Supervised Learning and Canonical Decomposition of Multivariate Functions

Nikos Kargas, Nicholas D. Sidiropoulos, *Fellow, IEEE*

***Abstract*—Learning a function from input and output data pairs is one of the most fundamental tasks in machine learning. In this work, we propose a generalization of the Canonical Polyadic Decomposition (CPD) from tensors to multivariate functions of continuous variables, and show how it can be applied to supervised learning. We approximate a compactly supported multivariate function using a tensor of truncated multidimensional Fourier series coefficients and propose a *hidden tensor factorization* formulation for learning a low-rank CPD model of the Fourier coefficients tensor. In contrast to prior work, our method is quite general as it can model any compactly supported multivariate function that can be well-approximated by a finite multidimensional Fourier series, and under certain conditions it guarantees that the unknown function is uniquely characterized by the given input-output data. Furthermore, our model naturally allows stochastic gradient updates allowing it to scale to larger datasets. We develop two optimization algorithms and demonstrate promising results on synthetic and real multivariate regression tasks.**

***Index Terms*—Canonical polyadic decomposition, supervised learning, Fourier series expansion.**

I. Introduction

Currently, the most popular methods for approximating nonlinear functions rely on neural networks. Despite their huge success, it is not well understood yet why they work so well, are difficult to interpret, and do not offer insights regarding the structure of the function being approximated. The universal approximation theorem [1] states that a feedforward neural network with a linear output layer and at least one hidden layer can approximate any continuous function on a compact set, provided that the number of hidden units is large enough. However, it is not guaranteed that the neural network will actually learn the true function; not only because training is a hard non-convex problem and the optimization algorithm may fail, but more importantly because the neural network may learn a different function which is also able to reproduce the training examples (i.e., one that agrees with the sought one over the training set). Two different networks that have fitted the training data equally well, may differ substantially on samples that have not been observed [2]. This is because the sought function may not be identifiable from the given input-output data. This motivates the study of models that possess identifiability guarantees, and learn unique mappings between the input-output data pairs. In this work, we take a step towards this direction and propose a novel method for learning a nonlinear function based on the Canonical Polyadic Decomposition (CPD), also known as Parallel Factor Analysis (PARAFAC) [3] of N-way tensors.

Tensor decomposition has been successfully applied in many fields such as machine learning, data mining, signal processing and statistics [4]–[6]. The CPD has been successful in modeling large multi-dimensional and multi-relational data [7], [8], understanding large knowledge bases [9], [10] and building recommender system models [11]. Tensor decomposition has also been used in deep learning, for speeding up and compressing neural network models [12], [13]. CPD is a powerful tool, that can help us build identifiable and parsimonious models which can approximate complex nonlinear functions [14]. Learning parsimonious models leads to significant reduction in the number of parameters, and identifiability of the parameters can offer important insights regarding the inner structure of a function.

In this paper, we tackle the problem of learning a nonlinear function, using a low-rank CPD model of a finite multidimensional Fourier series expansion of the unknown multivariate function. If a function is compactly supported and sufficiently smooth, then it can be well approximated using a finite sum of orthogonal complex exponentials. Finite orthogonal series expansion has been mainly used in univariate or bivariate regression tasks where an expansion based on cosine, wavelet or polynomial functions is used to predict a single output variable [15]. However, when dealing with multivariate functions with $N \gg 2$, the approach quickly becomes infeasible from the computation and memory point of view, since the number of series coefficients scales as $O\left(K^N\right)$, where K is the number of coefficients associated with each variable and N is the number of variables. To remedy this problem, we propose to decompose the N-way Fourier series tensor using CPD. This yields a *generalized canonical polyadic decomposition* (GCPD) representation of compactly supported multivariate functions. We present two methods for fitting the resulting model and evaluate our approach on several regression tasks. Our approach has the following important properties:

- **Expressiveness**: Our model is quite general, as it can model any compactly supported multivariate function that can be well approximated by a finite multidimensional Fourier series. The basis functions imply smoothness of the function so that the model is able to generalize, while the tensor rank and the number of Fourier coefficients are used to control the expressiveness of the model. Specifically, the rank and number of coefficients let us control the approximation error, which is bounded for a

Manuscript received July 21, 2020; revised December 1, 2020; accepted January 12, 2021. This work was supported by NSF IIS-1704074.

N. Kargas is with the Department of ECE, University of Minnesota, Minneapolis, MN 55455; N. D. Sidiropoulos is with the Department of ECE, Univercity of Virginia, Charlottesville, VA 22904 (e-mail: karga005@umn.edu; nikos@virginia.edu).

broad class of functions.

- **Efficiency**: The complexity of predicting the output response given an input vector scales linearly with the dimension of the input, which makes our approach suitable for large-scale problems.
- **Identifiability**: Under certain conditions, our approach guarantees that the unknown function is uniquely characterized by the given input-output data i.e., there is a single set of parameters that describes the nonlinear mapping between the input-output pairs. Identifiability of the parameters is necessary for model interpretability.

The rest of the paper is organized as follows. We first present related prior work and highlight the main differences between our proposed work and existing work in Section II. In Section III, we review necessary background on tensor decomposition and Fourier series. We present our model and two methods for learning it in Section IV. Finally, we provide experimental results on synthetic and real datasets in Section V, and we conclude in Section VI.

II. Related Work

Polynomial Networks (PN) and Factorization Machines (FM) are related approaches which rely on low-rank parameterization of the coefficient tensor of multivariate polynomials [16], [17]. The core model of the two approaches is the Polynomial Regression (PR) model. Polynomial regression is usually limited to low-order interactions since the computational complexity scales as $O\left(N^d\right)$, where d is the degree of the polynomial, rendering it impractical even for small number of variables. Both PN and FM use low-rank CPD models to break the curse of dimensionality. Their main difference is that FM uses a subset of all possible feature interactions, which makes the optimization problem multi-convex, and allows for a simple coordinate descent algorithm to be applied. FM has been extended to higher degree polynomials ($d > 3$) and has been very successful in sparse high-dimensional data regression tasks [18]. Other tensor decomposition models, such as the Tensor Train (TT) [19] and the Tucker model [20], have also been used for parameterizing polynomial functions [21], [22]. In [23] a tensor-based method was proposed that computes a decoupled representation of a vector-valued polynomial function. The proposed method builds a tensor that contains as frontal slabs the Jacobian matrix of the function, evaluated at a set of points and extracts a simpler function representation based on univariate polynomials. Its identifiability properties have been studied in [24].

Tensor decomposition has also been used for learning more general functions. For example, a TT decomposition approach has been proposed for learning a multidimensional Fourier series expansion of a function [25]. The authors have proposed a simple block coordinate descent algorithm with linear complexity on the input dimension. TT decomposition depends on the ordering of the input variables and has been shown to possess structure similar to that of a recurrent neural network [26], which is not desirable for non-sequential data. In addition, unlike the CPD, the model parameters of the TT decomposition are not identifiable, i.e., there may exist multiple sets of model parameters that can synthesize the same tensor.

Recently, a single high-order CPD model was used for modeling nonlinear functions with finite-alphabet inputs [14]. The problem was formulated as a tensor completion problem with smooth latent factors. The proposed approach cannot be directly applied to continuous inputs without a discretization step, which results in performance degradation when the discretization is coarse, or high complexity and poor generalization when the discretization is very fine. Generalizations of the CPD to multivariate functions of continuous inputs have been proposed in [27], [28]. The authors *assume* a low-rank model of the multivariate function where each component belongs to a reproducing kernel Hilbert space (RKHS). However, it is not clear what class of functions this model spans, and whether this is a valid assumption for a general multivariate function.

In this work, we propose a novel approach based on a generalized canonical polyadic decomposition suitable for modeling compactly supported multivariate functions with continuous inputs. Our model is very general since it only requires that the multivariate function has compact support and continuous derivatives. A drawback of the models in [27], [28] is that the complexity of a prediction depends on the number of training samples, which can be very high. For example, in the ALS algorithm developed in [28], the authors propose solving a kernel regression problem for each component at each ALS iteration, which is computationally very expensive. On the other hand, our approach is simple and computationally efficient – a parsimonious model that naturally lends itself to stochastic gradient updates, as we will see.

