algorithms

MDPI

Article

Tensor-Based Algorithms for Image Classification

Stefan Klus *,† and Patrick Gelß *,†

Department of Mathematics and Computer Science, Freie Universität Berlin, 14195 Berlin, Germany
* Correspondence: stefan.klus@fu-berlin.de (S.K.); p.gelss@fu-berlin.de (P.G.)
† These authors contributed equally to this work.

Received: 20 October 2019; Accepted: 7 November 2019; Published: 9 November 2019

Abstract: Interest in machine learning with tensor networks has been growing rapidly in recent years. We show that tensor-based methods developed for learning the governing equations of dynamical systems from data can, in the same way, be used for supervised learning problems and propose two novel approaches for image classification. One is a kernel-based reformulation of the previously introduced multidimensional approximation of nonlinear dynamics (MANDy), the other an alternating ridge regression in the tensor train format. We apply both methods to the MNIST and fashion MNIST data set and show that the approaches are competitive with state-of-the-art neural network-based classifiers.

Keywords: quantum machine learning; image classification; tensor train format; kernel-based methods; ridge regression

MSC: 15A69; 62J07; 65D18; 68Q32

1. Introduction

Tensor-based methods have become a powerful tool for scientific computing over the last years. In addition to many application areas, such as quantum mechanics and computational dynamics, where low-rank tensor approximations have been successfully applied, using tensor networks for supervised learning has gained a lot of attention recently. In particular, the canonical format and the tensor train format have been considered for quantum machine learning (There are different research directions in the field of quantum machine learning, here we understand it as using quantum computing capabilities for machine learning problems.) problems, see, e.g., [1–3]. A tensor-based algorithm for image classification using sweeping techniques inspired by the density matrix renormalization group (DMRG) [4] was proposed in [5,6] and further discussed in [7,8]. Interestingly, also researchers at Google are currently developing a tensor-based machine learning framework called "TensorNetwork (http://github.com/google/TensorNetwork)" [9,10]. The goal is to expedite the adoption of such methods by the machine learning community.

Our goal is to show that recently developed methods for recovering the governing equations of dynamical systems can be generalized in such a way that they can also be used for supervised learning tasks, e.g., classification problems. To learn the governing equations from simulation or measurement data, regression methods such as sparse identification of nonlinear dynamics (SINDy) [11,12] and its tensor-based reformulation multidimensional approximation of nonlinear dynamics (MANDY) [13] can be applied. The main challenge is often to choose the right function space from which the system representation is learned. Although SINDy and MANDy essentially select functions from a potentially large set of basis functions by applying regularized regression methods, other approaches allow nested

functions and typically result in nonlinear optimization problems, which are then frequently solved using (stochastic) gradient descent. By constructing a basis comprising tensor products of simple functions (e.g., functions depending only on one variable), extremely high-dimensional feature spaces can be generated.

In this work, we explain how to compute the pseudoinverse required for solving the minimization problem directly in the tensor train (TT) format, i.e., we replace the iterative approach from [5,6] by a direct computation of the least-squares solution and point out similarities with the aforementioned system identification methods. The reformulated algorithm can be regarded as a *kernelized* variant of MANDy, where the kernel is based on tensor products. This is also related to quantum machine learning ideas: As pointed out in [14], the basic idea of quantum computing is similar to kernel methods in that computations are performed implicitly in otherwise intractably large Hilbert spaces. Although kernel methods were popular in the 1990s, the focus of the machine learning community has shifted to deep neural networks in recent years [14]. We will show that, for simple image classification tasks, kernels based on tensor products are competitive with neural networks.

In addition to the kernel-based approach, we propose another DMRG-inspired method for the construction of TT decompositions of weight matrices containing the coefficients for the selected basis functions. Instead of computing pseudoinverses, a core-wise ridge regression [15] is applied to solve the minimization problem. Although the approach introduced in [5,6] only involves tensor contractions corresponding to single images of the training data set, we use TT representations of transformed data tensors, see [13,16], to include the entire training data set at once for constructing low-dimensional systems of linear equations. Combining an efficient computational scheme for the corresponding subproblems and truncated singular value decompositions [17], we call the resulting algorithm alternating ridge regression (ARR) and discuss connections to MANDy and other regularized regression techniques.

Although we describe the classification problems using the example of the iconic MNIST data set [18] and the fashion MNIST data set [19], the derived algorithms can be easily applied to other classification problems. There is a plethora of kernel and deep learning methods for image classification; a list of the most successful methods for the MNIST and fashion MNIST data sets including nearest-neighbor heuristics, support vector machines, and convolutional neural networks can be found on the respective website (http://yann.lecun.com/exdb/mnist/, http://github.com/zalandoresearch/fashion-mnist). We will not review these methods in detail, but instead focus on relationships with data-driven methods for analyzing dynamical systems. The main contributions of this paper are as follows.

- Extension of MANDy: We show that the efficacy of the pseudoinverse computation in the tensor train format can be improved by eliminating the need to left- and right-orthonormalize the tensor. Although this is a straightforward modification of the original algorithm, it enables us to consider large data sets. The resulting method is closely related to kernel ridge regression.
- Alternating ridge regression: We introduce a modified TT representation of transformed data tensors for the development of a tensor-based regression technique which computes low-rank representations of coefficient tensors. We show that it is possible to obtain results which are competitive with those computed by MANDy and, at the same time, reduce the computational costs and the memory consumption significantly.
- Classification of image data: Although originally designed for system identification, we apply these methods to classification problems and visualize the learned classifier, which allows us to interpret features detected in the images.

The remainder is structured as follows. In Section 2, we describe methods to learn governing equations of dynamical systems from data as well as a tensor-based iterative scheme for image classification and highlight their relationships. In Section 3, we describe how to apply MANDy to classification problems

and introduce the ARR approach based on the alternating optimization of TT cores. Numerical results are presented in Section 4, followed by a brief summary and conclusion in Section 5.

2. Prerequisites

We will introduce the original MNIST and the fashion MNIST data set, which will serve as guiding examples. Afterwards, SINDy and MANDy, as well as tensor-based methods for image classification problems, will be briefly discussed. In what follows, we will use the notation summarized in Table 1.

Table 1. Notation used in this work.

Symbol	Description
$X = \left[x^{(1)}, \ldots, x^{(m)}\right]$	data matrix in $\mathbb{R}^{d \times m}$
$Y = \left[y^{(1)}, \ldots, y^{(m)}\right]$	label matrix in $\mathbb{R}^{d' \times m}$
$n_1, \ldots, n_p$	mode dimensions of tensors
$r_0, \ldots, r_p$	ranks of tensor trains
$\psi_1, \ldots, \psi_p$	basis functions $\psi_\mu \colon \mathbb{R}^d \to \mathbb{R}^{n_\mu}$
Ψ_X / $\mathbf{\Psi}_X$	transformed data matrices/tensors
Ξ / $\mathbf{\Xi}$	coefficient matrices/tensors

2.1. MNIST and Fashion MNIST

The MNIST data set [18], see Figure 1a, contains grayscale (The methods described below can be easily extended to color images by defining basis functions for each primary color.) images of handwritten digits and the associated labels. The data set is split into 60,000 images for training and 10,000 images for testing. Each image is of size 28×28. Let $d = 784$ be the number of pixels of one image, and let the images, reshaped as vectors, be denoted by $x^{(j)} \in \mathbb{R}^d$ and the corresponding labels by $y^{(j)} \in \mathbb{R}^{d'}$, where $d' = 10$ is the number of different classes. Each label encodes a number in $\{0, \ldots, 9\}$, and the entries $y_i^{(j)}$ of the vector $y^{(j)}$ are given by

$$y_i^{(j)} = \begin{cases} 1, & \text{if } x^{(j)} \text{ contains the number } i-1, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

i.e., $y^{(j)} = [1, 0, 0, \ldots, 0]^\top$ represents 0, $y^{(j)} = [0, 1, 0, \ldots, 0]^\top$ represents 1, etc. This is also called one-hot encoding in machine learning.

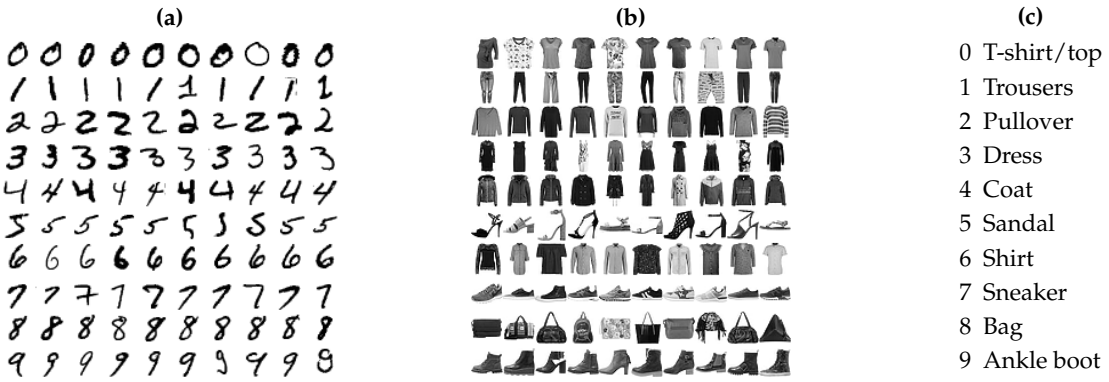

Figure 1. (**a**) Samples of the MNIST data set. (**b**) Samples of the fashion MNIST data set. Each row represents a different item type. (**c**) Corresponding labels for the fashion MNIST data set.

