Tensor-Train Discriminant Analysis

Seyyid Emre Sofuoglu, Selin Aviyente
Electrical and Computer Engineering, Michigan State University, East Lansing, MI, 48824, USA.
Email: sofuoglu@msu.edu, aviyente@egr.msu.edu,

***Abstract*—The rapid development of information technology is making it possible to collect massive amounts of multidimensional, multimodal data with high dimensionality in a diverse set of science and engineering disciplines. Although there has been a lot of recent work in the area of unsupervised tensor learning, extensions to supervised learning, feature extraction and classification are still limited. Moreover, most of the existing supervised tensor learning approaches are based on the Tucker model. However, this model has some limitations for large tensors including high memory and execution time costs. In this paper, we introduce a supervised learning approach for tensor classification based on the tensor-train model. In particular, we introduce two computationally efficient implementations of tensor-train discriminant analysis (TT-DA). The proposed approaches are evaluated on image classification tasks with respect to computation time, storage cost and classification accuracy.**

***Index terms*—** Tensor-Train, Tensor Networks, Multidimensional Discriminant Analysis, Supervised Tensor-Train.

I. Introduction

Statistical learning, pattern recognition and data mining with high dimensional data pose new challenges. Most of the state-of-the-art supervised learning algorithms assume that data instances or training samples are represented as vectors. However, in many real-world applications such as computer vision, data instances are more naturally represented as second-order or higher-order tensors, where the order of a tensor corresponds to the number of modes. Conventional supervised learning approaches applied to vectorized tensor samples are inadequate when dealing with massive multidimensional data as they cannot capture the cross-couplings across the different modes and suffer from increasing storage and computational costs [1], [2], [3]. Therefore, there is a growing need for supervised learning methods that can learn discriminant subspaces from tensor data while preserving its inherent multi-modal structure.

In recent years, supervised and unsupervised tensor subspace learning approaches based on the Tucker model have been proposed [4], [5], [6], [7], [8]. Some of these approaches such as Multilinear Principal Component Analysis (MPCA) [8] is successful at dimensionality reduction but not necessarily suitable for discriminative feature extraction. Others such as Multilinear Discriminant Analysis (MDA) [4] are not practical with increasing number of modes, due to exponential increase in the storage cost of Tucker model [9].

Tensor-Train (TT) model, on the other hand, provides better compression than Tucker models as it expresses a given high-dimensional tensor as the product of many low rank, 3-mode tensors [1]. TT model has been extended for various tasks such as PCA [10], manifold learning [11] and neural networks [12]. Although TT has been extended to various unsupervised learning tasks, it has not been fully explored in supervised learning settings. In this paper, we propose a TT based discriminant subspace learning framework to take advantage of the storage efficiency of this model. We also propose a method to reduce the complexity of finding a TT subspace by taking advantage of the flexibility of the TT structure.

This paper differs from the current work in two key ways. First, LDA is generalized to tensor data for the first time using the TT model. Thus, the efficiency of TT in a discriminant subspace learning setting is explored. Second, we improve the computational efficiency of the current TT subspace learning methods by introducing two algorithms that take advantage of the flexible TT structure.

The rest of the paper is organized as follows. In Section II, we provide background on TT decomposition and LDA. In Section III, we generalize TT for discriminant analysis and propose two different approaches to solve the corresponding optimization problem. In Section IV, we compare the proposed methods with state-of-the-art tensor subspace learning methods for classification applications.

II. Background

Let $\mathcal{Y}_c^k \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be a sample tensor where $c \in \{1, \ldots, C\}$ is the class index and $k \in \{1, \ldots, K\}$ is the sample index, from a given training data with C classes where each class has K samples.