Last but not least, our use of multidimensional Fourier series to convert a continuous multivariate function approximation problem to finite multidimensional tensor decomposition is pleasing, being near and dear to our signal processing fundamentals. Note that unlike conventional uses of multivariate Fourier series, which are typically limited to 2-D or 3-D signals, our approach entails N-D Fourier series with N that can easily run in the order of $30 - 100$, and much higher in modern applications. Our judicious use of CPD modeling enables us to tackle problems in this regime with relative ease.

III. Background

In this section, we review necessary background on tensor decomposition and Fourier series that will prove useful for developing our method.

A. Notation

We use the symbols $\mathbf{x}$, $\mathbf{X}$, $\mathcal{X}$ for vectors, matrices and tensors respectively. We use the notation $\mathbf{x}[n]$, $\mathbf{X}[:, n]$, $\mathcal{X}[:, :, n]$ to refer to a particular element of a vector, a column of a matrix and a slab of a tensor. Symbols $\circ$, $\otimes$, $\circledast$, $\odot$ denote the outer, Kronecker, Hadamard and Khatri-Rao (column-wise Kronecker) product respectively. The vectorization operator is denoted as $\mathrm{vec}(\mathbf{X})$, $\mathrm{vec}(\mathcal{X})$ for a matrix and tensor respectively. The set of integers $S = \{1, \ldots, M\}$ is denoted as $[M]$.

B. Canonical Polyadic Decomposition

A real-valued tensor is a multi-dimensional array $\mathcal{X} \in \mathbb{R}^{K_1 \times \cdots \times K_N}$, where N is the order of the tensor and K_n is the size of its nth mode. Canonical Polyadic Decomposition (CPD) expresses an N-way tensor $\mathcal{X}$ as a sum of rank-1 components, i.e.,

$$\mathcal{X} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]_R = \sum_{r=1}^{R} \mathbf{a}_r^1 \circ \mathbf{a}_r^2 \circ \cdots \circ \mathbf{a}_r^N, \quad (1)$$

where $\mathbf{A}_n = [\mathbf{a}_1^n, \cdots, \mathbf{a}_R^n] \in \mathbb{R}^{K_n \times R}$. Elementwise, we have

$$\mathcal{X}[k_1, \ldots, k_N] = \sum_{r=1}^{R} \prod_{n=1}^{N} \mathbf{A}_n[k_n, r]. \quad (2)$$

The rank R is the minimum number of components needed to synthesize $\mathcal{X}$. The CPD model is universal, i.e., every tensor admits a CPD of finite rank and CPD is unique under mild conditions – see [4] for a recent overview of results in this area.

There are several ways of expressing the CPD model. For example, the CPD of a tensor can be expressed in a matricized form using the mode-n matrix unfolding of $\mathcal{X} = [\![\mathbf{A}_1, \ldots, \mathbf{A}_N]\!]_R$

$$\mathcal{X}^{(n)} = (\odot_{i \neq n} \mathbf{A}_i)\, \mathbf{A}_n^T, \quad (3)$$

where

$$(\odot_{i \neq n} \mathbf{A}_i) = \mathbf{A}_N \odot \cdots \odot \mathbf{A}_{n+1} \odot \mathbf{A}_{n-1} \odot \cdots \odot \mathbf{A}_1.$$

The CPD can also be written in a vectorized form as

$$\mathrm{vec}(\mathcal{X}) = \left(\odot_{i=1}^{N} \mathbf{A}_i\right) \mathbf{1}. \quad (4)$$

The n-mode product of a tensor $\mathcal{X} \in \mathbb{R}^{K_1 \times \cdots \times K_N}$ with a matrix $\mathbf{V} \in \mathbb{R}^{K_n \times J}$ is denoted by $\mathcal{X} \times_n \mathbf{V}$, and is given by

$$(\mathcal{X} \times_n \mathbf{V})[k_1, \ldots, k_{n-1}, j, k_{n+1}, \ldots, k_N] = \sum_{k_n=1}^{K_n} \mathcal{X}[k_1, \ldots, k_N] \mathbf{V}[k_n, j],$$

i.e., each n-th mode fiber is multiplied by the columns of $\mathbf{V}$

$$(\mathcal{X} \times_n \mathbf{V})^{(n)} = \mathcal{X}^{(n)} \mathbf{V}. \quad (5)$$

If $\mathcal{X} = [\![\mathbf{A}_1, \ldots, \mathbf{A}_N]\!]_R$, then by combining (3) and (5) we have

$$\mathcal{X} \times_n \mathbf{V} = [\![\mathbf{A}_1, \ldots, \mathbf{V}^T \mathbf{A}_n, \ldots, \mathbf{A}_N]\!]_R. \quad (6)$$

C. Fourier Series

Fourier series can be used to represent a large class of periodic functions. A function $f : \mathbb{R} \to \mathbb{R}$ is periodic with period T, if $f(x + T) = f(x)\ \forall x \in \mathbb{R}$. One class of periodic functions that can be represented using a Fourier series, is square integrable functions $f \in L_2$ i.e., functions that have finite energy over a single period

$$\|f\|_{L_2}^2 := \int_T |f(x)|^2 dx < \infty. \quad (7)$$

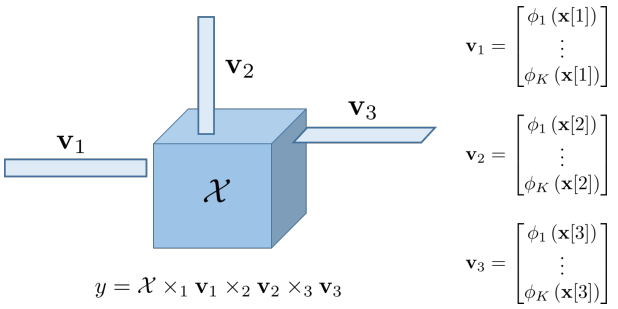


Fig. 1. Fourier series representation of a function of three variables. Tensor $\mathcal{X} \in \mathbb{R}^{K \times K \times K}$ contains the Fourier coefficients and vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ the values of the (cosine) basis functions for the input $\mathbf{x} \in \mathbb{R}^3$. The output is computed using the n-mode product.

Theorem 1: (e.g., see [29][p. 10]) Every periodic function $f \in L_2$ has a unique representation of the form

$$f(x) = \sum_{k \in \mathbb{Z}} \alpha_k e^{jk\omega_0 x},\ \alpha_k = \frac{1}{T} \int_T f(x) e^{-jk\omega_0 x} dx,$$

where $\omega_0 := \frac{2\pi}{T}$ and $(S_K f) := \sum_{k=-K}^{K} \alpha_k e^{jk\omega_0 x}$ converges in L_2 to f, i.e.,

$$\lim_{K \to \infty} \|S_K f - f\|_{L_2} = 0.$$

Fourier series can also be used for modeling non-periodic functions with compact support. Consider a univariate function $f(\cdot)$ supported on the interval $[0, 1]$. Note that restricting the domain of the function on the interval $[0, 1]$ can be done without loss of generality for any compactly supported function, by appropriate shift and scaling of the input. The function $f(\cdot)$ can be extended to $[-1, 1]$ as an even or odd function, by defining $f(x) = f(-x)$ or $f(x) = -f(-x)$ for $x < 0$. The new function can then be made periodic with period $T = 2$, by demanding that $f(x) = f(x+2) = f(x-2)$. This is called the *even (odd) periodic extension* of $f(\cdot)$ and it can be shown that it is expressed by a Fourier cosine or sine series respectively. Using the even periodic extension we have

$$f(x) = \alpha_0 + \sum_{k=1}^{\infty} \alpha_k \sqrt{2} \cos(k\pi x), \quad (8)$$

using orthogonal basis functions $\phi_0(x) = 1$ and $\phi_k(x) = \sqrt{2}\cos(k\pi x),\ k > 0$. In this work, we consider only the cosine extension but different basis functions can be used and can be easily incorporated in our framework e.g., Legendre, Chebyshev polynomials, wavelets [1].