The fashion MNIST data set [19] can be regarded as a shoo-in replacement for the original data set. There are again 60,000 training and 10,000 test images of size 28×28. Some samples are shown in Figure 1b and the corresponding labels in Figure 1c. Given a picture of a clothing item, the goal now is to identify the correct category, which is encoded as described above.

2.2. SINDy

SINDy [11] was originally developed to learn the governing equations of dynamical systems from data. We will show how it can, in the same way, be used for classification problems. Consider an autonomous ordinary differential equation of the form $\dot{x} = f(x)$, with $f\colon \mathbb{R}^d \to \mathbb{R}^d$. Given m measurements of the state of the system, denoted by $x^{(j)}$, $j = 1, \ldots, m$, and the corresponding time derivatives $y^{(j)} := \dot{x}^{(j)}$, the goal is to reconstruct the function f from the measurement data. Let $X = [x^{(1)}, \ldots, x^{(m)}] \in \mathbb{R}^{d \times m}$ and $Y = [y^{(1)}, \ldots, y^{(m)}] \in \mathbb{R}^{d \times m}$. That is, $d' = d$ in this case. To represent f, we select a vector-valued basis function $\Psi\colon \mathbb{R}^d \to \mathbb{R}^n$ and define the transformed data matrix $\Psi_X \in \mathbb{R}^{n \times m}$ by

$$\Psi_X = \begin{bmatrix} \Psi(x^{(1)}) & \ldots & \Psi(x^{(m)}) \end{bmatrix}. \tag{2}$$

Omitting sparsity constraints, SINDy then boils down to solving

$$\min_{\Xi} \left\| Y - \Xi^\top \Psi_X \right\|_F, \tag{3}$$

where

$$\Xi = \begin{bmatrix} \xi_1 & \ldots & \xi_d \end{bmatrix} \in \mathbb{R}^{n \times d} \tag{4}$$

is the coefficient matrix. Each column vector ξ_i then represents a function f_i, i.e.,

$$y_i^{(j)} \approx f_i(x^{(j)}) = \xi_i^\top \Psi(x^{(j)}). \tag{5}$$

We thus obtain a model of the form $\dot{x} = \Xi^\top \Psi(x)$, which approximates the possibly unknown dynamics. The solution of the minimization problem (3) with minimal Frobenius norm is given by

$$\Xi^\top = Y \Psi_X^+, \tag{6}$$

where $^+$ denotes the pseudoinverse, see [20].

2.3. Tensor-Based Learning

We will now briefly introduce the basic concepts of tensor decompositions and tensor formats as well as the tensor-based reformulation of SINDy, called MANDy, proposed in [13]. Additionally, recently introduced methods for supervised learning with tensor networks will be discussed.

2.3.1. Tensor Decompositions

To mitigate the curse of dimensionality when working with tensors $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_p}$, where $n_\mu \in \mathbb{N}$, we will exploit low-rank tensor approximations. The simplest approximation of a tensor of order p is a rank-one tensor, i.e., a tensor product of p vectors given by

$$\mathbf{T} = T^{(1)} \otimes T^{(2)} \otimes \cdots \otimes T^{(p)}, \tag{7}$$

where $T^{(\mu)}$, $\mu = 1, \dots, p$, are vectors in $\mathbb{R}^{n_\mu}$. If a tensor is written as the sum of r rank-one tensors, i.e.,

$$\mathbf{T} = \sum_{k=1}^{r} T^{(1)}_{:,k} \otimes T^{(2)}_{:,k} \otimes \cdots \otimes T^{(p)}_{:,k}, \tag{8}$$

with $T^{(\mu)} \in \mathbb{R}^{n_\mu \times r}$, this results in the so-called canonical format. In fact, any tensor can be expressed in this format, but we are particularly interested in low-rank representations of tensors in order to reduce the storage consumption as well as the computational costs. The same requirement applies to tensors expressed in the tensor train format (TT format), where a high-dimensional tensor is represented by a network of multiple low-dimensional tensors [21,22]. A tensor $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_p}$ is said to be in the TT format if

$$\mathbf{T} = \sum_{k_0=1}^{r_0} \cdots \sum_{k_p=1}^{r_p} \mathbf{T}^{(1)}_{k_0,:,k_1} \otimes \cdots \otimes \mathbf{T}^{(p)}_{k_{p-1},:,k_p}. \tag{9}$$

The tensors $\mathbf{T}^{(\mu)} \in \mathbb{R}^{r_{\mu-1} \times n_\mu \times r_\mu}$ of order 3 are called TT cores. The numbers r_μ are called *TT ranks* and have a strong influence on the expressivity of a tensor train. It holds that $r_0 = r_p = 1$ and $r_\mu \geq 1$ for $\mu = 1, \dots, p-1$. Figure 2a shows the graphical representation of a tensor train, which is also called Penrose notation, see [23].

The left- and right-unfoldings of a TT core $\mathbf{T}^{(\mu)}$ are given by the matrices

$$\mathcal{L}_\mu = \mathbf{T}^{(\mu)} \Big|^{r_\mu}_{r_{\mu-1}, n_\mu} \in \mathbb{R}^{(r_{\mu-1} \cdot n_\mu) \times r_\mu} \quad \text{and} \quad \mathcal{R}_\mu = \mathbf{T}^{(\mu)} \Big|^{n_\mu, r_\mu}_{r_{\mu-1}} \in \mathbb{R}^{r_{\mu-1} \times (n_\mu \cdot r_\mu)}, \tag{10}$$

respectively. Here, the indices of two modes of $\mathbf{T}^{(\mu)}$ are lumped into a single row or column index, whereas the remaining mode forms the other dimension of the unfolding matrix. We call the TT core $\mathbf{T}^{(\mu)}$ left-orthonormal if its left-unfolding is orthonormal with respect to the rows, i.e., $\mathcal{L}_\mu^\top \cdot \mathcal{L}_\mu = \mathrm{Id} \in \mathbb{R}^{r_\mu \times r_\mu}$. Correspondingly, a core is called right-orthonormal if its right-unfolding is orthonormal with respect to the columns, i.e., $\mathcal{R}_\mu \cdot \mathcal{R}_\mu^\top = \mathrm{Id} \in \mathbb{R}^{r_{\mu-1} \times r_{\mu-1}}$. In Penrose notation, orthonormal components are depicted by half-filled circles, cf. Figure 2b, where a tensor train with left-orthonormal cores is shown.

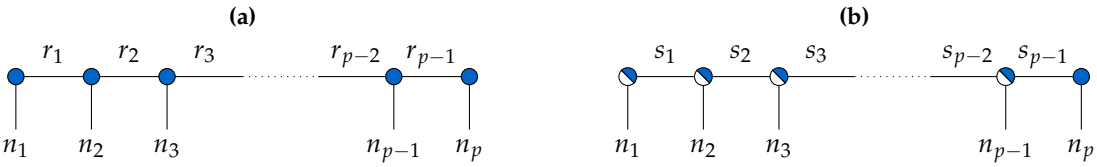

Figure 2. Graphical representation of tensor trains: (**a**) A core is depicted by a circle with different arms indicating the modes of the tensor and the rank indices. The first and the last tensor train (TT) core are regarded as matrices due to the fact that $r_0 = r_p = 1$. (**b**) Left-orthonormalized tensor train obtained by, e.g., sequential singular value decompositions (SVDs). Note that the TT ranks may change due to orthonormalization, e.g., when using (reduced/truncated) SVDs.

A given TT core can be left- or right-orthonormalized, respectively, by computing a singular value decomposition (SVD) of its unfolding. For instance, the components of an SVD of the form $\mathcal{L}_\mu = U \cdot \Sigma \cdot V^\top$ can be interpreted as a left-orthonormalized version of $\mathbf{T}^{(\mu)}$ coupled with the matrices Σ and $V^\top$. When we talk about, e.g., left-orthonormalization of the cores of a tensor train, we mean the application of sequential SVDs from left to right (also called *HOSVD*, cf. [24]) where U builds the updated core, while the non-orthonormal part $\Sigma \cdot V^\top$ is contracted with the subsequent TT core. As described in [13,16,25], left- and right-orthonormalization can be used to construct pseudoinverses of tensors. The general

idea is to construct a global SVD of a given tensor train by left- and right-orthonormalizing its cores. However, in Section 3.2, we will exploit the structure of transformed data tensors, as introduced in [13], to propose a different method for the construction of pseudoinverses, which significantly reduces the computational effort.