A. Tensor-Train Notation

Definition 1. (Vectorization, Matricization and Reshaping) $\mathbf{V}(.)$ is a vectorization operator such that $\mathbf{V}(\mathcal{Y}_c^k) \in \mathbb{R}^{I_1 I_2 \ldots I_N \times 1}$. $\mathbf{T}_n(.)$ is a tensor-to-matrix reshaping operator defined as $\mathbf{T}_n(\mathcal{Y}_c^k) \in \mathbb{R}^{I_1 \ldots I_n \times I_{n+1} \ldots I_N}$ and the inverse operator is denoted as $\mathbf{T}_n^{-1}(.)$. $\mathbf{M}_k(.)$ is a matrix reshaping operator defined as $\mathbf{M}_k(D) \in \mathbb{R}^{n_1 n_2 \ldots n_k \times n_{k+1} \ldots n_K}$ where $D \in \mathbb{R}^{n_1 \ldots n_d \times n_{d+1} \ldots n_K}$ and $d \neq k$.

Definition 2. (Left and right unfolding) The left unfolding operator creates a matrix from a tensor by taking all modes except the last mode as row indices and the last mode as column indices, i.e. $\mathbf{L}(\mathcal{Y}_c^k) \in \mathbb{R}^{I_1 I_2 \ldots I_{N-1} \times I_N}$ which is equivalent to $\mathbf{T}_{N-1}(\mathcal{Y}_c^k)$. Right unfolding transforms a tensor to a matrix by taking all the first mode fibers as column vectors, i.e. $\mathbf{R}(\mathcal{Y}_c^k) \in \mathbb{R}^{I_1 \times I_2 I_3 \ldots I_N}$ which is equivalent to $\mathbf{T}_1(\mathcal{Y}_c^k)$. The inverse of these operators are denoted as $\mathbf{L}^{-1}(.)$ and $\mathbf{R}^{-1}(.)$, respectively.

Definition 3. (Tensor Merging Product) Tensor merging product connects two tensors along some given sets of modes. For two tensors $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{B} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ where $I_n = J_m$ and $I_{n+1} = J_{m-1}$ for some n and m, tensor merging product is shown as [9]:

$$\mathcal{C} = \mathcal{A} \times_{n,n+1}^{m,m-1} \mathcal{B}. \tag{1}$$

$\mathcal{C} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+2} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-2} \times J_{m+1} \times \cdots \times J_M}$ is calculated as:

$$\mathcal{C}(i_1,\ldots,i_{n-1},i_{n+2},\ldots,i_N,j_1,\ldots,j_{m-2},j_{m+1},\ldots,j_M) = \sum_{t_1=1}^{I_n}\sum_{t_2=1}^{J_{m-1}}\big[\mathcal{A}(i_1,\ldots,i_{n-1},i_n=t_1,i_{n+1}=t_2,i_{n+1},\ldots,i_N)\ \mathcal{B}(j_1,\ldots,j_{m-2},j_{m-1}=t_2,j_m=t_1,j_{m+1},\ldots,j_M)\big]. \quad (2)$$

B. Tensor-Train Decomposition

Using tensor-train decomposition, each element of $\mathcal{Y}_c^k$ can be represented as:

$$\mathcal{Y}_c^k(i_1,i_2,\ldots,i_N) = \mathcal{U}_1(1,i_1,:)\mathcal{U}_2(:,i_2,:)\ldots\mathcal{U}_N(:,i_N,:)\mathbf{x}_c^k, \quad (3)$$

where $\mathcal{U}_n \in \mathbb{R}^{R_{n-1}\times I_n\times R_n}$, $R_n < I_n$ are the ranks of the corresponding mode $n \in \{1,\ldots,N\}$ and $\mathbf{x}_c^k \in \mathbb{R}^{R_N\times 1}$ are the projected sample vectors. TT decomposition given in (3) can be rewritten as, $\mathcal{Y}_c^k = \mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_N \times_3^1 \mathbf{x}_c^k$ [9].

When $\mathcal{Y}_c^k$ is vectorized, an equivalent matrix projection for (3) is obtained as:

$$\mathbf{V}(\mathcal{Y}_c^k) = \mathbf{L}(\mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_N)\mathbf{x}_c^k. \quad (4)$$

For the sake of simplicity, we define $U = \mathbf{L}(\mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_N)$ where $U \in \mathbb{R}^{I_1 I_2\ldots I_N\times R_N}$. When $\mathbf{L}(\mathcal{U}_n)$s are left orthogonal, U is also left orthogonal [13].