Similar to the univariate case, a multivariate function $f : [0, 1]^N \to \mathbb{R}$, can be represented using a tensor-product basis as

$$f(\mathbf{x}) = \sum_{k_1=0}^{\infty} \cdots \sum_{k_N=0}^{\infty} \alpha_{\mathbf{k}} \prod_{n=1}^{N} \phi_{k_n}(\mathbf{x}[n]), \quad (9)$$

[1]Chebyshev series is related to a Fourier cosine series through a change of variables.

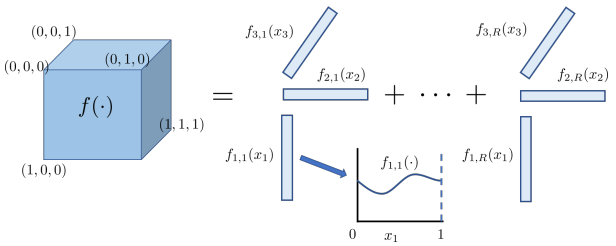


Fig. 2. Generalization of the CPD to compactly supported multivariate functions. A compactly supported function $f : [0,1]^3 \to \mathbb{R}$ is represented using a sum of R rank-1 components. Each rank-1 component is given by the product of three univariate functions.

where $\mathbf{k} = (k_1, \ldots, k_N) \in \mathbb{N}^N$ is a multi-index . The idea of a series estimator is to approximate $f(\cdot)$ by a truncated series with cutoffs $K_1, \ldots, K_N$, i.e.,

$$f(\mathbf{x}) = \sum_{k_1=0}^{K_1-1} \cdots \sum_{k_N=0}^{K_N-1} \alpha_{\mathbf{k}} \prod_{n=1}^{N} \phi_{k_n}(\mathbf{x}[n]). \tag{10}$$

Note that Equation (10) defines an inner-product between two tensors. Let us define an N-way tensor $\mathcal{X} \in \mathbb{R}^{K \times \cdots \times K}$ holding the Fourier coefficients with $\mathcal{X}[k_1, \ldots, k_N] = \alpha_{\mathbf{k}}$ and a rank-1 tensor $\mathcal{V} = [\![\mathbf{v}_1, \ldots, \mathbf{v}_N]\!]$, where $\mathbf{v}_n[k_n] = \phi_{k_n}(\mathbf{x}[n])$. Then, the inner product between these two tensors is given by

$$\begin{aligned} \langle \mathcal{X}, \mathcal{V} \rangle &= (\mathbf{v}_N \otimes \cdots \otimes \mathbf{v}_1)^T \mathrm{vec}(\mathcal{X}) \\ &= \mathcal{X} \times_1 \mathbf{v}_1 \cdots \times_N \mathbf{v}_N. \end{aligned} \tag{11}$$

A visualization is shown in Figure 1 for a function of 3 variables.

Theorem 1 can be extended to multi-dimensional Fourier series as well. However, convergence of the Fourier series does not automatically imply that a truncated Fourier series will be a good approximation for $f(\cdot)$. The approximation error between the function and the truncated series depends on how fast the Fourier coefficients tend to zero. The smoother the function is, the faster its Fourier coefficients and the error tend to zero as it is illustrated by the following theorem.

Theorem 2: [29][p. 164] Let $p \in \mathbb{N}$. If the partial derivatives $\frac{\partial^{\beta_1}}{\partial x_1} \cdots \frac{\partial^{\beta_N}}{\partial x_N} f(\mathbf{x})$ of $f(\cdot)$ exist and are absolutely integrable for all $\beta_1, \ldots, \beta_N$ with $\sum_{n=1}^{N} \beta_n \leq p$ then

$$\lim_{\|\mathbf{k}\|_2 \to \infty} (1 + \|\mathbf{k}\|_2^p) \alpha_{\boldsymbol{k}} = 0. \tag{12}$$

In other words, $\alpha_{\mathbf{k}}$ decays faster than the sequence $\frac{1}{1+\|\mathbf{k}\|_2^p}$. Finite orthogonal series expansion is mainly used in univariate or bivariate regression tasks where one has to estimate the Fourier coefficients from input-output data pairs.

IV. Proposed Approach

In this section, we give a detailed description of our approach. Specifically, we present our model in Section IV-A, and present two different ways for training the model in Sections IV-B, IV-C.

A. Generalized Canonical Polyadic Decomposition

Our goal is to learn a high-dimensional compactly supported function $f : [0,1]^N \to \mathbb{R}$ from M input-output pairs $\{\mathbf{x}_m, y_m\}_{m=1}^M$

$$y_m = f(\mathbf{x}_m) + w_m, \quad m \in [M],$$

where w_m represents noise. The output y_m may correspond to a continuous value we want to estimate (regression) or a discrete class label (classification). Initially, we assume that the data points $\{\mathbf{x}_m\}_{m=1}^M$ are uniformly and independently sampled from the unit hypercube $[0,1]^N$ and $w_m \sim \mathcal{N}(0, \sigma^2)$.

We propose using a truncated cosine series expansion with cutoffs $K_1 = \cdots = K_N = K$. Orthogonal series approximation is mainly used when N is small. When dealing with functions of many variables the approach becomes impractical even for small K, since the number of parameters grows exponentially with N. Even if we are able to compute the series coefficients, the variance of the estimates can be high when M is small. More complicated estimation procedures are required in this case involving for example, hard or soft thresholding of the coefficients [15]. One way of dealing with these problems is to further restrict the class of functions, using for example, additive models of univariate functions [15], [30].

Instead, we propose a novel approach based on low-rank tensor decomposition. Specifically, we propose fitting a low-rank CPD on the N-way cosine series coefficient tensor. Let us assume a rank-R CPD model for the N-way coefficient tensor $\mathcal{X} \in \mathbb{R}^{K \times \cdots \times K}$

$$\mathcal{X}[k_1, \ldots, k_N] = \sum_{r=1}^{R} \prod_{n=1}^{N} \mathbf{a}_n^r[k_n], \tag{13}$$

where $\mathbf{a}_n^r \in \mathbb{R}^K$. Substituting Equation (13) in Equation (10), we have

$$\begin{aligned} f(\mathbf{x}) &= \sum_{k_1=0}^{K-1} \cdots \sum_{k_N=0}^{K-1} \sum_{r=1}^{R} \prod_{n=1}^{N} \mathbf{a}_n^r[k_n] \mathbf{v}_n[k_n] \\ &= \sum_{r=1}^{R} \sum_{k_2,\ldots,k_N=1}^{K-1} \prod_{n=2}^{N} \mathbf{a}_n^r[k_n] \mathbf{v}_n[k_n] \underbrace{\sum_{k_1=0}^{K-1} \mathbf{a}_1^r[k_1] \mathbf{v}_1[k_1]}_{f_{n,r}(\mathbf{x}[1])}, \end{aligned} \tag{14}$$

where $\mathbf{v}_n[k_n] = \phi_{k_n}(\mathbf{x}[n])$. We note that Equation (14) can also be understood using the mode-n multiplication property in Equation (6). By carrying out the computations in this manner, the expression for $f(\cdot)$ is greatly simplified to

$$f(\mathbf{x}) = \sum_{r=1}^{R} f_{1,r}(\mathbf{x}[1]) \cdots f_{N,r}(\mathbf{x}[N]). \tag{15}$$

Each univariate function in Equation (15) can be expressed as

$$f_{n,r}(\mathbf{x}[n]) = \sum_{k_n=0}^{K-1} \mathbf{a}_n^r[k_n] \mathbf{v}_n[k_n] = \mathbf{v}_n^T \mathbf{A}[:, r], \tag{16}$$

i.e., each component $f_{n,r}(\cdot)$ is a expressed using a truncated Fourier cosine series. Comparing Equations (2) and (15) we can see that our model is a generalization of the CPD to compactly supported multivariate functions. Any compactly

