Unsupervised Domain Adaptation using Parallel Transport on Grassmann Manifold

Ashish Shrivastava Sumit Shekhar Vishal M. Patel

UMIACS, University of Maryland College Park, MD
{ashish, sshekha, pvishalm}@umiacs.umd.edu

Abstract

When designing classifiers for classification tasks, one is often confronted with situations where data distributions in the source domain are different from those present in the target domain. This problem of domain adaptation is an important problem that has received a lot of attention in recent years. In this paper, we study the challenging problem of unsupervised domain adaptation, where no labels are available in the target domain. In contrast to earlier works, which assume a single domain shift between the source and target domains, we allow for multiple domain shifts. Towards this, we develop a novel framework based on the parallel transport of union of the source subspaces on the Grassmann manifold. Various recognition experiments show that this way of modeling data with union of subspaces instead of a single subspace improves the recognition performance.

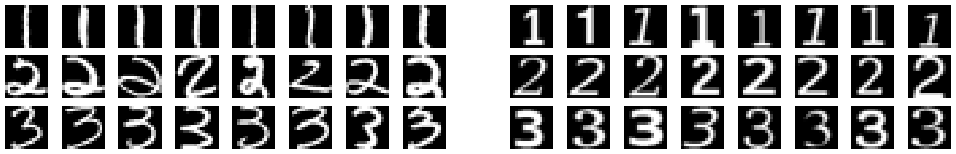
1. Introduction

Many machine learning problems learn a classification model with labeled training data, and using this model, predict the label of an unknown test sample. The fundamental assumption here is that the test data comes from the same distribution as the training data. However, in many practical cases, this does not hold. For instance, training data might be frontal faces captured under controlled illumination, while the test data consists of face images from the Internet, which can be different due to variations in sensor type, object pose, scene lighting, camera viewpoint, etc. Furthermore, as labeling the data requires significant human effort, there may not be enough labeled samples in the test domain. The problem of learning a good classifier for the test domain in such a scenario, is referred to as domain adaptation or domain transfer learning. Domain adaptation approach this problem by leveraging labeled data in a related domain, known as 'source' domain, when learning a classifier for unseen data in a 'target domain'. Although some special kinds of domain adaptation problems have been studied under different names such as covariate shift [27], class imbalance [16], and sample selection bias [15, 32], it has started gaining significant interest in computer vision only recently.

(a) (b) (c) (d)

Figure 1. Example images of two classes (keyboard and backpack) from four datasets: (a) Amazon, (b) Caltech, (c) DSLR (d) Webcam.

In this work, we focus on the challenging problem of unsupervised domain adaptation where the target domain does not have any labels. Various unsupervised domain adaptation methods have been proposed in the literature [3, 13, 12, 11, 21, 6]. However, a common assumption made in these approaches has been that the domain shift is global, irrespective of classes. This is, however, not true in many practical scenarios, as shown in Figures 1 and 2. Figure 1 shows examples of keyboard and backpack images in different domains. While backpack images show changes in shape and texture, keyboard images have variations in viewpoints, but not in texture. In Figure 2, we adapt a dataset of hand-written digits to computer generated digits such that their data distributions become closer to each other. It can be seen that the change in writ-



(a) Source domain. (b) Target domain.

Figure 2. Examples of (a) hand-written (source domain) digits, and (b) computer-typed (target domain) digits.

ing style is unique to each digit, and clearly, assuming a global domain shift is not optimal.

In our approach we assume that the data lies in a union of subspaces in both source as well as target domains. For the source domain, we assume that each class lies on a separate subspace which can be computed using Principal Component Analysis (PCA). However, for the unlabeled target domain, discovering the clusters corresponding to different classes is challenging. Several semi-supervised learning methods are available [5], which can be used to determine the target labels. But due to change in data distribution, such methods may not be desirable. Gong *et al.* [11] in their recent work describe a method to choose landmarks or source samples close to target samples for adaptation. Chen *et al.* [6] employed feature selection to choose features similar to both source and target. However, these approach are not effective for large domain shifts (*e.g.* frontal face to profile face adaptation). Hence, this is a chicken-and-egg problem, where one needs to match the source and the target distribution, and simultaneously update the target clusters. To this end, we propose an approach based on the parallel transport on Grassmann manifold to incrementally move the source subspaces towards target domain. The moved source subspaces are then used to improve the clustering in the target domain, and the steps are iterated. Having moved all the data and subspaces, a classification model is learned based on the representation of the labeled source samples on the moved subspaces. Given a novel test sample, its representation is similarly obtained, and the label is assigned using a linear support vector machine (SVM).