C. LDA

LDA for vectorized tensor data finds an orthogonal projection U that maximizes the discriminability of projections:

$$\underset{U}{\operatorname{argmin}}\, tr(U^\top(S_W - \lambda S_B)U) = \underset{U}{\operatorname{argmin}}\, tr(U^\top S U), \quad (5)$$

where $S = S_W - \lambda S_B$, S_W and S_B are, respectively, within-class and between-class scatter matrices given by:

$$S_W = \sum_{c=1}^{C}\sum_{k=1}^{K}\mathbf{V}(\mathcal{Y}_c^k - \mathcal{M}_c)\mathbf{V}(\mathcal{Y}_c^k - \mathcal{M}_c)^\top,$$
$$S_B = \sum_{c=1}^{C}\sum_{k=1}^{K}\mathbf{V}(\mathcal{M}_c - \mathcal{M})\mathbf{V}(\mathcal{M}_c - \mathcal{M})^\top,$$

where $\mathcal{M}_c = \frac{1}{K}\sum_{k=1}^{K}\mathcal{Y}_c^k$ is the class mean for each class c and $\mathcal{M} = \frac{1}{CK}\sum_{c=1}^{C}\sum_{k=1}^{K}\mathcal{Y}_c^k$ is the total mean of all sample tensors.

Since U is an orthogonal projection matrix, this is equivalent to minimizing within-class scatter and maximizing between class scatter of projections. (5) can be solved by taking the lowest R_N eigenvalues of $S \in \mathbb{R}^{I_1\ldots I_N\times I_1\ldots I_N}$ as U.

When the data are higher order tensors, LDA needs to first vectorize them and then find a solution as shown above. This creates several problems as the intrinsic structure of the data is destroyed and dimensionality, time and storage cost for the subspace increases exponentially. Thus, we propose to solve the above problem by constraining U to be a TT subspace to reduce the computational and storage complexity and to obtain a solution that will preserve the inherent structure.

III. Tensor-Train Discriminant Analysis (TTDA)

The goal of TTDA is to learn projection tensors $\mathcal{U}_n \in \mathbb{R}^{R_{n-1}\times I_n\times R_n}, n \in \{1,\ldots,N\}$ using TT decomposition such that the discriminability of projections $\mathbf{x}_c^k, \forall c,k$ is high and $\mathcal{U}_n$ are left orthogonal, i.e. $\mathbf{L}(\mathcal{U}_n)^\top\mathbf{L}(\mathcal{U}_n) = \mathbb{I}_{R_{n-1}I_n}$ where $\mathbb{I}_s$ is an identity matrix of size $s\times s$. To find optimal $\mathcal{U}_n$s, we equivalently rewrite (5) as:

$$\mathcal{U}_n = \underset{\hat{\mathcal{U}}_n}{\operatorname{argmin}}\, tr\Big[\mathbf{L}(\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \hat{\mathcal{U}}_n \times_3^1 \cdots \times_3^1 \mathcal{U}_N)^\top S\mathbf{L}(\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \hat{\mathcal{U}}_n \times_3^1 \cdots \times_3^1 \mathcal{U}_N)\Big], \quad (6)$$

which is dependent on $\mathcal{U}_k, \forall k \in \{1,\ldots,N\}, k \neq n$. Hence, the solution for each $\mathcal{U}_n$ will be computed iteratively, by fixing $\mathcal{U}_k, \forall k \neq n$ which is sub-optimal.

Let $\mathcal{U}_{n-1}^L = \mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_{n-1}$ and $\mathcal{U}_n^R = \mathcal{U}_{n+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N$. Using tensor merging product instead of unfolding operator and leaving $\hat{\mathcal{U}}_n$ out, it can be shown [11] that the solution to (6) is found by first calculating:

$$\mathcal{A}_n = tr_4^8\Bigg[\mathcal{U}_{n-1}^L \times_{1,\ldots,n-1}^{1,\ldots,n-1}\Big[\mathcal{U}_n^R \times_{1,\ldots,N-n}^{n+1,\ldots,N}\Big(\mathcal{U}_{n-1}^L \times_{1,\ldots,n-1}^{N+1,\ldots,N+n-1}(\mathcal{U}_n^R \times_{1,\ldots,N-n}^{N+n+1,\ldots,2N}\mathcal{S})\Big)\Big]\Bigg], \quad (7)$$

where $\mathcal{S} = \mathbf{T}_N^{-1}(S) \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ and $\mathcal{A}_n \in \mathbb{R}^{R_{n-1}\times I_n\times R_n\times R_{n-1}\times I_n\times R_n}$. tr_i^j denotes the trace operation on matrix slices defined by modes i and j which reduces these modes. Then, the solution to (6) is:

$$\mathcal{U}_n = \underset{\hat{\mathcal{U}}_n}{\operatorname{argmin}}\left(\hat{\mathcal{U}}_n \times_{1,2,3}^{1,2,3}\left(\mathcal{A}_n \times_{4,5,6}^{1,2,3}\hat{\mathcal{U}}_n\right)\right). \quad (8)$$

Let $A_n = \mathbf{T}_3(\mathcal{A}_n) \in \mathbb{R}^{R_{n-1}I_nR_n\times R_{n-1}I_nR_n}$, then (8) can be rewritten as:

$$\mathcal{U}_n = \underset{\hat{\mathcal{U}}_n}{\operatorname{argmin}}\,\mathbf{V}(\hat{\mathcal{U}}_n)^\top A_n\mathbf{V}(\hat{\mathcal{U}}_n). \quad (9)$$

This is a non-convex function due to unitary constraints and it can be solved by the algorithm proposed in [14]. The procedure above is computationally expensive due to the complexity of finding each $\mathcal{A}_n$, which is on the order of $\mathcal{O}\left(R_n\prod_{i=1}^{N}I_i^2\right)$.

A. Approximate Solution

In order to reduce the complexity, let $V_{R_N} \in \mathbb{R}^{I_1I_2\ldots I_N\times R_N}$ be the solution to the LDA problem in (5), that is, a matrix with columns corresponding to the R_N eigenvectors of S corresponding to the smallest eigenvalues. Then, the goal is to find a TT subspace $\{\mathcal{U}_1,\ldots,\mathcal{U}_N\}$ that will give the best approximation to V_{R_N}. First, all $\mathcal{U}_n$s are initialized by TT decomposition. Then, the optimization function can be reformulated as:

$$\min_U \|U - V_{R_N}\|_F^2 = \min_{\mathcal{U}_1,\mathcal{U}_2,\ldots,\mathcal{U}_N} \|\mathbf{L}(\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \mathcal{U}_N) - V_{R_N}\|_F^2. \quad (10)$$

To solve this problem for each $\mathcal{U}_n$, (10) can equivalently be written as [11]:

$$\underset{\hat{\mathcal{U}}_n}{\operatorname{argmin}}\left\|\left[\mathbb{I}_{I_n}\otimes\mathbf{L}(\mathcal{U}_{n-1}^L)\right]\mathbf{L}(\hat{\mathcal{U}}_n)\mathbf{R}(\mathcal{U}_n^R) - \mathbf{M}_n(V_{R_N})\right\|_F^2, \quad (11)$$

The above problem is a non-convex problem due to unitary constraints on $\hat{\mathcal{U}}_n$ where the convex part is in the form $\|HQP - G\|_F^2$ where $Q^\top Q = \mathbb{I}$ and can be solved using the algorithm proposed in [14]. A pseudo-code for this solution is given in Algorithm 1.