Algorithm 1 FSA-LR & WFSA-LR

Input: $\mathbf{X}$, $\mathbf{y}$, R, K
// Initialization //
if $R \leq K$ **then**
  $\{\mathbf{A}_n\}_{n=1}^N \leftarrow$ CP-GEVD($\mathcal{X}$)
end if
repeat
  // Training the model //
  for $n = 1$ **to** N **do**
    Solve for $\mathbf{A}_n$ via (20)
  end for
until termination criterion is satisfied
// Refinement step //
repeat
  for $n = 1$ **to** N **do**
    for $k = 1$ **to** K **do**
      Solve for $\mathbf{A}_n[k,:]$ via (22)
    end for
  end for
until termination criterion is satisfied
return $\{\mathbf{A}_n\}_{n=1}^N$

supported multivariate function can be modeled using a countable tensor of Fourier coefficients. Continuity allows us to truncate this tensor to a finite one by choosing the number of Fourier coefficients. The CPD model is universal, i.e, any finite coefficient tensor $\mathcal{X}$ admits a CPD with finite rank, which implies that any compactly supported multivariate function that can be expressed (approximated) as a finite multivariate Fourier series can be modeled (approximated, respectively) as in Equation (15). The proposed generalized canonical polyadic decomposition is visualized for a function of three variables in Figure 2.

Employing this model, we circumvent the curse of dimensionality, as the number of parameters drops from $O(K^N)$ to $O(KNR)$. The output of the model given the input vector $\mathbf{x}$ is given via (15), (16) as

$$\widehat{y} = (\mathbf{v}_1^T\mathbf{A}_1 \circledast \cdots \circledast \mathbf{v}_N^T\mathbf{A}_N)\mathbf{1} = (\circledast_{n=1}^N \mathbf{v}_n^T\mathbf{A}_n)\mathbf{1}. \quad (17)$$

The complexity of computing the response of a new data point is $O(NKR)$, which makes our model suitable for high-dimensional data. The number of coefficients K controls the desired smoothness of the function and the rank controls the expressiveness. Next, we propose two different methods for learning the model.

B. Direct Optimization

Let us denote the training set by $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_M]^T \in \mathbb{R}^{M\times N}$ and $\mathbf{y} = [y_1, \ldots, y_M]^T \in \mathbb{R}^M$. Our aim is to compute a low-rank representation of the Fourier series coefficients. The Fourier coefficients $\alpha_{\mathbf{k}}$ are given by

$$\alpha_{\mathbf{k}} = \int_0^1 \cdots \int_0^1 f(\mathbf{x})\phi_{\mathbf{k}}(\mathbf{x})d\mathbf{x},$$

where $\phi_{\mathbf{k}}(\mathbf{x}) = \prod_{n=1}^N \phi_{k_n}(\mathbf{x}[n])$. Assuming that $\{\mathbf{x}_m\}_{m=1}^M$ are uniformly sampled from the unit hypercube, a natural estimator for each coefficient $\alpha_{\mathbf{k}}$ is

$$\mathcal{X}[k_1, \ldots, k_N] = \widehat{\alpha}_{\mathbf{k}} = \frac{1}{M}\sum_{m=1}^M y_m\phi_{\mathbf{k}}(\mathbf{x}_m). \quad (18)$$

Equation (18) is an unbiased estimator for the coefficients i.e.,

$$\mathbb{E}[\widehat{\alpha}_{\mathbf{k}}] = \mathbb{E}\left[(f(\mathbf{x}) + w)\phi_{\mathbf{k}}(\mathbf{x})\right] = \alpha_{\mathbf{k}}.$$

Having obtained an estimate for each coefficient, a straightforward approach is to approximate $\mathcal{X}$ using a low-rank CPD model by minimizing the squared error

$$\min_{\{\mathbf{A}_n\}_{n=1}^N} \|\mathcal{X} - [\![\mathbf{A}_1, \ldots, \mathbf{A}_N]\!]_R\|_F^2. \quad (19)$$

We refer to this approach as Fourier series Approximation with Low-Rank constraint (FSA-LR). Optimization problem (19) involves a multilinear form, and the most popular approach for tackling this kind of problem is via Alternating Least Squares (ALS) [4]. A nice property of ALS is that it ensures monotonic decrease of the cost function. Additionally, convergence to a stationary point can be guaranteed by adding a proximal regularization term to the cost function [31]. The key idea of ALS is to cyclically update the variables while keeping all but one fixed. The updates for each factor matrix $\mathbf{A}_n$ are given by

$$\mathbf{A}_n \leftarrow \arg\min_{\mathbf{A}} \|\mathcal{X}^{(n)} - (\odot_{i\neq n}\mathbf{A}_i)\mathbf{A}^T\|_F^2. \quad (20)$$

When the tensor rank is small, CPD can be computed exactly. For example, if the factor matrices are full column rank, the CPD can be computed using a Generalized Eigenvalue Decomposition [3], [32]. In practice, $\mathcal{X}$ will never be low-rank due to noise. Nevertheless, in many practical applications, the associated tensors can be well-approximated using low-rank and the output of GEVD provides a good initial estimate.

When M is small, we expect the variance of the coefficient estimates to be large. Therefore we use a refinement step, by weighting each coefficient inversely proportional to its variance. An estimate of the variance can be obtained using the training set

$$\widehat{\sigma}^2_{k_1,\ldots,k_N} = \frac{1}{M}\sum_{m=1}^M \left(\mathcal{X}[k_1, \ldots, k_N] - y_m\phi_{\mathbf{k}}(\mathbf{x}_m)\right)^2.$$

Let us define the tensor $\mathcal{W}$ as $\mathcal{W}[k_1, \ldots, k_N] = \widehat{\sigma}^{-1}_{k_1,\ldots,k_N}$. We refine the estimate obtained from (19) by solving

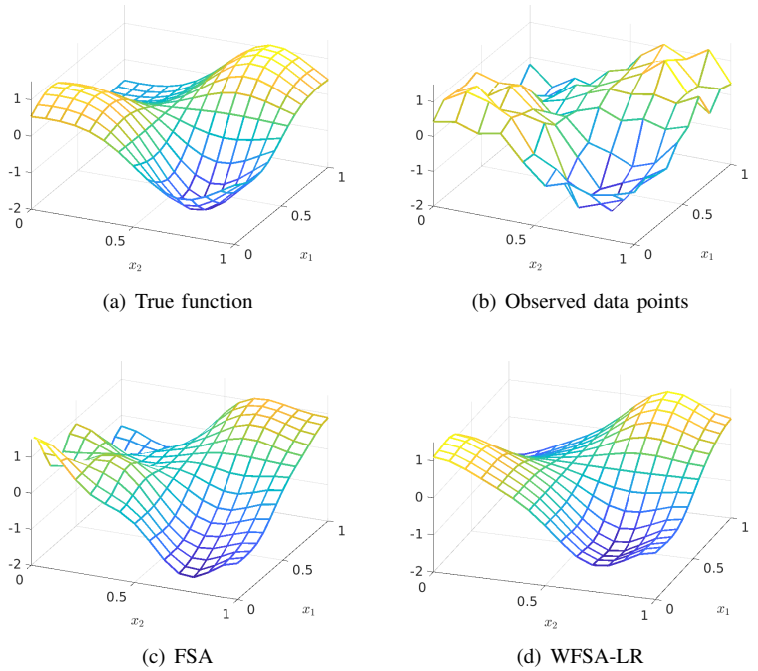
$$\min_{\{\mathbf{A}_n\}_{n=1}^N} \|\mathcal{W} \circledast (\mathcal{X} - [\![\mathbf{A}_1, \ldots, \mathbf{A}_N]\!]_R)\|_F^2. \quad (21)$$