1.1. Organization of the paper

The paper is organized as follows. In Section 2, we review some recent domain adaptation methods. Various terms and notations are defined in Section 3. Details of the proposed method are given in Section 4. Computation of domain invariant features is then discussed in Section 5. Experimental results are presented in Section 6 and Section 7 concludes the paper with a brief summary and discussion.

2. Related work

Domain adaptation methods can be broadly divided into three main categories: supervised, unsupervised and semi-supervised. Since, our approach is unsupervised, in this section, we mainly review some of the recent related unsupervised domain adaptation methods.

Unsupervised adaptation was first proposed for natural language processing by Blitzer *et al.* [3] which introduced structural correspondence learning to automatically induce correspondences among features from different domains. The problem of domain adaptation was introduced for visual recognition by Saenko *et al.* [25, 19] in a semi-supervised setting. They learn a linear transformation that minimizes domain induced effects when data from both the domains are projected on it. Gopalan *et al.* [13] extended it to the unsupervised setting, using manifold-based interpolation to compute a domain-invariant feature. The idea of interpolation has been further explored in [12, 11, 21]. A co-training based adaptation method was presented in [6].

There are also some closely related but not equivalent machine learning problems that have been studied extensively, including transfer learning or multi-task learning [4], semi-supervised learning [5] and multiview analysis [26]. A review of domain adaptation methods from machine learning and the natural language processing communities can be found in [17]. A survey on the related field of transfer learning can be found in [22].

3. Background

In order to discuss our algorithm in detail, we need to first establish some terms and notations for the Grassmann manifold $\mathcal{G}_{n,d}$ which is a quotient space of special orthogonal group (of dimension n), denoted by $SO(n)$. Note that a matrix $\mathbf{Q} \in SO(n)$ is an orthogonal matrix, i.e. $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_n$, with determinant equal to $+1$, where $\mathbf{I}_n$ is $n \times n$ identity matrix. For a detailed discussion of these topics please refer to [7, 30, 8] and references therein.

3.1. Tangent Space

Tangent space at a point p on a manifold is a tangent plane, of the same dimension as that of the manifold, with origin at p. Let the subspace spanned by an orthonormal matrix $\mathbf{S}$ be denoted by $[\mathbf{S}] \in \mathcal{G}_{n,d}$, and orthogonal completion of $\mathbf{S}$ by $\mathbf{Q}_S$, i.e., $\mathbf{Q}_S \in SO(n)$ such that

$$\mathbf{S} = \mathbf{Q}_S^T \begin{bmatrix} \mathbf{I}_d \\ \mathbf{0}_{n-d\times d} \end{bmatrix},$$

where $\mathbf{I}_d$ is the identity matrix in $\mathbb{R}^{d\times d}$. Note that

$$\mathbf{Q}_S \triangleq [\mathbf{S} \;\; \mathbf{S}_\perp],$$

where $\mathbf{S}_\perp^T\mathbf{S}_\perp = \mathbf{I}_{(n-d)}$ and

$$\mathbf{S}_\perp^T\mathbf{S} = \mathbf{0}_{(n-d)\times d}.$$

Here, $(.)^T$ denotes the matrix transpose operator. The tangent space of $[\mathbf{S}]$ is given by,

$$T_{[\mathbf{S}]}(\mathcal{G}_{n,d}) := \{\mathbf{S}_\perp\mathbf{B} \quad | \quad \mathbf{B} \in \mathbb{R}^{(n-d)\times d}\}. \tag{1}$$