Algorithm 1 Tensor-Train Discriminant Analysis-Approximated Tensor Network (TTDA-ATN)

Input: Input tensors $\mathcal{Y}_c^k \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ where $c \in \{1, \ldots, C\}$ and $k \in \{1, \ldots, K\}$, λ, $R_1, \ldots, R_N$
Output: $\mathcal{U}_n, n \in \{1, \ldots, N\}$, and $\mathbf{x}_c^k, \quad \forall c, k$

$\mathcal{U}_n \leftarrow \mathbb{I}_{I_n}(:, 1 : R_n)$, $\forall n \in \{1, \ldots, N\}$.
$S \leftarrow S_W - \lambda S_B$.
$V_{R_N} \leftarrow V(:, 1 : R_N)$, where $[V, \Lambda] = \text{eig}(S)$.
for $n = 1 : N$ **do**
  $H \leftarrow \mathbb{I}_{I_n} \otimes \mathbf{L}(\mathcal{U}_{n-1}^L)$.
  $P \leftarrow \mathbf{R}(\mathcal{U}_n^R)$.
  $G \leftarrow \mathbf{M}_n(V_{R_N})$.
  $Q \leftarrow \text{argmin}_Q(\|HQP - G\|_F^2), \quad Q^\top Q = \mathbb{I}$.
  $\mathcal{U}_n \leftarrow \mathbf{L}^{-1}(Q)$
end for
$U = \mathbf{L}(\mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_N)$
$\mathbf{x}_c^k \leftarrow U^\top \mathbf{V}(\mathcal{Y}_c^k), \quad \forall c, k$.

B. Two-Way TTDA (TTDA-TW)

As shown in [11], the procedure to calculate $\mathcal{A}_n$ and solve (6) is computationally expensive. On the other hand, TTDA-ATN requires the solution of LDA, i.e. V_{R_N}, which increases the complexity exponentially as mentioned before. Thus, there is a need to develop a method to reduce the complexity of computing $\mathcal{A}_n$s, taking advantage of TT structure.

With TT, by permuting the sample mode and expressing column and row spaces using two sets of $\mathcal{U}_n$s, the inputs can be projected to a core matrix. Thus, we propose the following procedure to increase the efficiency of the representation:

1) Separate the $\mathcal{U}_n$s into two different sets as left and right projection subspaces. Then, TT can be re-expressed:
$$\mathcal{Y}_c^k = \mathcal{U}_1 \times_3^1 \cdots \times_3^1 \mathcal{U}_m \times_3^1 Z_c^k \times_2^1 \mathcal{U}_{m+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N, \quad (12)$$
where $Z_c^k \in \mathbb{R}^{R_m \times R_{m+1}}$ is the projected sample, $\mathcal{U}_n \in \mathbb{R}^{R_n \times I_n \times R_{n+1}}, n \geq m+1$ and $\mathcal{U}_n \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}, n < m+1$.
2) Solve for the left-set $\mathcal{U}_1, \ldots, \mathcal{U}_m$ by applying TTDA on projections of $\mathcal{Y}_c^k \forall c, k$ on the right-set $\mathcal{U}_{m+1}, \ldots, \mathcal{U}_N$.
3) Solve for the right-set $\mathcal{U}_{m+1}, \ldots, \mathcal{U}_N$ by applying TTDA on projections of $\mathcal{Y}_c^k \forall c, k$ on the left-set $\mathcal{U}_1, \ldots, \mathcal{U}_m$.
4) Stop when a convergence criterion is met or a fixed number of iterations is reached.

To select m, we use a center of mass approach and find the m that minimizes $|\prod_{i=1}^m I_i - \prod_{j=m+1}^N I_j|$. This way, the problem can be separated into two parts which have similar computational complexities.

Let $U_m^L = \mathbf{L}(\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \mathcal{U}_m) = \mathbf{L}(\mathcal{U}_m^L) \in \mathbb{R}^{I_1 \ldots I_m \times R_m}$ and $U_m^R = \mathbf{R}(\mathcal{U}_{m+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N) = \mathbf{R}(\mathcal{U}_m^R) \in \mathbb{R}^{R_{m+1} \times I_{m+1} \ldots I_N}$, then (12) can be equivalently written as:
$$\mathbf{T}_m(\mathcal{Y}_c^k) = (U_m^L Z_c^k) U_m^R. \quad (13)$$
Using the right orthogonality of U_m^R, $\mathbf{T}_m(\mathcal{Y}_c^k) {U_m^R}^\top = U_m^L Z_c^k$. Since this is in the same form as (4) when U_m^R is fixed, we can solve for $\mathcal{U}_1, \ldots, \mathcal{U}_m$ using the same procedure as solving (8) [11]. Now, instead of solving for N different $\mathcal{U}_i$s, we will only solve for $m < N$ different $\mathcal{U}_i$s, which will greatly reduce the computational cost even when this operation is done for both left and right sets.