This is a weighted tensor factorization problem. Similarly as before, we can tackle the weighted tensor decomposition problem with an ALS algorithm. The optimization problem can be equivalently written as

$$\min_{\{\mathbf{A}_n\}_{n=1}^N} \left\|\widehat{\mathcal{X}}^{(n)} - \mathcal{W}^{(n)} \circledast \left((\odot_{i\neq n}\mathbf{A}_j)\,\mathbf{A}_n^T\right)\right\|_F^2.$$

A simple update rule can be obtained by fixing all except for a single row of $\mathbf{A}_n$. The update for each row is given by

$$\mathbf{A}_n[k_n,:] \leftarrow \arg\min_{\mathbf{a}_{n,k_n}} \|\mathbf{z}_{n,k_n} - \mathbf{Q}_{n,k_n}\mathbf{a}_{n,k_n}\|_F^2, \quad (22)$$



(a) True function (b) Observed data points

(c) FSA (d) WFSA-LR

Fig. 3. Toy example on learning a rank-1 bivariate function.

where

$$\mathbf{z}_{n,k_n} = \widehat{\mathcal{X}}^{(n)}[:,k_n], \mathbf{Q}_{n,k_n} = \text{diag}(\mathcal{W}^{(n)}[:,k_n])\left(\odot_{i\neq n}\mathbf{A}_j\right).$$

After the algorithm has converged, we use the learned factor matrices $\{\mathbf{A}_n\}_{n=1}^N$ to compute the model prediction for a new data point $\mathbf{x}$ according to Equation (17). We refer to this approach as Weighted Fourier Series Approximation with Low-Rank constraint (WFSA-LR). The full procedure is shown in Algorithm 1.

Toy Example: An example to illustrate the effect of the weighted low-rank decomposition is shown in Figure 3. We generate $M = 100$ data points $\{\mathbf{x}_m, y_m\}_{m=1}^M$ according to

$$y_m = f_1(\mathbf{x}_m[1])f_2(\mathbf{x}_m[2]) + w_m,$$

which is a product of univariate functions and corresponds to a rank-1 model. The functions $f_1(\cdot), f_2(\cdot)$ are defined on the interval $[0, 1]$ as a mixture of Gaussians

$$f_n(x) = s_1^n\mathcal{N}\left(x;\mu_{1,n},\sigma_{1,n}^2\right) + s_2^n\mathcal{N}\left(x;\mu_{2,n},\sigma_{2,n}^2\right),$$

where $s_{i,r}^n \in \{-1, 1\}$.

The true function is shown in Figure 3(a). Figure 3(b) shows the output of the function for 100 uniformly sampled data points, distorted by white Gaussian noise. We use these points as our training set and compare Fourier Series Approximation (FSA) learned without assuming low-rank against FSA-LR and WFSA-LR with rank $R = 1$. For plain FSA we estimate the Fourier series coefficients using all the training data points and Equation (18). For FSA-LR and WFSA-LR we further decompose the estimated coefficient tensor using a CPD model of rank-1. We evaluate the learnt functions on unseen data points using Equation (10). We choose the number of Fourier coefficients using a validation set (15% of the training set). Figure 3(c) shows the function learned without assuming a low-rank solution. WFSA-LR filters the noise and produces a smoother estimate as shown in Figure 3(d) (where 'plain' FSA-LR is not shown to save space). The improvement was also numerically verified by computing the Mean Squared Error (MSE) at 300 unseen points. The MSE of the different methods is 0.067 (FSA), 0.039 (FSA-LR), and 0.027 (WFSA-LR).

The downside of FSA-LR and WFSA-LR is that every Fourier coefficient needs to be estimated (but not necessarily stored, as it can be estimated on-the-fly during computations). Additionally, in order to have unbiased estimates of the coefficients, we require that the samples are uniformly drawn from the unit hypercube, something that we cannot guarantee in applications. In the following, we bypass both these restrictions.

C. Hidden Tensor Factorization

A standard approach in supervised learning is to learn a function by minimizing the empirical risk. In this section we propose fitting the coefficients directly on the training data by minimizing the error between the true and predicted response i.e.,

$$\frac{1}{M}\sum_{m=1}^{M} L(y_m - f(\mathbf{x}_m)) + G(f), \tag{23}$$

where $L(\cdot)$ is a loss function and $G(\cdot)$ a regularization term. In this work, we focus on regression tasks and choose the squared error as our loss function. Let us define matrices $\mathbf{V}_n \in \mathbb{R}^{K\times M}$ as $\mathbf{V}_n[k,m] = \phi_k\left(\mathbf{x}^m[n]\right)$. Recall that each output response can be viewed as an inner product between two tensors

Algorithm 2 FSA-HTF (ALS)

Input: $\mathbf{X}$, $\mathbf{y}$, $\mathbf{X}_{\text{val}}$, $\mathbf{y}_{\text{val}}$ R, K_0, K, $\max_{\text{iter}}$
Initialize $\{\mathbf{A}_n \in \mathbb{R}^{K_0 \times F}\}_{n=1}^N$
repeat
  for $n = 1$ **to** N **do**
    Compute $\mathbf{Q}_n$ via (25)
    Solve for $\mathbf{A}_n$ via (26)
  end for
  Compute MSE_{val} using $\mathbf{X}_{\text{val}}$, $\mathbf{y}_{\text{val}}$
  if $K_0 < K$ and MSE_{val} has not improved **then**
    for $n = 1$ **to** N **do**
      $\mathbf{A}_n \leftarrow \begin{bmatrix} \mathbf{A}_n \\ \mathbf{0}^T \end{bmatrix}$
    end for
    $K_0 = K_0 + 1$
  end if
until $\max_{\text{iter}}$ is reached or MSE_{val} stops improving
return $\{\mathbf{A}_n\}_{n=1}^N$

as in Equation (11). By minimizing the loss function in Equation (23) we arrive at the following optimization problem

$$\min_{\{\mathbf{A}_n\}_{n=1}^N} \frac{1}{M} \left\| \mathbf{y} - (\odot_{n=1}^N \mathbf{V}_n)^T \left(\odot_{i=1}^N \mathbf{A}_i\right) \mathbf{1} \right\|_2^2 + \sum_{n=1}^N \rho \|\mathbf{A}_n\|_F^2,$$

where we have used the vectorized representation of the tensor $\mathcal{X}$ and ρ is a regularization parameter. We observed that adding this regularization term enhances the generalization ability of our method in practice. The problem is conceptually similar to a linear least squares problem, but with a tensor rank-constrained solution [33]. Moreover, the measurement matrix has a very special structure, as each row corresponds to a rank-1 tensor. We refer to this approach as Fourier Series Approximation via Hidden Tensor Factorization (FSA-HTF), because we do not directly observe the full tensor. Instead, we aim at recovering the CPD model from rank-1 measurements of the 'hidden' tensor. Because of the special structure of the measurement matrix, we can avoid instantiating the tensor and the Khatri-Rao product. In scalar form we have

$$\min_{\{\mathbf{A}_n\}_{n=1}^N} \frac{1}{M} \sum_{m=1}^M \left(y_m - \left(\circledast_{n=1}^N (\mathbf{V}_n[:,m]^T \mathbf{A}_n)\right) \mathbf{1}\right)^2 + \sum_{n=1}^N \rho \|\mathbf{A}_n\|_F^2.$$

Similarly as before, we can develop a simple ALS algorithm. Fixing all variables except for $\mathbf{A}_n$ we have

$$\min_{\mathbf{A}_n} \frac{1}{M} \sum_{m=1}^M \left(y_m - \mathbf{V}_n[:,m]^T \mathbf{A}_n \mathbf{Q}_n[:,m]\right)^2 + \rho \left\|\mathbf{A}_n\right\|_F^2, \tag{24}$$

where $\mathbf{Q}_n \in \mathbb{R}^{R \times M}$ with

$$\mathbf{Q}_n = \left(\circledast_{i \neq n} (\mathbf{A}_i^T \mathbf{V}_i)\right). \tag{25}$$

Optimization problem (24) is a least squares problem. Setting the gradient equal to zero we have the following set of linear equations

$$\frac{1}{M} \sum_{m=1}^M \left(\mathbf{V}_n[:,m]^T \mathbf{A}_n \mathbf{Q}_n[:,m]\right) \mathbf{V}_n[:,m] \mathbf{Q}_n^T[:,m] + \rho \mathbf{A}_n = \frac{1}{M} \sum_{m=1}^M y_m \mathbf{V}[:,m] \mathbf{Q}^T[:,m]. \tag{26}$$

We solve the system of linear equations using conjugate gradient descent [34]. Conjugate gradient is an iterative algorithm with a complexity of $O(KRM)$ per each iteration.

