. *Applied Optics*, *59*(6), 1780–1785, https://doi.org/10.1364/AO.382215

Yang, F.-J. (2018). An implementation of naive Bayes classifier. *2018 international conference on computational science and computational intelligence (csci)* (pp. 301–306). IEEE.

Yeung, C., Tsai, J.-M., King, B., Pham, B., Ho, D., Liang, J., ... Raman, A.P. (2021). Multiplexed supercell metasurface design and optimization with tandem residual networks. *Nanophotonics*, *10*(3), 1133–1143, https://doi.org/10.1515/nanoph-2020-0549

Zhang, H., Ijaz, M., Blaikie, R.J. (2023). Recent review of surface plasmons and plasmonic hot electron effects in metallic nanostructures. *Frontiers of Physics*, *18*, 63602, https://doi.org/10.1007/s11467-023-1328-9

Zhang, X., Zhao, S., Wang, J., Guo, L., Wang, X., Sun, H. (2024). Purity-preserving kernel tensor low-rank learning for robust subspace clustering. *IEEE Transactions on Circuits and Systems for Video Technology*, *34*(3), 1900–1913, https://doi.org/10.1109/TCSVT.2023.3299318