Once the left-set is obtained, we proceed to solve for the right-set. Using left orthogonality of U_m^L, (12) can also be written as:
$$\begin{aligned} {U_m^L}^\top \mathbf{T}_m(\mathcal{Y}_c^k) &= Z_c^k U_m^R, \\ \mathbf{T}_m(\mathcal{Y}_c^k)^\top U_m^L &= {U_m^R}^\top {Z_c^k}^\top, \end{aligned} \quad (14)$$
where ${U_m^R}^\top = \mathbf{R}\left(\mathcal{U}_{m+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N\right)^\top = \mathbf{L}\left(\mathcal{U}_N \times_1^3 \times \mathcal{U}_{N-1} \times_1^3 \cdots \times_1^3 \mathcal{U}_{m+1}\right)$. Thus, (14) has the same structure as (4) when U_m^L is fixed and can be used to solve for $\mathcal{U}_{m+1}, \ldots, \mathcal{U}_N$. After finding $\mathcal{U}_n, n \in \{1, \ldots, N\}$, the corresponding Z_c^k is found as follows:
$$Z_c^k = {U_m^L}^\top \mathbf{T}_m(\mathcal{Y}_c^k) {U_m^R}^\top. \quad (15)$$

IV. Experiments

In this work, we used COIL-100 and Yale-B Face datasets to test the proposed methods.

COIL-100: The dataset consists of 7,200 RGB images of 100 objects of size 128×128. Each object has 72 images, where each image corresponds to a different pose angle ranging from 0 to 360 degrees with increments of 5 degrees. For our experiments, we used grayscale images of 20 objects and downsampled these to 64×64. Each sample image was then reshaped to create a tensor of size $8 \times 8 \times 8 \times 8$, i.e. $\mathcal{Y}_c^k \in \mathbb{R}^{8 \times 8 \times 8 \times 8}$.

Yale-B: The Yale-B Face database [16, 17] is a collection of face images with varying illumination angles and poses. In this paper, a subset which consists of 20 subjects with 64 different illumination angles under the same pose was used. We downsampled the images to 96×84 and reshaped each image to a tensor of size $12 \times 8 \times 7 \times 12$, i.e. $\mathcal{Y}_c^k \in \mathbb{R}^{12 \times 8 \times 7 \times 12}$.

We compare the proposed methods with TTNPE-ATN[11], DGTDA[4], CMDA[4] and LDA for different experimental settings. For LDA, TTDA-ATN and TTNPE-ATN, $\lambda = 1$ is selected. For TTDA-TW, we experimented with several λ values. For DGTDA, λ is set to the largest eigenvalue of S_B/S_W as proposed in [4]. 1-nearest neighbor classification is used for all algorithms.

First, we compare all methods for varying levels of normalized storage cost which is defined as the logarithm of the total number of elements in subspaces and projections divided by the number of elements in the original tensors. For each dataset, 20 samples from each class were selected randomly as training data and the remaining samples were used for testing. This experiment was repeated 10 times and average results are presented in Figs. 1 and 2. From Figs. 1(a) and 2(a), it is observed that for low storage complexity, TTDA-TW and DGTDA are the most computationally efficient methods. With increasing ranks, the subspace sizes for TT methods increase which increases the complexity for convex optimization and results in higher subspace computation times. In general, LDA, TTDA-ATN and TTNPE have higher complexity as they all require an eigenvalue decomposition of a matrix of