When the number of samples M is large, we propose using Stochastic Gradient Descent (SGD). At each step we update all factors simultaneously by first sampling a batch $\mathcal{F} \subset \{1, \ldots, M\}$ of size $|\mathcal{F}|$ and taking a gradient step i.e,

$$\mathbf{A}_n \leftarrow \mathbf{A}_n - \alpha \mathbf{G}_n, \tag{27}$$

where α is the stepsize and $\mathbf{G}_n$ is the gradient with respect to the sampled points

$$\mathbf{G}_n = \frac{1}{|\mathcal{F}|} \sum_{m \in \mathcal{F}} \left(\mathbf{V}_n[:,m]^T \mathbf{A}_n \mathbf{Q}_n[:,m]\right) \mathbf{V}_n[:,m] \mathbf{Q}_n^T[:,m] + \rho \mathbf{A}_n - \frac{1}{|\mathcal{F}|} \sum_{m=1}^M y_m \mathbf{V}[:,m] \mathbf{Q}^T[:,m].$$

Each step of SGD has a complexity of $O(NKR|\mathcal{F}|)$. For the termination of both algorithms we compute the MSE on a validation set and stop if a limit on the number of iterations has been reached, or the validation MSE has not improved in the last T iterations. The full procedures are shown in Algorithm 2 and Algorithm 3 respectively.

Initialization: It was empirically observed that random initialization was not effective in real data experiments for FSA-HTF. The algorithm exhibits better performance when starting from a smoother and simpler model. Therefore, when training FSA-HTF with ALS, we generate factor matrices $\mathbf{A}_n \in \mathbb{R}^{K_0 \times F}$, where K_0 is a value smaller than K. If the current MSE on the validation stops improving, we augment factors $\mathbf{A}_n$ by adding an extra row. The algorithm is terminated when no improvement is observed on the validation set and $K_0 = K$. When training FSA-HTF via SGD, factors $\mathbf{A}_n$ are initially set to zero except for the first K_0 rows. For all experiments we set $K_0 = 2$.

D. Uniqueness of the proposed model

In this section, we briefly discuss conditions under which the proposed model is identifiable i.e., there exist unique factor matrices $\mathbf{A}_n$ that synthesize the tensor $\mathcal{X}$. This is important because it tells us when it is possible to recover the underlying nonlinear function, and hence learn the true mapping between the input and output pairs.

If the function $f(\cdot)$ can be written as a finite multidimensional Fourier series, then under certain rank conditions on the Fourier series coefficient tensor, the decomposition is unique.

Algorithm 3 FSA-HTF (SGD)

Input: $\mathbf{X}$, $\mathbf{y}$, $\mathbf{X}_{\text{val}}$, $\mathbf{y}_{\text{val}}$ R, K, $|\mathcal{F}|$
Initialize $\{\mathbf{A}_n\}_{n=1}^N$
repeat
  Sample $|\mathcal{F}|$ data points
  for $n = 1$ **to** N **do**
    Update $\mathbf{A}_n$ via (27)
  end for
  Compute MSE_{val} using $\mathbf{X}_{\text{val}}$, $\mathbf{y}_{\text{val}}$
until $\max_{\text{iter}}$ is reached or MSE_{val} stops improving
return $\{\mathbf{A}_n\}_{n=1}^N$

Specifically, given a decomposition of the Fourier coefficient tensor $\mathcal{X} = [\![\mathbf{A}_1, \ldots, \mathbf{A}_N]\!]_R$, if

$$\sum_{n=1}^{N} k_{\mathbf{A}_n} \geq 2R + N - 1,$$

then the factor matrices are unique up to common permutation and scaling/counterscaling of their columns [4]. Here, $k_{\mathbf{A}}$ denotes the *Kruskal rank* of the matrix $\mathbf{A}$, which is equal to the largest integer such that every subset of $k_{\mathbf{A}}$ columns are linearly independent. For generic $\mathbf{A}_n$ and maximal u such that $K \geq 2^u$, it holds that the decomposition of $\mathcal{X}$ is almost surely unique if $R \leq 2^{(N-1)(u-1)}$ [35]. The upshot is that under certain rank conditions on the Fourier coefficient tensor, it is possible to recover a unique representation of the multivariate function *from a finite set of input-output measurements*.

Remark: We should mention that the above identifiability results are derived for tensors under an exact low-rank decomposition. In practice, the Fourier series coefficient tensor will not have an exact low-rank decomposition mainly due to noise and/or limited number of samples. However, practical experience is that many real-life tensors can be well-approximated using low ranks and the uniqueness results often carry even in the approximate case – as we will also demonstrate in the experimental results section. Even when the low-rank assumption does not strictly hold, our approach is a very reasonable "principal component" approximation and can identify a controllable approximation of the true function.

V. Experimental Results

In this section, we evaluate the proposed approach on synthetic and real datasets.

A. Synthetic Data: Low-rank GCPD Model

As a first sanity check, we consider a function which can modeled using a low-rank generalized canonical polyadic decomposition model similar to Equation (15).

$$y_m = \sum_{r=1}^{R} \prod_{n=1}^{N} f_{n,r}(\mathbf{x}_m[n]) + w_m. \tag{28}$$

Each function $f_{n,r}(\cdot)$ is defined on the interval $[0, 1]$ as follows

$$f_{n,r}(x) = \sum_{i=1}^{2} s_{i,r}^n \mathcal{N}\left(x; \mu_{i,r}^n, {\sigma_{i,r}^n}^2\right),$$

where $s_{i,r}^n$ takes the value $+1$ or -1 with equal probability, $\mu_{1,r}^n \sim \mathcal{U}(0.1, 0.4)$, $\mu_{2,r}^n \sim \mathcal{U}(0.6, 0.9)$ and $\sigma_{i,r}^n \sim \mathcal{U}(0.2, 0.5)$. We set $N = 5$, $R = 3$ and generate 1000 data points that are uniformly sampled from the $[0, 1]$ interval. We generate the corresponding outputs using Equation (28). We use 85% of the data as the training set and the remaining as the test set. We compare 6 different approaches; FSA (without low-rank assumption), FSA-LR, WFSA-LR, FSA-HTF, a feedforward neural network with sigmoid activation function and a feedforward neural network with ReLU activation function. The parameters of the different models, i.e., number of coefficients, rank, number of layers and number of nodes for each layer are tuned using 5-fold cross-validation. We repeat the above procedure 10 times and report the average results.

We first compare the performance of the methods while varying the Signal-to-Noise Ratio (SNR). Figure 4(a) shows the MSE on the test set as a function of the SNR. As expected, the performance of WFSA-LR is slightly better than FSA-LR. Both perform better than plain FSA but after a certain point they reach a plateau, which is due to the small number of samples and the high variance of the estimator. FSA-HTF performs the best, as it generalizes better and is able to learn the true function. We observe that the choice of the nonlinear function is crucial as it highly affects the performance of the neural network. Specifically, the neural network with the sigmoid activation function performs the worst and fails to generalize to the test set. On the other hand, the neural network with the ReLU activation function performs reasonably well and its performance increases as we decrease the noise variance.