3.2. Exponential Map

Exponential map is a tool to map a point in the tangent plane to the manifold. Let $\mathbf{S}_\perp\mathbf{A} \in T_{[\mathbf{S}]}(\mathcal{G}_{n,d})$ be a point in tangent plane at point $[\mathbf{S}] \in \mathcal{G}_{n,d}$, then the exponential map,

$$\exp_{[\mathbf{S}]} : T_{[\mathbf{S}]}(\mathcal{G}_{n,d}) \to \mathcal{G}_{n,d},$$

is defined as

$$\exp_{[\mathbf{S}]}(\mathbf{A}) = \Psi_{\mathbf{A}}(1),$$

where $\Psi_{\mathbf{A}}(t)$ is a constant speed geodesic that starts at point $[\mathbf{S}]$, i.e., $\Psi_{\mathbf{A}}(0) = [\mathbf{S}]$, with initial velocity $\mathbf{A}$ and reaches $\Psi_{\mathbf{A}}(1)$ in unit time. For the Grassmann manifold, this geodesic is given by,

$$\mathbf{\Psi}_{\mathbf{A}}(t) = \mathbf{Q}_S \exp\left(t \begin{bmatrix} \mathbf{0} & \mathbf{A}^T \\ -\mathbf{A} & \mathbf{0} \end{bmatrix}\right)\mathbf{J}, \tag{2}$$

where, $\mathbf{Q}_S = [\mathbf{S} \;\; \mathbf{S}_\perp]$, $\mathbf{S}_\perp\mathbf{A} \in T_{[S]}(\mathcal{G}_{n,d})$, and

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_d \\ \mathbf{0}_{n-d\times d} \end{bmatrix} \in \mathbb{R}^{n\times d}.$$

Since the exponential map is the geodesic at $t = 1$, it is given by,

$$\exp_{[\mathbf{S}]}(\mathbf{A}) = \mathbf{Q}_S \exp\left(\begin{bmatrix} \mathbf{0} & \mathbf{A}^T \\ -\mathbf{A} & \mathbf{0} \end{bmatrix}\right)\mathbf{J}. \tag{3}$$

3.3. Parallel Transport

Let $\mathbf{\Delta} \triangleq \mathbf{S}_\perp\mathbf{B} \in \mathbb{R}^{n\times d}$ be a point on the tangent plane at the point $[\mathbf{S}] \in \mathcal{G}_{n,d}$. Then parallel transport of $\mathbf{\Delta}$, along the geodesic $\mathbf{\Psi}_{\mathbf{A}}(t)$ in direction $\mathbf{A}$, consists of moving $\mathbf{\Delta}$ at a new location such that it stays parallel to itself with respect to the geodesic. In other words, one can imagine moving $\mathbf{\Delta}$ in such a way that every infinitesimal step consists of a parallel displacement of $\mathbf{\Delta}$ in Euclidean space, followed by removal of the normal component. See Figure 3(a) for the illustration of this idea and refer to [8] for a detailed discussion. Parallel transport of $\mathbf{\Delta}$ for Grassmann manifold is given by,

$$\tau\mathbf{\Delta}(t) = \mathbf{Q}_S \exp\left(t \begin{bmatrix} \mathbf{0} & \mathbf{A}^T \\ -\mathbf{A} & \mathbf{0} \end{bmatrix}\right) \begin{bmatrix} \mathbf{0} \\ \mathbf{B} \end{bmatrix}, \tag{4}$$

where $\mathbf{B} \in \mathbb{R}^{(n-d)\times d}$ is the initial direction to reach from $\mathbf{S}$ to the exponential map of $\mathbf{\Delta}$, i.e. $\mathbf{\Delta} = \mathbf{S}_\perp\mathbf{B}$.

4. Proposed Framework

Let there be N_s training samples in the source domain, denoted by a matrix

$$\mathbf{Y}^s = [\mathbf{y}_1^s, \ldots, \mathbf{y}_{N_s}^s] \in \mathbb{R}^{n\times N_s}$$

with labels $\{l_i\}_{i=1}^{N_s} \in \{1, \ldots, C\}$, where n is the feature dimension and C is the number of classes. We denote the unlabeled samples at target domain by matrix

$$\mathbf{Y}^t = [\mathbf{y}_1^t, \ldots, \mathbf{y}_{N_t}^t] \in \mathbb{R}^{n\times N_t},$$

where N_t is the total number of samples in the target domain.