We also represent TT cores as two-dimensional arrays containing vectors as elements. In this notation, a single core of a tensor train $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_p}$ is written as

$$\left[\!\left[\mathbf{T}^{(\mu)} \right]\!\right] = \begin{bmatrix}\!\!\begin{bmatrix} \mathbf{T}^{(\mu)}_{1,:,1} & \cdots & \mathbf{T}^{(\mu)}_{1,:,r_\mu} \end{bmatrix}\!\! \\ \vdots \\ \begin{bmatrix} \mathbf{T}^{(\mu)}_{r_{\mu-1},:,1} & \cdots & \mathbf{T}^{(\mu)}_{r_{\mu-1},:,r_\mu} \end{bmatrix}\!\! \end{bmatrix}. \tag{11}$$

Then, the expression $\mathbf{T} = \left[\!\left[\mathbf{T}^{(1)} \right]\!\right] \otimes \cdots \otimes \left[\!\left[\mathbf{T}^{(p)} \right]\!\right]$ is used for representing tensor trains $\mathbf{T}$, cf. [13,26,27].

2.3.2. MANDy

MANDy [13] is a tensorized version of SINDy and constructs counterparts of the transformed data matrices (2) directly in the TT format. Two different types of decompositions, namely, the coordinate- and the function-major decomposition, were introduced in [13]. In [16], the technique for the construction of the transformed data tensors was generalized to arbitrary lists of basis functions. This will be explained in more detail in Section 3.1. Given data matrices $X, Y \in \mathbb{R}^{d \times m}$ and basis functions $\psi_\mu \colon \mathbb{R}^d \to \mathbb{R}^{n_\mu}$, $\mu = 1, \ldots, p$, the tensor-based representation of the corresponding transformed data tensors $\mathbf{\Psi}_X \in \mathbb{R}^{n_1 \times \cdots \times n_p \times m}$ enables us to solve the reformulated minimization problem

$$\min_{\mathbf{\Xi}} \left\| Y - \mathbf{\Xi}^\top \mathbf{\Psi}_X \right\|_F \tag{12}$$

so that the coefficients are given in form of a tensor train $\mathbf{\Xi} \in \mathbb{R}^{n_1 \times \cdots \times n_p \times d}$, cf. Section 2.2. Instead of identifying the governing equations of dynamical systems from data, see [13], we seek to classify images using MANDy. The only difference is that $\mathbf{\Psi}_X$ now contains the transformed images and Y the corresponding labels. As the matrix Y may have different dimensions than X, i.e., $Y \in \mathbb{R}^{d' \times m}$, the aim is to find the optimal solution of (12) in the form of a tensor train $\mathbf{\Xi} \in \mathbb{R}^{n_1 \times \cdots \times n_p \times d'}$. We will discuss the explicit representation of transformed data tensors and their pseudoinversion in Section 3.

2.3.3. Supervised Learning with Tensor Networks

It has been shown in [5,6] that tensor-based optimization schemes can be adapted to supervised learning problems. A given input vector x is mapped into a higher-dimensional space using a feature map $\mathbf{\Psi}$ before being classified by a decision function $f \colon \mathbb{R}^d \to \mathbb{R}^{d'}$ of the form

$$f(x) = \mathbf{\Xi}^\top \mathbf{\Psi}(x), \tag{13}$$

where $\mathbf{\Xi}$ is a coefficient tensor in TT format. The ith entry of the vector $f(x)$ then represents the likelihood that the image x belongs to the class with label $i - 1$. The transformation defined in [5,6] reads as follows,

$$\mathbf{\Psi}(x) = \begin{bmatrix} \cos(\alpha\, x_1) \\ \sin(\alpha\, x_1) \end{bmatrix} \otimes \begin{bmatrix} \cos(\alpha\, x_2) \\ \sin(\alpha\, x_2) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \cos(\alpha\, x_d) \\ \sin(\alpha\, x_d) \end{bmatrix}, \tag{14}$$

where α is a parameter. However, the originally proposed choice of $\alpha = \frac{\pi}{2}$ is often not optimal. This will be discussed in more detail below. The function $\mathbf{\Psi}$ assigns each pixel of the image a two-dimensional vector, inspired by the spin-vectors encountered in quantum mechanics [6]. It was illustrated in [14] how such a transformation can be implemented as a quantum feature map, where the information is encoded in the amplitudes of qubits. Embedding data into quantum Hilbert spaces might be interesting in cases where the quantum device evaluates kernels faster or where kernels cannot be simulated by classical computers anymore [14].

Due to the tensor structure, $\mathbf{\Psi}(x)$ is a tensor with 2^d entries, which, for the original MNIST image size, amounts to $n \approx 10^{236}$ basis functions. In [5,6], the image size is first reduced to 14×14 pixels by averaging groups of four pixels, which then results in "only" $n \approx 10^{59}$ basis functions. Thus, storing the full coefficient matrix is clearly infeasible since $\mathbf{\Xi} \in \mathbb{R}^{2 \times \cdots \times 2 \times d'} \cong \mathbb{R}^{n \times d'}$. Here, d' appears as an additional tensor index since the decision function is computed for all d' labels simultaneously.

To learn the tensor $\mathbf{\Xi}$ from training data, a DMRG/ALS-related algorithm (cf. [4,28]) that sweeps back and forth along the cores and iteratively minimizes the cost function

$$\min_{\mathbf{\Xi}} \sum_{j=1}^{m} \left\| y^{(j)} - \mathbf{\Xi}^\top \mathbf{\Psi}(x^{(j)}) \right\|_2^2 \tag{15}$$

is devised. The suggested algorithm varies two neighboring cores at the same time, which allows for adapting the tensor ranks, and computes an update using a gradient descent step. The tensor ranks are reduced by truncated SVDs to control the computational costs. The truncation of the TT ranks can also be interpreted as a form of regularization. For more details, we refer to [5,6].

Different techniques to improve the original algorithm presented in [5] were proposed. In [29], the image data is preprocessed using a discrete cosine transformation and the ordering of the pixels is optimized in order to reduce the ranks. In [10], the DMRG-based sweeping method was replaced by a stochastic gradient descent approach, where the gradient is computed with the aid of automatic differentiation. Furthermore, it was shown that GPUs allow for an efficient solution of such problems.

3. Tensor-Based Classification Algorithms

We will now describe two different tensor-based classification approaches. First, we show how to combine MANDy with kernel-based regression techniques, so as to derive an efficient method for the computation of the pseudoinverse of the transformed data tensor. Then, a classification algorithm based on the alternating optimization of the TT cores of the coefficient tensor is proposed.

3.1. Basis Decomposition

As above, let $x \in \mathbb{R}^d$ be a vector and $\psi_\mu \colon \mathbb{R}^d \to \mathbb{R}^{n_\mu}$, $\mu = 1, \ldots, p$, basis functions. We consider the rank-one tensors

$$\mathbf{\Psi}(x) = \psi_1(x) \otimes \cdots \otimes \psi_p(x) = \begin{bmatrix} \psi_{1,1}(x) \\ \vdots \\ \psi_{1,n_1}(x) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \psi_{p,1}(x) \\ \vdots \\ \psi_{p,n_p}(x) \end{bmatrix} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}. \tag{16}$$

For m different vectors stored in a data matrix $X = [x^{(1)}, \ldots, x^{(m)}] \in \mathbb{R}^{d \times m}$, we must construct transformed data tensors $\mathbf{\Psi}_X \in \mathbb{R}^{n_1 \times \cdots \times n_p \times m}$ with $(\mathbf{\Psi}_X)_{:,\ldots,:,j} = \mathbf{\Psi}(x^{(j)})$. In [13,16], this was achieved by multiplying (with the aid of the tensor product) the rank-one decompositions given in (16) for all vectors,

$x^{(1)}, \ldots, x^{(m)}$, by additional unit vectors and subsequently summing them up. The transformed data tensor can then be represented using the following canonical/TT decompositions,

$$\begin{aligned}
\boldsymbol{\Psi}_X &= \sum_{j=1}^{m} \boldsymbol{\Psi}\left(x^{(j)}\right) \otimes e_j \\
&= \sum_{j=1}^{m} \psi_1\left(x^{(j)}\right) \otimes \cdots \otimes \psi_p\left(x^{(j)}\right) \otimes e_j \\
&= \left[\!\!\left[\psi_1\left(x^{(1)}\right) \quad \cdots \quad \psi_1\left(x^{(m)}\right) \right]\!\!\right] \otimes \left[\!\!\left[\begin{matrix} \psi_2\left(x^{(1)}\right) & & 0 \\ & \ddots & \\ 0 & & \psi_2\left(x^{(m)}\right) \end{matrix} \right]\!\!\right] \otimes \cdots \\
&\quad \cdots \otimes \left[\!\!\left[\begin{matrix} \psi_p\left(x^{(1)}\right) & & 0 \\ & \ddots & \\ 0 & & \psi_p\left(x^{(m)}\right) \end{matrix} \right]\!\!\right] \otimes \left[\!\!\left[\begin{matrix} e_1 \\ \vdots \\ e_m \end{matrix} \right]\!\!\right] \\
&= \left[\!\!\left[\boldsymbol{\Psi}_X^{(1)} \right]\!\!\right] \otimes \cdots \otimes \left[\!\!\left[\boldsymbol{\Psi}_X^{(p+1)} \right]\!\!\right],
\end{aligned} \tag{17}$$