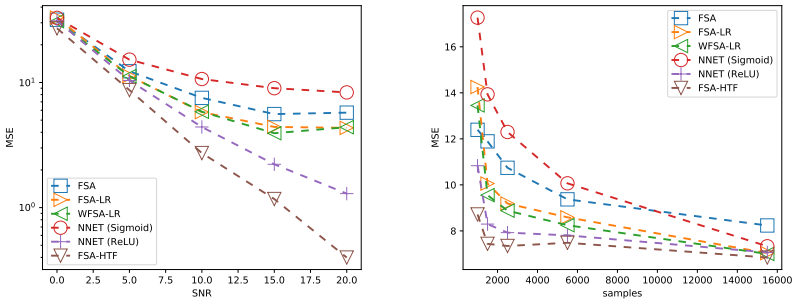
Next, we compare the methods while keeping the SNR fixed and increasing the number of training samples. We set $\text{SNR} = 5$ and fix the number of test samples to be 500. We vary the number of training samples from 500 to 15000. Figure 4(b) shows the MSE on the test set as a function of the total number of samples. Both FSA-HTF and the neural network with the ReLu activation function are able to learn the function using approximately 1000 samples. The performance of all methods improves as we increase the number of samples and we observe that using 15000 samples, all approaches except for FSA, perform similarly.

B. Synthetic Data: 1-Layer Neural Network

Here we consider a function which is not necessarily low-rank. We generate 1000 data points according to a neural network with 1 hidden layer with 20 nodes and ReLU activation function. We set the input dimension $N = 5$. Specifically, the input-output data pairs are generated according to

$$y = \mathbf{w}_2^T \text{ReLU}(\mathbf{W}_1 \mathbf{x}), \tag{29}$$

where $\mathbf{W}_1 \in \mathbb{R}^{20 \times 5}$, $\mathbf{w}_2 \in \mathbb{R}^{20}$. We use 85% of the data as the training set and the remaining as the test set. The parameters of the different models, i.e., number of coefficients, rank, number of layers and number of nodes for each layer are tuned using 5-fold cross-validation. For the two neural networks, we make sure that the true parameters are also examined during the cross-validation. We perform 10 Monte Carlo simulations and report the average results.



(a) True model: GCPD, $M = 1000$.

(b) True model: GCPD, SNR=5 dB.

(c) True model: NN, $M = 1000$.

(d) True model: NN, SNR=5 dB.

Fig. 4. MSE on datasets generated using a mixture of products of univariate functions (top row) and a 1-Layer neural network using ReLU (bottom row).

We compare the performance of the methods while varying the SNR. Figure 4(c) shows the MSE on the test set as a function of the SNR. As before, we observe that FSA, FSA-LR and WFSA-LR quickly reach a plateau. The neural network which uses the correct activation function and agrees with the model that generated the data, performs the best. Note that misspecification of the activation function hurts the performance. The neural network which uses a sigmoid activation function performs much worse. FSA-HTF performs reasonably well and its performance increases as we decrease the noise variance.

Next, we compare the methods when keeping the SNR fixed SNR $= 5$ and increasing the number of training samples (Fig. 4(d)). The MSE drops as the number of samples increases for all methods. FSA-HTF and the two neural network approaches perform similarly when we increase the number of samples to 15000.

The above experiments serve as a sanity check that our method can indeed learn a function which admits a low-rank model, but can also learn functions that are not known to be low-rank – such as a neural network with random weights. Moreover, we see how important is the choice of the activation function and how it can affect the performance of a neural network.

C. Real Data

Next, we evaluate the proposed approach on several regression tasks using datasets obtained from the UCI machine learning repository [36]. The name and dimensions of each dataset are shown in Table I. For real data experiments we evaluate only FSA-HTF since real data are not uniformly and independently sampled, hence violating the conditions assumed by FSA-LR. We compare our approach against various standard baselines: Linear Regression (LR), Polynomial Regression (PR), Support Vector Regression (SVR) with a Gaussian kernel, a Neural Network with ReLU activation function (NNET) and a Decision Tree (DT). We also compare our method against two other tensor methods, namely AMP [28] which assumes a low-rank generalized canonical polyadic decomposition model where each component belongs to a RKHS; and our own earlier CSID method [14] which leverages a low-rank CPD model but requires the input data to be discrete (or discretized).

TABLE I
DATASETS INFORMATION.

DATASET	N	M
QSAR AQUATIC TOXICITY (QSAR)	9	546
ENERGY EFFICIENCY (EE)	8	788
AIRFOIL SELF-NOISE (ASN)	6	1503
SKILLCRAFT MASTER TABLE (SMT)	18	3337
ABALONE (AB)	8	4177
CYCLE POWER PLANT (CPP)	4	9568
SUPERCONDUCTIVTY (SC)	81	21263
PHYSICOCHEMICAL PROPERTIES (PP)	9	45730

For each dataset we split the data into two sets; 85% used for training and 15% used for testing. The parameters for each

TABLE II
MSE PERFORMANCE OF STANDARD BASELINE MODELS.

DATASET	LR	PR	SVR (RBF)	NNET	DT
QSAR	1.41 ± 0.31	1.41 ± 0.44	$\mathbf{1.19 \pm 0.30}$	1.42 ± 0.55	1.64 ± 0.37
EE [1]	8.15 ± 1.13	0.76 ± 0.08	4.69 ± 0.68	0.31 ± 0.08	0.30 ± 0.03
EE [2]	10.39 ± 1.84	2.87 ± 0.36	7.36 ± 1.66	2.03 ± 0.71	3.21 ± 0.26
ASN	23.27 ± 1.74	15.95 ± 1.21	11.09 ± 1.60	6.61 ± 1.24	6.45 ± 0.35
SMT	1.03 ± 0.06	1.02 ± 0.09	1.01 ± 0.05	1.03 ± 0.06	1.06 ± 0.04
AB	4.91 ± 0.30	4.54 ± 0.12	4.44 ± 0.21	$\mathbf{4.33 \pm 0.22}$	5.29 ± 0.22
CPP	19.57 ± 0.50	17.18 ± 0.41	15.60 ± 0.51	16.08 ± 0.50	$\mathbf{14.85 \pm 0.97}$
SC	315.93 ± 6.76	171.52 ± 5.41	213.60 ± 6.71	$\mathbf{125.12 \pm 8.13}$	130.56 ± 8.35
PP	27.10 ± 0.27	25.05 ± 1.06	21.24 ± 0.42	17.55 ± 0.38	19.24 ± 0.23

TABLE III
MSE PERFORMANCE OF CPD-BASED MODELS.

DATASET	AMP [28]	CSID [14]	FSA-HTF (THIS PAPER)
QSAR	1.65 ± 0.50	1.48 ± 0.17	1.38 ± 0.53
EE [1]	0.31 ± 0.06	0.18 ± 0.02	$\mathbf{0.16 \pm 0.10}$
EE [2]	0.46 ± 0.10	0.34 ± 0.06	$\mathbf{0.24 \pm 0.09}$
ASN	8.73 ± 1.29	$\mathbf{3.05 \pm 0.53}$	4.01 ± 0.32
SMT	0.99 ± 0.05	0.98 ± 0.06	$\mathbf{0.94 \pm 0.06}$
AB	4.75 ± 0.3	4.95 ± 0.21	4.52 ± 0.30
CPP	N/A	15.25 ± 0.45	15.00 ± 0.66
SC	N/A	N/A	127.76 ± 11.86
PP	N/A	18.21 ± 0.45	$\mathbf{16.77 \pm 0.35}$

TABLE IV
AVERAGE EXECUTION TIME PER MONTE-CARLO SIMULATION IN SECONDS.

Dataset	NNET	FSA-HTF
QSAR	96	103
EE [1]	267	174
EE [2]	256	209
ASN	294	206
SMT	193	657
AB	385	244
CPP	625	211
SC	1521	4549
PP	3434	5120

method are chosen using 5-fold cross-validation and we report the average MSE of 5 Monte-Carlo simulations. When training the PR model, we test 2nd and 3rd order polynomials for all datasets except for SC due to the high computational cost. For the neural network model, we vary the number of layers from 1 to 5 and the number of nodes for each layer from 10 to 500. For FSA-HTF we vary the number of coefficients per mode from 5 to 50 and the rank from 5 to 30. We used alternating least squares for all datasets except for SC and PP for which we used SGD due to the larger number of samples. All the baselines except for CSID were implemented using the sklearn library [37].