Our goal is to learn a classifier using both source and target data that can classify a novel test data that may come from any of these domains. Hence, we want to find a basis in which source as well as target data are well represented and, for a given class, representation of the source samples is similar to that of the target domain. Clearly, using the basis computed from the source data samples alone will represent the source domain data well but not the target domain data. Similarly, a basis computed from target data alone, will not represent the source data well. Hence, the learned classifier will not perform well when applied on the test data. Therefore, our approach is to learn source and target subspaces separately, and incrementally move the source domain subspaces towards the target domain subspaces. Towards achieving this goal, first, we assume that the source data lies in a union of subspaces. As explained later, we separately cluster the source and the target data into M clusters and discover M subspaces. Let the subspaces in the source and the

target domains be denoted by matrices $\{\mathbf{S}_m^s\}_{m=1}^M$ and $\{\mathbf{S}_m^t\}_{m=1}^M$, respectively. Denote the dimension of each subspace by (n, d), i.e., $\mathbf{S}_m^s, \mathbf{S}_m^t \in \mathbb{R}^{n\times d}$. These subspaces can be computed using PCA on the subset of samples. For the source domain, each subset belongs to a class, and for the target domain subsets are computed using the source subspaces as explained in section 4.2. Given the subspaces in the source and the target domains, first we compute separate Karcher means [18] of the source subspaces, denoted by $\boldsymbol{\mu}_S$ and the target subspaces, denoted by $\boldsymbol{\mu}_T$. A geodesic direction $\mathbf{A}$ is computed between these two means. Next, all the subspaces are parallelly translated along the geodesic with the initial direction $\mathbf{A}$. This parallel translation is done incrementally, and after moving all the subspaces along the geodesic by a small amount, the Karcher mean $\boldsymbol{\mu}_S$ is recomputed with the translated subspaces. In what follows, we describe these steps in detail and provide a step-by-step algorithm for computing intermediate subspaces. Figure 3(b) presents an overview of our method.

4.1. Computation of Subspaces in the Source Domain

With the assumption that the data lies on the union of subspaces, we compute total M subspaces. These can be computed by, first, clustering the source domain data into M clusters using any clustering algorithm, e.g. k-Means, sparse subspace clustering [9] etc. and, then, performing PCA on each cluster. However, since the labels are available in the source domain, we divide the data into C subsets (i.e. set $M = C$) according to their labels and perform the PCA for each subset independently. Thus, we get C different subspaces and denote them by matrices $\mathbf{S}_c^s \in \mathbb{R}^{n\times d}$, $c = 1, \cdots, C$. Note, that if the number of samples in each class is very small or there are too many classes, it will be prudent to perform a clustering algorithm to reduce the number of clusters.

4.2. Clustering of Target Data and Computation of Subspaces

Since we are not provided the labels in the target domain, one way to cluster the target domain samples is to perform a clustering algorithm. However, we find that clustering the target data using the source subspaces works better in practice. For each class $c = 1, \dots C$, we find a sample $\mathbf{y}_{i^*}^t$ in the target domain that is the closest to the source subspace corresponding to the c^{th} class, i.e.,

$$i^* = \arg \min_{i=1\dots,N_t} \|\mathbf{y}_i^t - \mathbf{S}_c^s \mathbf{S}_c^{sT} \mathbf{y}_i^t\|.$$

Then, m nearest neighbors of $\mathbf{y}_{i^*}^t$ are used to compute, using the PCA, the c^{th} class subspace $\mathbf{S}_c^t$ in the target domain.