where e_j, $j = 1, \ldots, m$, denote the unit vectors of the standard basis in the m-dimensional Euclidean space. An entry of $\boldsymbol{\Psi}_X$ is given by

$$(\boldsymbol{\Psi}_X)_{i_1,\ldots,i_p,j} = \psi_{1,i_1}\left(x^{(j)}\right) \cdot \ldots \cdot \psi_{p,i_p}\left(x^{(j)}\right), \tag{18}$$

for $1 \leq i_k \leq n_k$ and $1 \leq j \leq m$. Thus, the matrix-based counterpart of $\boldsymbol{\Psi}_X$, see (2), would be given by the mode-p unfolding

$$\Psi_X = \boldsymbol{\Psi}_X \Big|^{m}_{n_1,\ldots,n_p}. \tag{19}$$

That is, the modes $n_1, \ldots, n_p$ represent row indices of the unfolding, and mode m is the column index. However, for the purpose of this paper, we modify the representation of our transformed data tensors. First, realize that the last core of the TT representation in (17) can be neglected, as it is only a reshaped identity matrix. The result is then a tensor network with an "open arm", which can be regarded as a tensor train with an additional column mode located at the last core, see Figure 3a. Second, this additional mode can be shifted to any TT core of the decomposition. This is shown in Figure 3b. We will benefit from these modifications in Section 3.3 when constructing the subproblems for the ALS-inspired approach. Consider the TT decomposition $\widehat{\boldsymbol{\Psi}}_X$ given by

$$\widehat{\boldsymbol{\Psi}}_X = \left[\!\!\left[\boldsymbol{\Psi}_X^{(1)} \right]\!\!\right] \otimes \cdots \otimes \left[\!\!\left[\boldsymbol{\Psi}_X^{(p-1)} \right]\!\!\right] \otimes \left[\!\!\left[\begin{matrix} \psi_p\left(x^{(1)}\right) \\ \vdots \\ \psi_p\left(x^{(m)}\right) \end{matrix} \right]\!\!\right]. \tag{20}$$

Note that this tensor is an element of the tensor space $\mathbb{R}^{n_1 \times \cdots \times n_p}$, i.e., $\widehat{\boldsymbol{\Psi}}_X$ has no additional column dimension, and it holds that

$$\widehat{\boldsymbol{\Psi}}_X \Big|_{n_1,\ldots,n_p} = \Psi_X \cdot [1, \ldots, 1]^\top. \tag{21}$$

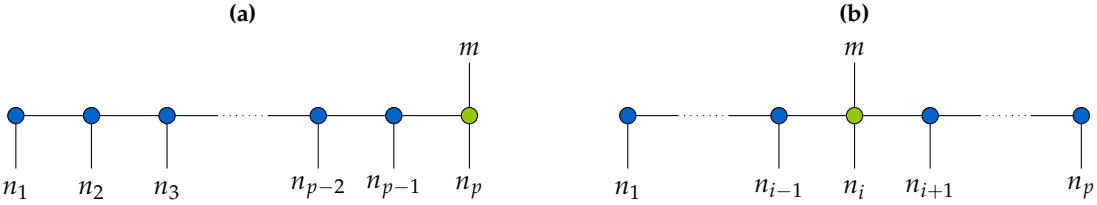

Figure 3. TT representation of transformed data tensors: (**a**) As in [13], the first p cores (blue circles) are given by (17). The direct contraction of the two last TT cores in (17) can be regarded as an operator-like TT core with a row and column mode (green circle). (**b**) The additional column mode can be shifted to any of the p TT cores.

Now, we define $\widehat{\mathbf{\Psi}}_{X,\mu} \in \mathbb{R}^{n_1 \times \cdots \times n_p \times m}$ to be the tensor derived from $\widehat{\mathbf{\Psi}}_X$ by replacing the μth core by

$$\widehat{\mathbf{\Psi}}_{X,\mu}^{(\mu)} = \begin{bmatrix}\begin{bmatrix} | & 0 & \cdots & 0 \\ \psi_\mu(x_1) & \vdots & & \vdots \\ | & 0 & \cdots & 0 \end{bmatrix} & & & 0 \\ & \ddots & & \\ 0 & & & \begin{bmatrix} 0 & \cdots & 0 & | \\ \vdots & & \vdots & \psi_\mu(x_m) \\ 0 & \cdots & 0 & | \end{bmatrix}\end{bmatrix} \in \mathbb{R}^{m \times n_\mu \times m \times m}, \tag{22}$$

where the outer modes correspond to the rank dimensions, whereas the inner modes represent the dimensions of the matrices. Analogously, for the first and the last core of $\widehat{\mathbf{\Psi}}_{X,\mu}$ the nondiagonal core structure has to be used. The 4-dimensional TT core (22) naturally represents a component of a TT operator. In what follows, we will not need to store the whole TT core given in (22). Otherwise, this would mean that we have to save $m^3 \cdot n$ scalar entries (not using a sparse format). However, from a theoretical point of view, $\mathbf{\Psi}_X$ in Figure 3a and $\widehat{\mathbf{\Psi}}_{X,\mu}$ in Figure 3b represent the same tensor in $\mathbb{R}^{n_1 \times \cdots \times n_p \times m}$, see Appendix A.

3.2. Kernel-Based MANDy

Given a training set $X \in \mathbb{R}^{d \times m}$, the corresponding label matrix $Y \in \mathbb{R}^{d' \times m}$, and a set of basis functions $\psi_\mu \colon \mathbb{R}^d \to \mathbb{R}^{n_\mu}$, $\mu = 1, \ldots, p$, we exploit the canonical representation of $\mathbf{\Psi}_X$ given in (17) for kernel-based MANDy. The aim is to solve the optimization problem (12), i.e., we try to find a coefficient tensor $\Xi \in \mathbb{R}^{n_1 \times \cdots \times n_p \times d'}$ such that $\Xi^\top \mathbf{\Psi}_X$ is as close as possible to the corresponding label matrix $Y \in \mathbb{R}^{d' \times m}$. The solution of (12) with minimal Frobenius norm is given by $\Xi^\top = Y \mathbf{\Psi}_X^+$, cf. (6). Note that, compared to standard SINDy/MANDy, the matrix Y here does not necessarily have the same dimensions as X. Due to potentially large ranks of the transformed data tensor $\mathbf{\Psi}_X$, the direct computation of the pseudoinverse using left- and right-orthonormalization, as proposed in [13], would be computationally expensive. However, using the identity $\mathbf{\Psi}_X^+ = (\mathbf{\Psi}_X^\top \mathbf{\Psi}_X)^+ \mathbf{\Psi}_X^\top$, we can rewrite the coefficient tensor as

$$\Xi^\top = Y \left(\mathbf{\Psi}_X^\top \mathbf{\Psi}_X \right)^+ \mathbf{\Psi}_X^\top. \tag{23}$$

The contraction of $\mathbf{\Psi}_X^\top$ and $\mathbf{\Psi}_X$ yields a Gram matrix $G \in \mathbb{R}^{m \times m}$ whose entries are given by the resulting kernel function $k(x, x') = \langle \mathbf{\Psi}(x), \mathbf{\Psi}(x') \rangle$, i.e.,

$$G_{i,j} = k\left(x^{(i)}, x^{(j)} \right) = \left\langle \mathbf{\Psi}\left(x^{(i)} \right), \mathbf{\Psi}\left(x^{(j)} \right) \right\rangle. \tag{24}$$

Note that due to the tensor structure of $\mathbf{\Psi}_X$, we obtain

$$k\left(x^{(i)}, x^{(j)}\right) = \prod_{\mu=1}^{p} \left\langle \psi_\mu\left(x^{(i)}\right), \psi_\mu\left(x^{(j)}\right)\right\rangle, \tag{25}$$

i.e., a product of p local kernels.

Remark 1. *For the basis functions defined in* (14)*, this can be simplified to*

$$k(x, x') = \prod_{i=1}^{d} \cos\left(\alpha(x_i - x_i')\right), \tag{26}$$

which is a product of cosine kernels, cf. [5].

The product structure of the kernel allows us to compute the Gram matrix G as a Hadamard product (denoted by $\odot$) of p matrices, that is,

$$G = \Theta_1 \odot \Theta_2 \odot \cdots \odot \Theta_p, \tag{27}$$

where $\Theta_\mu \in \mathbb{R}^{m\times m}$ is given by

$$\Theta_\mu = \left[\psi_\mu\left(x^{(1)}\right), \ldots, \psi_\mu\left(x^{(m)}\right)\right]^\top \cdot \left[\psi_\mu\left(x^{(1)}\right), \ldots, \psi_\mu\left(x^{(m)}\right)\right]. \tag{28}$$

We now define $Z := YG^+ \in \mathbb{R}^{d'\times m}$, which can be obtained by solving the system $ZG = Y$ (in the least-squares sense if G is singular). The decision function f, cf. (13), is then given by

$$f(x) = \underbrace{Z\mathbf{\Psi}_X^\top}_{=:\Xi^\top}\mathbf{\Psi}(x) = Z\begin{bmatrix} k(x_1, x) \\ \vdots \\ k(x_m, x)\end{bmatrix} \tag{29}$$

and again only requires kernel evaluations. As above, we can use a sequence of Hadamard products to compute $\mathbf{\Psi}_X^\top\mathbf{\Psi}(x)$. The classification problem can thus be solved as summarized in Algorithm 1.