TABLE I
COMPARISON OF CLASSIFICATION ACCURACY OF ALGORITHMS ON COIL-100 DATASET

%	Hold-out ratios r (%)		
Algorithms	6.9	13.8	50
LDA	83.8 ± 1.4	91.8 ± 1.47	98.4 ± 0.45
TTNPE	49.9 ± 3.34	61.6 ± 2.45	81.1 ± 2.76
CMDA	76.3 ± 1.14	85.8 ± 1.22	94.8 ± 0.84
DGTDA	68.2 ± 1.14	79.6 ± 1.61	95.6 ± 0.9
TTDA-ATN	86 ± 1.72	$\mathbf{93.6 \pm 1.1}$	98.7 ± 0.39
TTDA-TW($\lambda = 10$)	$\mathbf{86.9 \pm 2.44}$	90.5 ± 1.13	73.8 ± 1.1
TTDA-TW($\lambda = 10^2$)	84.4 ± 1.75	92.4 ± 1.08	$\mathbf{99.2 \pm 0.28}$

TABLE II
COMPARISON OF CLASSIFICATION ACCURACY OF ALGORITHMS ON YALE-B FACE DATASET

%	Hold-out ratios r (%)		
Algorithms	12.5	25	50
LDA	24.5 ± 1.37	32.5 ± 1.4	37.3 ± 1.77
TTNPE	29.6 ± 1.6	38.3 ± 1.26	43.4 ± 1.63
CMDA	72.4 ± 2.48	82.3 ± 0.965	$\mathbf{90 \pm 1.55}$
DGTDA	11.8 ± 0.827	11.3 ± 0.612	19.3 ± 1.18
TTDA-ATN	23.7 ± 1.43	31.2 ± 1.57	35.9 ± 1.82
TTDA-TW($\lambda = 20$)	$\mathbf{77 \pm 2.3}$	55.1 ± 8.36	30.3 ± 4.19
TTDA-TW($\lambda = 50$)	57.5 ± 9.86	$\mathbf{83.9 \pm 1.88}$	84.3 ± 10.2
TTDA-TW($\lambda = 70$)	40.8 ± 7.87	81.4 ± 2.4	$\mathbf{89.6 \pm 1.4}$

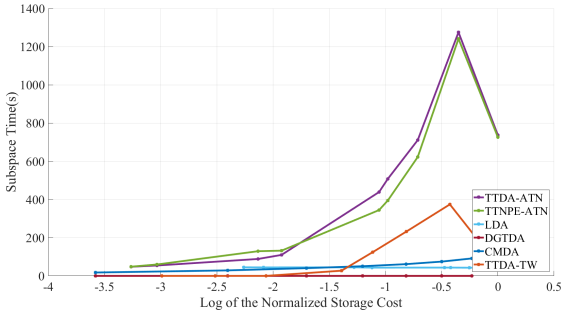
size $I_1 \dots I_N \times I_1 \dots I_N$. It is also observed from Figs. 1(b) and 2(b) that, in general, proposed methods give the highest accuracy with respect to storage cost. Overall, TTDA-TW gives the best classification accuracy results for an appropriately selected value of λ. Other methods give similarly high accuracy at much higher storage costs.

Second, using a fixed storage cost, we experimented with different hold-out ratios r defined as the ratio of the number of training samples to the total number of samples. This experiment was repeated 10 times and average classification accuracy results with standard deviation are presented in Tables I, II. It is observed that the proposed methods have the best results for similar storage complexities. Although for some r TTDA-ATN and CMDA have slightly higher accuracy, TTDA-TW achieves similar accuracy at much lower computational complexity. Also, since λ values have an impact on the performance of TTDA-TW, higher classification accuracy may be obtained by optimizing λ. This can be done easily as computation of TTDA-TW subspaces is not expensive at lower storage costs.

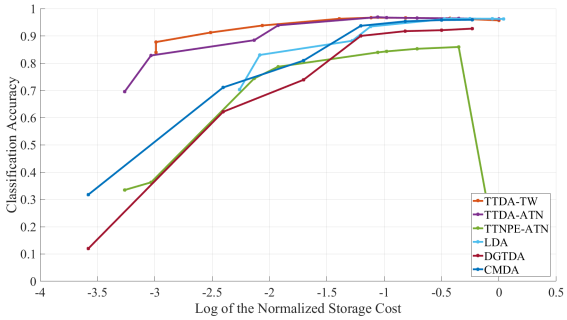
V. CONCLUSION

In this work, we proposed two novel supervised TT based algorithms for tensor object classification. First, we proposed to learn a TT subspace model to approximate the LDA projection matrix. Next, we proposed a two-way TT algorithm with better computational efficiency and fidelity to original data structure. The proposed approaches are tested on image classification tasks and compared to LDA, unsupervised TT and supervised Tucker decomposition based methods.