4.3. Computation of Karcher Mean in the Source and the Target Domains

Our goal is to move the source subspaces towards the target subspaces, hence, we need to compute a direction in which each subspace should be moved. The most natural choice is to compute the direction between the means of the source and the target subspaces. Since each of the subspaces spanned by the matrices $\mathbf{S}_c^s$ and $\mathbf{S}_c^t$ lies on a Grassmann manifold $\mathcal{G}_{n,d}$, we compute the Karcher mean [18] that is consistent with the geometry of this space. The Karcher mean of the source subspace, $\boldsymbol{\mu}_S \in \mathbb{R}^{n\times d}$ and that of the target subspaces $\boldsymbol{\mu}_T \in \mathbb{R}^{n\times d}$ are computed using an iterative method as presented in Algorithm 1.

Algorithm 1: Algorithm for computing the Karcher mean of multiple subspaces on $\mathcal{G}_{n,d}$ [24].

Input: Set of C subspaces $\{\mathbf{S}_c\}_{c=1}^C \in \mathcal{G}_{n,d}$, maximum number of iterations T.
Output: Karcher mean $\boldsymbol{\mu}$.

Algorithm:
Randomly pick one of $\mathbf{S}_c$'s as initial estimate $\boldsymbol{\mu}_0$.
for $i = 1, \dots T$ **do**
1. Compute inverse exponential map

$$\boldsymbol{\nu}_c = \exp_{\boldsymbol{\mu}_i}^{-1}(\mathbf{S}_c), \forall c = 1, \dots C.$$

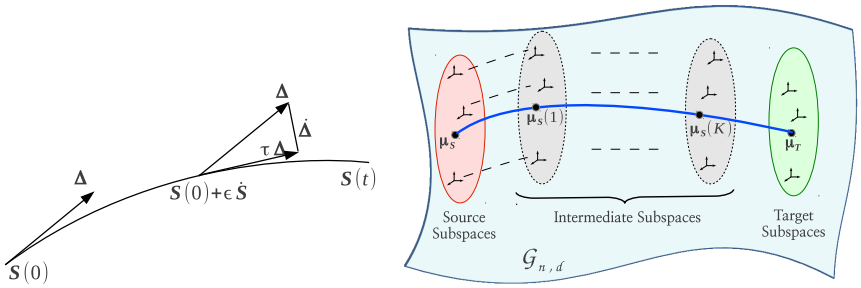
2. Compute average tangent vector

$$\bar{\boldsymbol{\nu}} = \frac{1}{C} \sum_{c=1}^{C} \boldsymbol{\nu}_c.$$

if $\|\bar{\boldsymbol{\nu}}\|$ *is small* **then**
break.
end
3. Move $\boldsymbol{\mu}_i$ in average tangent direction, i.e., $\boldsymbol{\mu}_{j+1} = \exp_{\boldsymbol{\mu}_j}(\epsilon\bar{\boldsymbol{\nu}})$, where $\epsilon \geq 0$ and typically set to 0.5.
end
return $\mathbf{A}$.

4.4. Computation of the Direction from Source to Target

Given two subspaces $\boldsymbol{\mu}_S$ and $\boldsymbol{\mu}_T$, we need to compute the initial direction $\mathbf{A}$ at the point $\boldsymbol{\mu}_S$ and, moving along the geodesic, reaches $\boldsymbol{\mu}_T$ in unit time. This can be computed using Algorithm 2.



(a) (b)

Figure 3. (a) Computation of parallel transport. (b) Illustration of parallel transport of union of subspace along the geodesic between the source and the target means.

4.5. Parallel Transport of the Source Subspaces

In order to parallelly transport all the source subspaces $\mathbf{S}_c^s$, along the direction $\mathbf{A}$, first we need to project these subspaces onto the tangent plane at $\boldsymbol{\mu}_S$. This can be done by computing directions $\mathbf{B}_c$, using Algorithm 2, such that a geodesic starting at $\boldsymbol{\mu}_S$ reaches $\mathbf{S}_c^s$ in unit time with initial velocity $\mathbf{B}_c$. Then, the projection of $\mathbf{S}_c^s$ onto the tangent plane at $\boldsymbol{\mu}_S$ is given by $\boldsymbol{\Delta}_c^s = \boldsymbol{\mu}_{S\perp}\mathbf{B}_c$, where $[\boldsymbol{\mu}_S \;\; \boldsymbol{\mu}_{S\perp}]$ is the orthogonal completion of $\boldsymbol{\mu}_S$. From (4), the parallel transport of $\boldsymbol{\Delta}_c^s$ at time t is given by,