Algorithm 1 Kernel-based MANDy for classification.

Input: Training set X and label matrix Y, test set $\tilde{X}$, basis functions.
Output: Label matrix $\tilde{Y}$.

1: Compute G using (27) and (28).
2: Solve $ZG = Y$.
3: Define the decision function f using (29).
4: Apply f to every vector $\tilde{x}$ in the test set, and store the resulting vectors $\tilde{y}$ in the matrix $\tilde{Y}$.
5: The index i of the largest entry of $\tilde{y}$ determines the detected label, i.e., set $\tilde{y} = e_i$.

We could also replace the pseudoinverse G^+ by the regularized inverse $(G + \varepsilon\mathrm{Id})^{-1}$, where ε is the regularization parameter, which would lead to a slightly different system of linear equations. However, for the numerical experiments in Section 4, we do not use regularization. Algorithm 1 is

equivalent to kernel ridge regression (see, e.g., [15]) with a tensor product kernel. This is not surprising, as we are solving simple least-squares problems.

Remark 2. *Note that the kernel does not necessarily have to be based on tensor products of basis functions for this method to work, we could also simply use, e.g., a Gaussian kernel, which for the MNIST data set leads to slightly lower but similar classification rates. Tensor-based kernels, however, have an exponentially large yet explicit feature space representation and additional structure that could be exploited to speed up computations. Moreover, the kernel-based algorithm outlined above can in the same way be applied to time-series data to learn governing equations in potentially infinite-dimensional feature spaces.*

Compared to the method proposed in [5,6], the advantage of our approach, which can be regarded as a kernel-based formulation of MANDy (or SINDy), is that we can compute a closed-form solution without necessitating any iterations or sweeps. However, even though this approach for classification problems computes an optimal solution of the minimization problem (12), the runtime as well as the memory consumption of the algorithm depend crucially on the size of the training data set (and also the number of labels), and the resulting coefficient tensor $\boldsymbol{\Xi}$ has no guaranteed low-rank structure. We will now propose an alternating optimization method which circumvents this problem.

3.3. Alternating Ridge Regression

In what follows, we will use the TT representation illustrated in Figure 3b for the transformed data tensor $\boldsymbol{\Psi}_X \in \mathbb{R}^{n_1 \times \cdots \times n_p \times m}$. Even though we do not consider a TT operator, the proposed approach is closely related to the DMRG method [4], also called alternating linear scheme (ALS) [28]. As in [5,6], the idea here is to compute a low-rank TT approximation of the coefficient tensor $\boldsymbol{\Xi}$ by an alternating scheme. That is, a low-dimensional system of linear equations has to be solved for each TT core. Our approach is outlined in Algorithm 2.

First, note that instead of solving the minimization problem (12), we can also find separate solutions of

$$\min_{\boldsymbol{\Xi}_i} \left\| Y_{i,:} - \boldsymbol{\Xi}_i^\top \boldsymbol{\Psi}_X \right\|_2 \tag{30}$$

for each row of Y. As these systems can be solved independently, Algorithm 2 can be easily parallelized. We then use a DMRG/ALS-inspired scheme to split the optimization problem (30) into p subproblems. The micromatrix M_μ of such a subproblem can be built from three different parts, namely, $\widehat{\boldsymbol{\Psi}}_{X,\mu}^{(\mu)}$, P_μ, and Q_μ. The latter are both collected in a left and right stack to avoid repetitive computations. Note that P_μ is determined by contracting $P_{\mu-1}$ with the $(\mu-1)$th cores of $\boldsymbol{\Xi}_i$ and $\widehat{\boldsymbol{\Psi}}_X$. Analogously, Q_μ is build from $Q_{\mu+1}$ and the $(\mu+1)$th cores of $\boldsymbol{\Xi}_i$ and $\widehat{\boldsymbol{\Psi}}_X$. During the first half sweep of Algorithm 2, we only have to compute the matrices P_μ, as the used matrices Q_μ are not based on any updated cores. Afterwards, the matrices Q_μ are (re-)computed during the second half. See [28] for further details and Figure 4 for a graphical illustration of the construction of the subproblems and the extraction of the optimized core. Note that it is not necessary to store the (sparse) core $\widehat{\boldsymbol{\Psi}}_{X,\mu}^{(\mu)}$ in its full representation as a 4-dimensional array to construct the matrix M_μ. By using, e.g., NumPy's `einsum` the TT core can be replaced a (dense) matrix containing the corresponding function evaluations.

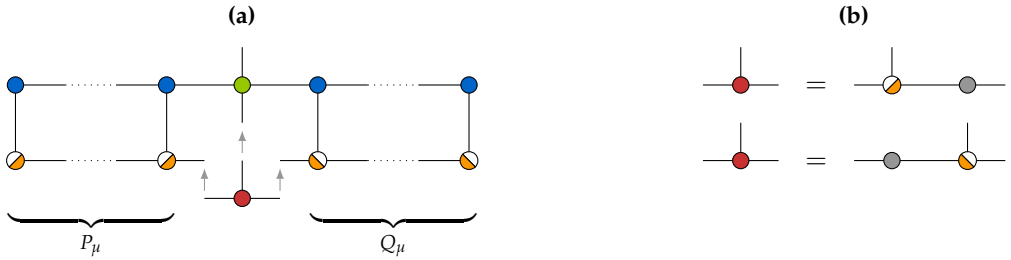

Figure 4. Construction and solution of the subproblem for the μth core: (**a**) The 4-dimensional core of $\widehat{\mathbf{\Upsilon}}_{X,\mu}$ (green circle) is contracted with the matrices P_μ and Q_μ constructed by joining the fixed cores of the coefficient tensor (orange circles) with the corresponding cores of the transformed data tensor. The matricization then defines the matrix M_μ. (**b**) The TT core (red circle) obtained by solving the low-dimensional minimization problem is decomposed (e.g., using QR factorization) into a orthonormal tensor and a triangular matrix. The orthonormal tensor then yields the updated core.

Algorithm 2 Alternating ridge regression (ARR) for classification.

Input: Training set X and label matrix Y, test set $\tilde{X}$, basis functions, initial guesses.

Output: Label matrix $\tilde{Y}$.

1: **for** $i = 1, \ldots, d'$ **do** *(parallelizable)*
2: Define $w = Y_{i,:} = [Y_i^{(1)}, \ldots, Y_i^{(m)}]$.
3: Define initial guess $\mathbf{\Xi}_i$ and right-orthonormalize.
4: Compute right stack $Q_p, \ldots, Q_1$.
5: **for** $\mu = 1, \ldots, p-1$ **do** *(first half sweep)*
6: Compute P_μ.
7: Construct micromatrix M_μ from $P_\mu, \widehat{\mathbf{\Upsilon}}^{(\mu)}_{X,\mu}, Q_\mu$.
8: Determine truncated SVD solution of $\min_v \left\| w - vM_\mu \right\|_2$.
9: Apply QR decomposition to extract updated core.
10: **for** $\mu = p, \ldots, 1$ **do** *(second half sweep)*
11: Compute Q_μ.
12: Construct micromatrix M_μ from $P_\mu, \widehat{\mathbf{\Upsilon}}^{(\mu)}_{X,\mu}, Q_\mu$.
13: Determine truncated SVD solution of $\min_v \left\| w - vM_\mu \right\|_2$.
14: **if** $\mu > 1$ **then**
15: Apply QR decomposition to extract updated core.
16: **else**
17: Set the updated core to a reshape of v.
18: Repeat lines 5–17 to increase accuracy (if needed).
19: Define $\mathbf{\Xi}$ using (31) and set $y = f(x)$ using (13).
20: The index of the largest entry of y determines the detected label, see Algorithm 1.

By orthonormalizing the fixed cores of Ξ, and using truncated SVDs [17] for solving the subsystems, we can interpret our approach as a core-wise ridge regression approximating the solution obtained by kernel-based MANDy, see Appendix B. After approximating the coefficient tensor

$$\Xi = \sum_{i=1}^{d'} \Xi_i \otimes e_i, \tag{31}$$

the decision function f is given by (13). The main difference between our approach and the method introduced in [5,6] is that we do not update the TT cores of Ξ using gradient descent steps. Instead we solve a low-dimensional system of linear equations corresponding to the entire training data set whose solution yields the updated core. Moreover, we solve a minimization problem for each row of the label matrix Y. Using the modified basis decomposition introduced in Section 3.1, it is possible to significantly reduce the storage consumption of the stack, see Algorithm 2 Lines 4 and 11. If we only use a fixed representation for Ψ_X, as given in (17), the additional mode would lead to a much higher storage consumption of the right stack. Thus, our method provides an efficient construction of the subproblems.