The performance of the standard baseline models is shown in Table II and the performance of the tensor based models in Table III. Among the standard baselines, the NNET approach works the best followed by the DT. The linear and polynomial models do not perform as well as they are usually too restrictive in practice, especially for larger datasets. Overall, we observe that the performance of our proposed method is very satisfactory, outperforming the baselines in 4 datasets and performing comparable to the winning method in the remaining ones. Our method outperforms the neural network in the smaller datasets (QSAR, EE, ASB and SMT) but also in two of the larger datasets (CPP, PP).

Compared to the tensor methods, FSA-HTF performs better in all except one dataset. AMP is a more complex method which assumes that each component $f_{n,r}(\cdot)$ belongs to a RKHS. It requires solving a kernel regression for each n and r at each step with a complexity of $O(M^3)$ making it not suitable even for datasets of moderate size. We did not run AMP for the larger datasets (CPP, SC, PP) since the computational complexity was very high and the algorithm did not terminate in a reasonable amount of time (> 1 day of training including cross validation). CSID is more suitable for datasets with discrete input as it requires a discretization step otherwise which leads to performance degradation. As shown in table III, it performs worse than FSA-HTF in all but one datasets which may be due to the function not being sufficiently smooth.

In table IV we report the average execution time of 5 Monte-Carlo simulations (including cross validation) for the neural network and the FSA-HTF algorithm. The average execution times of the two methods are comparable (same order of magnitude) although we haven't optimized our algorithms to take advantage of the algebraic structure of our model.

VI. Conclusion

In this paper, we proposed a generalized CPD representation of functions which follows from compactness of support and continuous differentiability. Our method approximates a compactly supported multivariate function using a tensor of truncated multidimensional Fourier series coefficients and under a low-rank assumption in the Fourier domain, provides correct identification of the unknown function. We proposed two learning algorithms: one that explicitly constructs and decomposes the Fourier tensor (WFSA-LR), and another that learns the CPD model indirectly, without instantiating the Fourier tensor (FSA-HTF). In the synthetic data experiments, we observed that when the true function is indeed low-rank, WFSA-LR requires more data but eventually both methods converge to the same solution. For real data, FSA-HTF is more suitable since the conditions assumed by WFSA-LR are violated. Our algorithm FSA-HTF demonstrated promising results on real multivariate regression tasks with tensors of high-order using very low ranks.

References

[1] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[3] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis," *UCLA Working Papers Phonetics*, vol. 16, pp. 1–84, 1970.

[4] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, July 2017.

[5] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Tensors for data mining and data fusion: Models, applications, and scalable algorithms," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 2, pp. 1–44, 2016.

[6] N. Kargas, N. D. Sidiropoulos, and X. Fu, "Tensors, learning, and "Kolmogorov extension" for finite-alphabet random vectors," *IEEE Transactions on Signal Processing*, vol. 66, no. 18, pp. 4854–4868, 2018.

[7] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos, "Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries," in *Proceedings of the 18th International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 316–324.

[8] N. Vervliet, O. Debals, L. Sorber, and L. De Lathauwer, "Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 71–79, 2014.

[9] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016, pp. 2071–2080.

[10] T. Lacroix, N. Usunier, and G. Obozinski, "Canonical tensor decomposition for knowledge base completion," in *International Conference on Machine Learning*, 2018, pp. 2863–2872.

[11] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering," in *Proceedings of the 4th ACM Conference on Recommender Systems*, 2010, pp. 79–86.

[12] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned CP-decomposition," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[13] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 442–450.

[14] N. Kargas and N. D. Sidiropoulos, "Nonlinear system identification via tensor completion," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020.

[15] S. Efromovich, *Nonparametric curve estimation: methods, theory, and applications*. Springer Science & Business Media, 2008.

[16] S. Rendle, "Factorization machines," in *Proceedings of the IEEE International Conference on Data Mining*, Dec 2010, pp. 995–1000.

[17] M. Blondel, M. Ishihata, A. Fujino, and N. Ueda, "Polynomial networks and factorization machines: New insights and efficient training algorithms," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016, pp. 850–858.

[18] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata, "Higher-order factorization machines," in *Advances in Neural Information Processing Systems*, 2016, pp. 3351–3359.

[19] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[20] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[21] A. Novikov, M. Trofimov, and I. Oseledets, "Exponential machines," in *International Conference on Learning Representations Workshop*, 2016.

[22] I. Perros, F. Wang, P. Zhang, P. Walker, R. Vuduc, J. Pathak, and J. Sun, "Polyadic regression and its application to chemogenomics," in *Proceedings of the SIAM International Conference on Data Mining*, 2017, pp. 72–80.

[23] P. Dreesen, M. Ishteva, and J. Schoukens, "Decoupling multivariate polynomials using first-order information and tensor decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 2, pp. 864–879, 2015.

[24] P. Comon, Y. Qi, and K. Usevich, "Identifiability of an x-rank decomposition of polynomial maps," *SIAM Journal on Applied Algebra and Geometry*, vol. 1, no. 1, pp. 388–414, 2017.

[25] S. Wahls, V. Koivunen, H. V. Poor, and M. Verhaegen, "Learning multidimensional fourier series with tensor trains," in *Proceedings of the IEEE Global Conference on Signal and Information Processing*, 2014, pp. 394–398.

[26] V. Khrulkov, A. Novikov, and I. V. Oseledets, "Expressive power of recurrent neural networks," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[27] M. Signoretto, L. De Lathauwer, and J. A. Suykens, "Learning tensors in reproducing kernel hilbert spaces with multilinear spectral penalties," *arXiv preprint*, 2013.

[28] T. Suzuki, H. Kanagawa, H. Kobayashi, N. Shimizu, and Y. Tagami, "Minimax optimal alternating minimization for kernel nonparametric tensor learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 3783–3791.

[29] G. Plonka, D. Potts, G. Steidl, and M. Tasche, *Numerical Fourier Analysis*. Springer, 2018.

[30] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics, 2009.

[31] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[32] S. E. Leurgans, R. T. Ross, and R. B. Abel, "A decomposition for three-way arrays," *SIAM Journal on Matrix Analysis and Applications*, vol. 14, no. 4, pp. 1064–1083, 1993.

[33] M. Boussé, N. Vervliet, I. Domanov, O. Debals, and L. De Lathauwer, "Linear systems with a canonical polyadic decomposition constrained solution: Algorithms and applications," *Numerical Linear Algebra with Applications*, vol. 25, no. 6, p. e2190, 2018.

[34] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[35] L. Chiantini and G. Ottaviani, "On generic identifiability of 3-tensors of small rank," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 3, pp. 1018–1037, 2012.

[36] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Nikos Kargas received the Diploma and M.Sc. degree in electronic and computer engineering from the Technical University of Crete (TUC), Chania, Greece, in 2013 and 2015, respectively, and Ph.D. degree in electrical engineering from University of Minnesota, Minneapolis, in 2020. His research interests include machine learning, statistics and optimization.

Nicholas D. Sidiropoulos (Fellow, IEEE) received the Diploma in electrical engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park, College Park, MD, USA, in 1988, 1990, and 1992, respectively. He has served on the faculty of the University of Virginia, the University of Minnesota, and the Technical University of Crete, Greece, prior to his current appointment as Louis T. Rader Professor and Chair of ECE at UVA. From 2015 to 2017, he was an ADC Chair Professor with the University of Minnesota. His research interests are in signal processing, communications, optimization, tensor decomposition, and factor analysis, with applications in machine learning and communications. He received the NSF/CAREER award in 1998, the IEEE Signal Processing Society (SPS) Best Paper Award in 2001, 2007, and 2011, served as IEEE SPS Distinguished Lecturer (2008–2009), and as Vice President - Membership of IEEE SPS (2017–2019). He received the 2010 IEEE Signal Processing Society Meritorious Service Award, and the 2013 Distinguished Alumni Award from the University of Maryland, Department of ECE. He is a fellow of EURASIP (2014).