$$\begin{aligned}\tau\boldsymbol{\Delta}_c^s(t) &= \mathbf{Q}_{\boldsymbol{\mu}_S} \exp\left(t \begin{bmatrix} \mathbf{0} & \mathbf{A}^T \\ -\mathbf{A} & \mathbf{0} \end{bmatrix}\right) \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_c \end{bmatrix} \\ &= [\boldsymbol{\mu}_S(t) \;\; \boldsymbol{\mu}_{S\perp}(t)] \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_c \end{bmatrix} = \boldsymbol{\mu}_{S\perp}(t)\mathbf{B}_c, \end{aligned} \quad (5)$$

where $\boldsymbol{\mu}_S(t)$ is the mean at time t after moving $\boldsymbol{\mu}_S$ towards $\boldsymbol{\mu}_T$. Note that $\tau\boldsymbol{\Delta}_c^s(t)$ is in the tangent plane at $\boldsymbol{\mu}_S(t)$, and in order to bring it back to the manifold we need to use exponential map defined in (3), i.e.,

$$\mathbf{S}_{ct}^s = [\boldsymbol{\mu}_S(t) \;\; \boldsymbol{\mu}_{S\perp}(t)] \exp\left(\begin{bmatrix} \mathbf{0} & \mathbf{B}_c^T \\ -\mathbf{B}_c & \mathbf{0} \end{bmatrix}\right)\mathbf{J}. \quad (6)$$

4.6. Overall Algorithm for Computing the Intermediate Union of Subspaces

Having described all the steps in the previous sections, we now summarize our approach. After computing the subspaces in source and target domains, we compute their respective Karcher means, $\boldsymbol{\mu}_S$ and $\boldsymbol{\mu}_T$, on the Grassmann manifold. Next, initial direction $\mathbf{A}$ is computed such that a geodesic starting from $\boldsymbol{\mu}_S$ reaches $\boldsymbol{\mu}_T$ in unit time. Also, for all $c = 1, \ldots, C$, we compute initial directions $\mathbf{B}_c$ such that a geodesic from $\boldsymbol{\mu}_S$ reaches $\mathbf{S}_c^s$ in unit time. All the directions can be computed using Algorithm 2. Then, we move $\boldsymbol{\mu}_S$ by t using the exponential map defined in (3), and parallel transport all the source subspaces using (6). Having moved the subspaces closer to the target domain, we re-compute the source mean using transported source subspaces. This process of moving the source subspaces and computing their mean is repeated until K intermediate subspaces $\mathbf{S}_c^s(k), k = 1, \ldots, K$ are computed. These steps are summarized in Algorithm 3. Furthermore, we compute K_1 (set less than 5 in our experiments) more intermediate subspaces between each pair of the translated source subspace $\mathbf{S}_c^s(K)$ and target subspaces $\mathbf{S}_c^t$. This is done by first computing the initial direction between $\mathbf{S}_c^s(K)$ and $\mathbf{S}_c^t$ (Algorithm 2) and then sampling the geodesic at K_1 equally spaced intermediate points using (2).

Algorithm 2: Algorithm for computing the direction between two subspaces on $\mathcal{G}_{n,d}$ [10].

Input: Two subspaces $\mathbf{S}^{(1)}, \mathbf{S}^{(2)} \in \mathbb{R}^{n,d}$.
Output: Initial velocity $\mathbf{A}$, such that $\mathbf{S}_\perp^{(1)}\mathbf{A} \in T_{[S^{(1)}]}(\mathcal{G}_{n,d})$.