4. Numerical Results

We apply the tensor-based classification algorithms described in Sections 3.2 and 3.3 to both the MNIST and fashion MNIST data sets, choosing the basis defined in (14) and setting $\alpha \approx 0.59$. This value was determined empirically for the MNIST data set, but also leads to better classification rates for the fashion MNIST set. Kernel-based MANDy as well as ARR are available in Scikit-TT (https://github.com/PGelss/scikit_tt). The numerical experiments were performed on a Linux machine with 128 GB RAM and an Intel Xeon processor with a clock speed of 3 GHz and eight cores.

For the first approach, using kernel-based MANDy, we do not apply any regularization techniques. For the ARR approach, we set the TT rank for each solution Ξ_i, see Algorithms 2–10, and repeat the scheme five times. Here, we use regularization, i.e., truncated SVDs with a relative threshold of 10^{-2} are applied to the minimization problems given in Algorithm 2 (Lines 8 and 13). The obtained classification rates for the reduced and full MNIST and fashion MNIST data are shown in Figure 5.

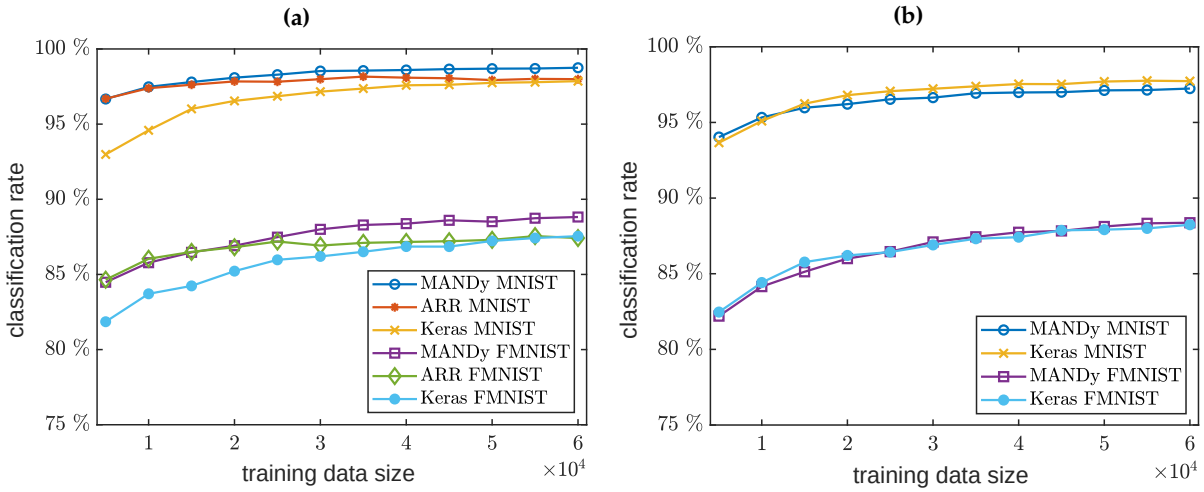

Figure 5. Results for MNIST and fashion MNIST: (**a**) Classification rates for the reduced 14 × 14 images. (**b**) Classification rates for full 28x28 images. Reducing the image size by averaging over groups of pixels improves the performance of the algorithm.

Similarly to [5,6], we first apply the classifiers to the reduced data sets, see Figure 5a. Using MANDy, we obtain classification rates of up to 98.75% for the MNIST and 88.82% for the fashion MNIST data set. Using the ARR approach, the classification rates are not monotonically increasing, which may simply be an effect of the alternating optimization scheme. The highest classification rates we obtain are 98.16% for the MNIST data and 87.55% for the fashion MNIST data. We typically obtain a 100% classification rate for the training data (as a consequence of the richness of the feature space). This is not necessarily a desired property as the learned model might not generalize well to new data, but seems to have no detrimental effects for the simple MNIST classification problem. As shown in Figure 5b, kernel-based MANDy can still be applied when considering the full data sets without reducing the image size. Here, we obtain classification rates of up to 97.24% for the MNIST and 88.37% for the fashion MNIST data set. That we obtain lower classification rates for the full images as compared to the reduced ones might be due to the fact that pixel-by-pixel comparisons of images are not expedient. The averaging effect caused by downscaling the images helps to detect coarser features. This is similar to the effect of convolutional kernels and pooling layers. In principle, ARR can also be used for the classification of the full data sets. So far, however, our numerical experiments produced only classification rates significantly lower than those obtained by applying MANDy (95.94% for the MNIST and 82.18% for fashion MNIST data set). This might be due to convergence issues caused by the kernel. The application to higher-order transformed data tensors and potential improvements of ARR will be part of our future research.

Figure 5 also shows a comparison with tensorflow. We run the code provided as a classification tutorial (www.tensorflow.org/tutorials/keras/basic_classification) ten times and compute the average classification rate. The input layer of the network comprises 784 nodes (one for each pixel; for the reduced data sets, we thus have only 196 input nodes), followed by two dense layers with 128 and 10 nodes. The layer with 10 nodes is the output layer containing probabilities that a given image belongs to the class represented by the respective neuron. Note that although more sophisticated methods and architectures for these problems exists—see the (fashion) MNIST website for a ranking—the results show that our tensor-based approaches are competitive with state-of-the-art deep-learning techniques.

To understand the numerical results for the MNIST data set (obtained by applying kernel-based MANDy to all 60,000 training images), we analyze the misclassified images, examples of which are displayed in Figure 6a. For misclassified images x, the entries of $f(x)$, see (29), are often numerically zero, which implies that there is no other image in the training set that is similar enough so that the kernel can pick up the resemblance. Some of the remaining misclassified digits are hard to recognize even for humans. Histograms demonstrating which categories are misclassified most often are shown in Figure 6b. Here, we simply count the instances where an image with label i was assigned the wrong label j. The digits 2 and 7, as well as 4 and 9, are confused most frequently. Additionally, we wish to visualize what the algorithm detects in the images. To this end, we perform a sensitivity analysis as follows. Starting with an image whose pixel values are constant everywhere (zero or any other value smaller than one, we choose 0.5), we set pixel (i, j) to one and compute $y = f(x)$ for this image. The process is repeated for all pixels. For each label, we then plot a heat map of the values of y. This tells us which pixels contribute most to the classification of the images. The resulting maps are shown in Figure 6c. Except for the digit 1, the results are highly similar to the images obtained by averaging over all images containing a certain digit.

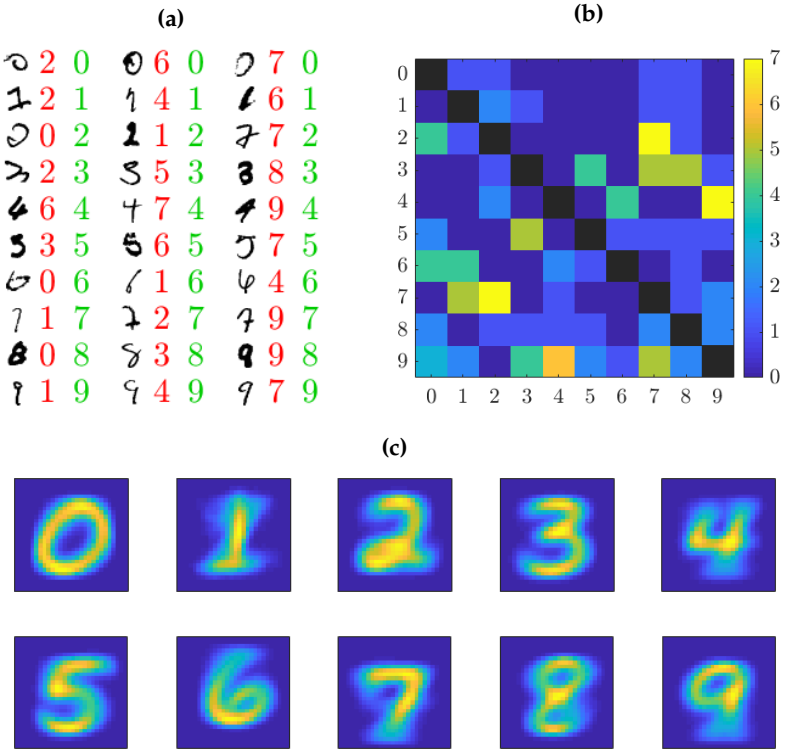

Figure 6. MNIST classification: (**a**) Images misclassified by kernel-based MANDy described in Section 3.2. The original image is shown in black, the identified label in red, and the correct label in green. (**b**) Histograms illustrating which categories are misclassified most often. The rows represent the correct labels of the misclassified image and the columns the detected labels. (**c**) Visualizations of the learned classifiers showing a heat map of the classification function obtained by applying it to images that differ in one pixel.

Figure 7 shows examples of misclassified images and the corresponding histogram as well as the results of the sensitivity analysis for the fashion MNIST data set. We see that the images of shirts (6) are most difficult to classify (due to the ambiguity in the category definitions), whereas trousers (1) and bags (8) have the lowest misclassification rates (probably due to their distinctive shapes). In contrast to the MNIST data set, the results of the sensitivity analysis differ widely from the average images. The classifier for coats (4), for instance, "looks for" a zipper and coat pockets, which are not visible in the "average coat", and the classifier for dresses (3) seems to base the decision on the presence of creases, which are also not distinguishable in the "average dress". The interpretation of other classifiers is less clear, e.g., the ones for sandals (5) and sneakers (7) seem to be contaminated by other classes.