TTDA-ATN has been shown to be effective in classification tasks having low storage complexity while TTDA-TW achieves computational efficiency, low storage cost and high classification accuracy all at the same time.

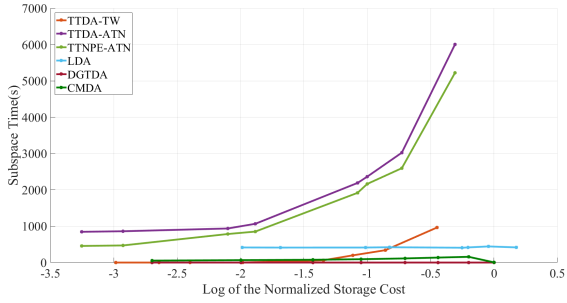


(a)

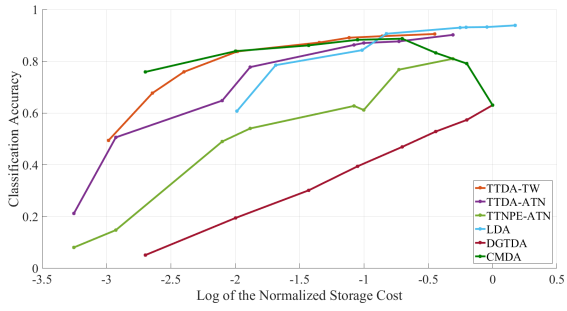


(b)

Fig. 1. Comparison of computation time, storage complexity and accuracy for COIL: (a) Subspace computation time vs storage complexity, (b) Classification accuracy vs storage complexity for different methods.



(a)



(b)

Fig. 2. Comparison of computation time, storage complexity and accuracy for Yale B: (a) Subspace computation time vs storage complexity, (b) Classification accuracy vs storage complexity for different methods.

REFERENCES

[1] A. Cichocki, "Era of big data processing: A new approach via tensor networks and tensor decompositions," *arXiv preprint arXiv:1403.2048*, 2014.

[2] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[3] 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, 2017.

[4] Q. Li and D. Schonfeld, "Multilinear discriminant analysis for higher-order tensor data classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2524–2537, 2014.

[5] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, and H.-J. Zhang, "Discriminant analysis with tensor representation," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 526–532.

[6] D. Tao, X. Li, X. Wu, and S. J. Maybank, "General tensor discriminant analysis and gabor features for gait recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, 2007.

[7] H. Wang, S. Yan, T. S. Huang, and X. Tang, "A convengent solution to tensor subspace learning." in *IJCAI*, 2007, pp. 629–634.

[8] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Mpca: Multilinear principal component analysis of tensor objects," *IEEE Transactions on Neural Networks*, vol. 19, no. 1, pp. 18–39, 2008.

[9] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, D. P. Mandic *et al.*, "Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives," *Foundations and Trends® in Machine Learning*, vol. 9, no. 6, pp. 431–673, 2017.

[10] J. A. Bengua, P. N. Ho, H. D. Tuan, and M. N. Do, "Matrix product state for higher-order tensor compression and classification," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 4019–4030, 2017.

[11] W. Wang, V. Aggarwal, and S. Aeron, "Tensor train neighborhood preserving embedding," *IEEE Transactions on Signal Processing*, vol. 66, no. 10, pp. 2724–2732, 2018.

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

[13] S. Holtz, T. Rohwedder, and R. Schneider, "On manifolds of tensors of fixed tt-rank," *Numerische Mathematik*, vol. 120, no. 4, pp. 701–731, 2012.

[14] Z. Wen and W. Yin, "A feasible method for optimization with orthogonality constraints," *Mathematical Programming*, vol. 142, no. 1-2, pp. 397–434, 2013.