Comparing the runtimes of both approaches applied to the reduced data sets with 60,000 training images, kernel-based MANDy needs approximately one hour for the construction of the decision function (29). On the other hand, ARR needs less than 10 minutes to compute the coefficient tensor assuming we parallelize Algorithm 2.

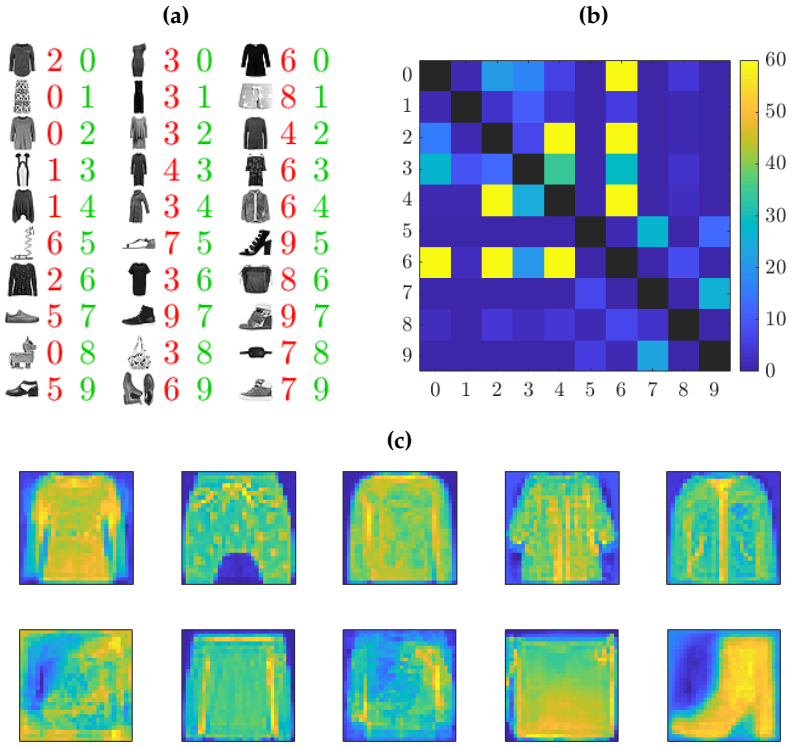

Figure 7. Fashion MNIST classification: (**a**) Misclassified images. (**b**) Histogram of misclassified images. (**c**) Visualizations of the learned classifiers.

5. Conclusions

In this work, we presented two different tensor-based approaches for supervised learning. We showed that a kernel-based extension of MANDy can be utilized for image classification. That is, extending the method to arbitrary least-squares problems (originally, MANDy was developed to learn governing equations of dynamical systems) and using sequences of Hadamard products for the computation of the pseudoinverse, we were able to demonstrate the potential of kernel-based MANDy by applying it to the MNIST and fashion MNIST data sets. Additionally, we proposed the alternating optimization scheme ARR, which approximates the coefficient tensors by low-rank TT decompositions. Here, we used a mutable tensor representation of the transformed data tensors in order to construct low-dimensional regression problems for optimizing the TT cores of the coefficient tensor.

Both approaches use an exponentially large set of basis functions in combination with least-squares regression techniques on a given set of training images. The results are encouraging and show that methods exploiting tensor products of simple basis functions are able to detect characteristic features in image data. The work presented in this paper constitutes a further step towards tensor-based techniques for machine learning.

The reason why we can handle the extremely high-dimensional feature space spanned by basis functions is its tensor product format. Besides, the general questions of the choice of basis functions and the expressivity of these functions, the rank-one tensor products that were used in this work can, in principle, be replaced by other structures, which might result in higher classification rates. For instance, the transformation of an image could be given by a TT representation with higher ranks or hierarchical tensor decompositions (with the aim to detect features on different levels of abstraction).

Furthermore, we could define different basis functions for each pixel, vary the number of basis functions per pixel, or define basis functions for groups of pixels.

Even though kernel-based MANDy computes the minimum norm solution of the considered regression problems as an exact TT decomposition, the method is likely to suffer from high ranks of the transformed data tensors and might thus not be competitive for large data sets. At the moment, we are computing the Gram matrix for the entire training data set. However, a possibility to speed up computations and to lower the memory consumption is found in exploiting the properties of the kernel. That is, if the kernel almost vanishes if two images differ significantly in at least one pixel (as it is the case for the specific kernel used in this work, provided that the originally proposed value $\alpha = \frac{\pi}{2}$ is used), the Gram matrix is essentially sparse when setting entries smaller than a given threshold to zero. Using sparse solvers would allow us to handle much larger data sets. Moreover, the construction of the Gram matrix is highly parallelizable and it would be possible to use GPUs to assemble it in a more efficient fashion.

Further modifications of ARR such as different regression methods for the subproblems, an optimized ordering of the TT cores, and specific initial coefficient tensors can help to improve the results. We provided an explanation for the stability of ARR, but the properties of alternating regression schemes have to be analyzed in more detail in the future.

Author Contributions: Conceptualization, S.K. and P.G.; methodology, S.K. and P.G.; software, S.K. and P.G.; writing, S.K. and P.G.

Funding: This research has been funded by Deutsche Forschungsgemeinschaft (DFG) through grant CRC 1114 "Scaling Cascades in Complex Systems". Part of this research was performed while S.K. was visiting the Institute for Pure and Applied Mathematics (IPAM), which is supported by the National Science Foundation (Grant No. DMS-1440415).

Acknowledgments: We would like to thank Michael Götte and Alex Goeßmann from the TU Berlin for interesting discussions related to tensor decompositions and system identification. The publication of this article was funded by Freie Universität Berlin.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Representation of Transformed Data Tensors

Proposition A1. *For all $i \in \{1, \dots, p\}$, it holds that*

$$\widehat{\mathbf{\Psi}}_{X,i} \Big|_{n_1,\dots,n_d}^{m} = \mathbf{\Psi}_X \Big|_{n_1,\dots,n_d}^{m}.$$

That is, the TT decompositions $\widehat{\mathbf{\Psi}}_{X,i}$ and $\mathbf{\Psi}_X$ represent the same tensor in $\mathbb{R}^{n_1 \times \cdots \times n_p \times m}$.

Proof. An entry of $\widehat{\mathbf{\Psi}}_{X,\mu}$, $1 < \mu < p$, is given by

$$\left(\widehat{\mathbf{\Psi}}_{X,\mu}\right)_{i_1,\dots,i_p,j} = \sum_{k_1=1}^{m} \cdots \sum_{k_{p-1}=1}^{m} \left(\widehat{\mathbf{\Psi}}_{X,\mu}^{(1)}\right)_{1,i_1,k_1} \cdot \ldots \cdot \left(\widehat{\mathbf{\Psi}}_{X,\mu}^{(\mu)}\right)_{k_{\mu-1},i_\mu,j,k_\mu} \cdot \ldots \cdot \left(\widehat{\mathbf{\Psi}}_{X,\mu}^{(p)}\right)_{k_{p-1},i_p,1}.$$

By definition,

$$\left(\widehat{\mathbf{\Psi}}_{X,\mu}^{(\mu)}\right)_{k_{\mu-1},i_\mu,j,k_\mu} \neq 0 \quad \Leftrightarrow \quad k_{\mu-1} = j = k_\mu.$$

On the other hand, an entry of $\widehat{\mathbf{\Psi}}^{(\nu)}_{X,\mu}$ with $\nu \neq \mu$ and $1 < \nu < p$ is nonzero if and only if $k_{\nu-1} = k_\nu$. It follows that

$$\begin{aligned}\left(\widehat{\mathbf{\Psi}}_{X,\mu}\right)_{i_1,\ldots,i_p,j} &= \left(\widehat{\mathbf{\Psi}}^{(1)}_{X,\mu}\right)_{1,i_1,j} \cdot \ldots \cdot \left(\widehat{\mathbf{\Psi}}^{(\mu)}_{X,\mu}\right)_{j,i_\mu,j,j} \cdot \ldots \cdot \left(\widehat{\mathbf{\Psi}}^{(p)}_{X,\mu}\right)_{j,i_p,1} \\ &= \psi_{1,i_1}(x_j) \cdot \ldots \cdot \psi_{\mu,i_\mu}(x_j) \cdot \ldots \cdot \psi_{p,i_p}(x_j),\end{aligned}$$

This can be shown in an analogous fashion for $\mu = 1$ and $\mu = p$. □

Appendix B. Interpretation of ARR as ALS Ridge Regression

The following reasoning will elucidate the relation between ARR, ridge regression, and kernel-based MANDy. We only outline the rough idea without concrete proofs. Let R_μ denote the retraction operator, see [28], consisting of the fixed TT cores $\Xi^{(1)}, \ldots, \Xi^{(\mu-1)}$ and $\Xi^{(\mu+1)}, \ldots, \Xi^{(p)}$ of the solution Ξ at any iteration step of Algorithm 2. Furthermore, assume that $\Xi^{(1)}, \ldots, \Xi^{(\mu-1)}$ are left- and $\Xi^{(\mu+1)}, \ldots, \Xi^{(p)}$ right-orthonormal. In Lines 8 and 13 of Algorithm 2, we consider the system (with a slight abuse of notation)

$$y = M_\mu x = \left(\mathbf{\Psi}_X^\top \cdot R_\mu\right) x.$$

The application of a truncated SVD to the matricization of $\mathbf{\Psi}_X^\top \cdot R_\mu$ (as done in Algorithm 2) is then similar to a regularization in the form of

$$\min_x \left\{ \|y - M_\mu x\|_2^2 + \varepsilon \|x\|_2^2 \right\} \tag{A1}$$

with appropriate regularization parameter ε, i.e., $x \approx M_\mu^+ y$ for both approaches, see [17,30]. The formulation (A1) is known as Tikhonov's smoothing functional, ridge regression, or ℓ^2 regularization (which, of course, could also directly be applied in Algorithm 2). The solution of (A1) is also the solution of the regularized normal equation

$$M_\mu^\top y = \left(M_\mu^\top M_\mu + \varepsilon \mathrm{Id}\right) x,$$

see, e.g., [31]. As $R_\mu^\top R_\mu = \mathrm{Id}$, it follows that

$$\left(R_\mu^\top \mathbf{\Psi}_X\right) y = \left(R_\mu^\top \left(\mathbf{\Psi}_X \mathbf{\Psi}_X^\top + \varepsilon \mathrm{Id}\right) R_\mu\right) x.$$

In fact, this is a subproblem corresponding to the application of ALS [28] to the tensor-based system

$$\mathbf{\Psi}_X y = \left(\mathbf{\Psi}_X \mathbf{\Psi}_X^\top + \varepsilon \mathrm{Id}\right) \Xi. \tag{A2}$$

Note that all requirements for the application of ALS are satisfied since $\mathbf{\Psi}_X \mathbf{\Psi}_X^\top + \varepsilon \mathrm{Id}$ is a symmetric positive definite tensor operator and R_μ is orthonormal. The system of linear equations given in (A2) is then equivalent to the minimization problem

$$\min_\Xi \left\{ \left\| y - \mathbf{\Psi}_X^T \Xi \right\|_2^2 + \varepsilon \|\Xi\|_2^2 \right\}.$$

For sufficiently small ε, it holds that $\Xi \approx \mathbf{\Psi}_X^+ y$, see [32], meaning Algorithm 2 computes an approximation of the coefficient tensor resulting from the application of kernel-based MANDy, see Section 3.2.

References

1. Beylkin, G.; Garcke, J.; Mohlenkamp, M.J. Multivariate Regression and Machine Learning with Sums of Separable Functions. *SIAM J. Sci. Comput.* **2009**, *31*, 1840–1857. doi:10.1137/070710524.
2. Novikov, A.; Podoprikhin, D.; Osokin, A.; Vetrov, D. Tensorizing Neural Networks. In *Advances in Neural Information Processing Systems 28 (NIPS)*; Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA; 2015; pp. 442–450.
3. Cohen, N.; Sharir, O.; Shashua, A. On the expressive power of deep learning: A tensor analysis. In *29th Annual Conference on Learning Theory*; Feldman, V., Rakhlin, A.; Shamir, O., Eds.; Proceedings of Machine Learning Research; Columbia University: New York, NY, USA; 2016; Volume 49, pp. 698–728.
4. White, S.R. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.* **1992**, *69*, 2863–2866. doi:10.1103/Physrevlett.69.2863.
5. Stoudenmire, E.M.; Schwab, D.J. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*; Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA; 2016; pp. 4799–4807.
6. Stoudenmire, E.M.; Schwab, D.J. Supervised learning with quantum-inspired tensor networks. *arXiv* **2016**, arXiv:1605.05775.
7. Stoudenmire, E.M. Learning relevant features of data with multi-scale tensor networks. *Quantum Sci. Technol.* **2018**, *3*, 034003. doi:10.1088/2058-9565/Aaba1a.
8. Huggins, W.; Patil, P.; Mitchell, B.; Whaley, K.B.; Stoudenmire, E.M. Towards quantum machine learning with tensor networks. *Quantum Sci. Technol.* **2019**, *4*, 024001. doi:10.1088/2058-9565/Aaea94.
9. Roberts, C.; Milsted, A.; Ganahl, M.; Zalcman, A.; Fontaine, B.; Zou, Y.; Hidary, J.; Vidal, G.; Leichenauer, S. TensorNetwork: A library for physics and machine learning. *arXiv* **2019**, arXiv:1905.01330.
10. Efthymiou, S.; Hidary, J.; Leichenauer, S. TensorNetwork for Machine Learning. *arXiv* **2019**, arXiv:1906.06329.
11. Brunton, S.L.; Proctor, J.L.; Kutz, J.N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci. USA* **2016**, *113*, 3932–3937. doi:10.1073/Pnas.1517384113.
12. Rudy, S.H.; Brunton, S.L.; Proctor, J.L.; Kutz, J.N. Data-driven discovery of partial differential equations. *Sci. Adv.* **2017**, *3*. doi:10.1126/Sciadv.1602614.
13. Gelß, P.; Klus, S.; Eisert, J.; Schütte, C. Multidimensional Approximation of Nonlinear Dynamical Systems. *J. Comput. Nonlinear Dyn.* **2019**, *14*, 061006. doi:10.1115/1.4043148.
14. Schuld, M.; Killoran, N. Quantum machine learning in feature Hilbert spaces. *Phys. Rev. Lett.* **2019**, *122*, 040504. doi:10.1103/Physrevlett.122.040504.
15. Shawe-Taylor, J.; Cristianini, N. *Kernel Methods for Pattern Analysis*; Cambridge University Press: Cambridge, UK, 2004. doi:10.1017/CBO9780511809682.
16. Nüske, F.; Gelß, P.; Klus, S.; Clementi, C. Tensor-based EDMD for the Koopman analysis of high-dimensional systems. *arXiv* **2019**, arXiv:1908.04741.
17. Hansen, P.C. The truncated SVD as a method for regularization. *BIT Numer. Math.* **1987**, *27*, 534–553. doi:10.1007/Bf01937276.
18. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. doi:10.1109/5.726791.
19. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
20. Golub, G.H.; van Loan, C.F. *Matrix Computations*, 4th ed.; The Johns Hopkins University Press: Baltimore, MD, USA, 2013.
21. Oseledets, I.V. A New Tensor Decomposition. *Dokl. Math.* **2009**, *80*, 495–496. doi:10.1134/S1064562409040115.
22. Oseledets, I.V. Tensor-Train Decomposition. *SIAM J. Sci. Comput.* **2011**, *33*, 2295–2317. doi:10.1137/090752286.
23. Penrose, R. Applications of negative dimensional tensors. In *Combinatorial Mathematics and Its Applications*; Welsh, D.J.A., Ed.; Academic Press Inc.: Cambridge, MA, USA, 1971; pp. 221–244.

24. Oseledets, I.V.; Tyrtyshnikov, E.E. Breaking the Curse of Dimensionality, Or How to Use SVD in Many Dimensions. *SIAM J. Sci. Comput.* **2009**, *31*, 3744–3759. doi:10.1137/090748330.
25. Klus, S.; Gelß, P.; Peitz, S.; Schütte, C. Tensor-based dynamic mode decomposition. *Nonlinearity* **2018**, *31*. doi:10.1088/1361-6544/Aabc8f.
26. Gelß, P.; Matera, S.; Schütte, C. Solving the Master Equation Without Kinetic Monte Carlo. *J. Comput. Phys.* **2016**, *314*, 489–502. doi:10.1016/J.Jcp.2016.03.025.
27. Gelß, P.; Klus, S.; Matera, S.; Schütte, C. Nearest-neighbor interaction systems in the tensor-train format. *J. Comput. Phys.* **2017**, *341*, 140–162. doi:10.1016/J.Jcp.2017.04.007.
28. Holtz, S.; Rohwedder, T.; Schneider, R. The Alternating Linear Scheme for Tensor Optimization in the Tensor Train Format. *SIAM J. Sci. Comput.* **2012**, *34*, A683–A713. doi:10.1137/100818893.
29. Liu, Y.; Zhang, X.; Lewenstein, M.; Ran, S. Entanglement-guided architectures of machine learning by quantum tensor network. *arXiv* **2018**, arXiv:1803.09111.
30. Groetsch, C.W. *Inverse Problems in the Mathematical Sciences*; Vieweg+Teubner Verlag: Wiesbaden, Germany, 1993. doi:10.1007/978-3-322-99202-4.
31. Zhdanov, A.I. The method of augmented regularized normal equations. *Comput. Math. Math. Phys.* **2012**, *52*, 194–197. doi:10.1134/S0965542512020169.
32. Barata, J.C.A.; Hussein, M.S. The Moore–Penrose pseudoinverse: A tutorial review of the theory. *Braz. J. Phys.* **2012**, *42*, 146–165. doi:10.1007/S13538-011-0052-Z.