Algorithm:

1. Compute orthogonal completion $\mathbf{Q}$ of $\mathbf{S}^{(1)}$, i.e. $\mathbf{Q} = [\mathbf{S}^{(1)} \;\; \mathbf{S}_\perp^{(1)}]$.
2. Compute thin CS decomposition of $\mathbf{Q}^T\mathbf{S}^{(2)}$, given by
$$\mathbf{Q}^T\mathbf{S}^{(2)} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix} = \begin{bmatrix} \mathbf{V}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2 \end{bmatrix} \begin{bmatrix} \Gamma \\ \Sigma \end{bmatrix} \mathbf{V}^T \; .$$
3. Compute $\{\theta_i\} = \arccos(\gamma_i) = \arcsin(\sigma_i)$, where γ_i, and σ_i are the diagonal elements of Γ and Σ, respectively.
4. Form a diagonal matrix $\boldsymbol{\Theta}$ with θ_i's in diagonal. Set $\mathbf{A} = \mathbf{V}_2\boldsymbol{\Theta}\mathbf{V}_1^T$.

return $\mathbf{A}$.

5. Computation of Features Invariant to Domain Shifts

We want to represent each sample on the intermediate union of subspaces. Since a sample is likely to

Algorithm 3: Algorithm for computing intermediate union of subspaces.

Input: Source domain data $\mathbf{Y}^s$, labels l, and unlabeled target domain data $\mathbf{Y}^t$, number of intermediate subspaces K, parameter t, subspace dimension d.

Output: Intermediate source subspaces $\mathbf{S}_c^s(k), k = 1, \ldots, K$.

Algorithm:

Compute subspaces, $\mathbf{S}_c^s$ and $\mathbf{S}_c^t, \forall c$ (Section 4.1).

Compute the Karcher means $\boldsymbol{\mu}_S$ and $\boldsymbol{\mu}_T$ (Section 4.3).

for $k = 1, \ldots, K$ **do**

1. Compute the direction $\mathbf{A}$ between $\boldsymbol{\mu}_S$ and $\boldsymbol{\mu}_T$ (Algorithm 2).
2. Compute the direction $\mathbf{B}_c$ between $\boldsymbol{\mu}_S$ and $\mathbf{S}_c^s$ (Algorithm 2).
3. Compute $\boldsymbol{\mu}_S(t)$ by moving $\boldsymbol{\mu}_S$ along geodesic with initial direction $\mathbf{A}$ using (2).
4. Compute $\mathbf{S}_{ct}^s, \forall c = 1, \ldots, C$, i.e. parallel transport of all the subspaces using (6).
5. Set $\mathbf{S}_c^s(k) = \mathbf{S}_{ct}^s$, and $\mathbf{S}_c^s = \mathbf{S}_{ct}^s$.
6. Compute the Karcher mean $\boldsymbol{\mu}_S$ using transported source subspaces $\mathbf{S}_c^s(k)$.

end

return $\mathbf{S}_c^s(k)$.

belong to one of the subspaces, we first concatenate all the subspaces $\mathbf{S}_c^s(k), \forall c = 1, \ldots, C$ at the k^{th} intermediate position and form a dictionary

$$\mathbf{D}_k \triangleq [\mathbf{S}_1^s(k), \ldots, \mathbf{S}_C^s(k)].$$

Now, if a sample $\mathbf{y}$ belongs to class c then it can be well represented by the elements of the subspace $\mathbf{S}_c^s(k)$. However, instead of enforcing a sample to belong to only one subspace, we relax this constraint and allow it to be represented by sparse linear combination of the elements of $\mathbf{D}_k$. In other words, if $\mathbf{y}$ belongs to c^{th} class and we write $\mathbf{y}$ as a linear combinations of samples from $\mathbf{D}_k$ then most of the coefficients corresponding to the $\mathbf{S}_c^s(k)$ will be non zero and the coefficients corresponding to the other subspaces are likely to be close to zero. One can find the sparse coefficients corresponding to $\mathbf{y}_i^s$ over the dictionary $\mathbf{D}_k$ by solving the following optimization problem,

$$\mathbf{x}_{ki}^s = \arg\min_{\mathbf{z}} \|\mathbf{y}_i^s - \mathbf{D}_k\mathbf{z}\|, \text{ subject to } \|\mathbf{z}\|_0 \leq T_0, \quad (7)$$

where $\|.\|_0$ is ℓ_0 norm which counts the number of non-zero elements in a vector and T_0 is the sparsity parameter which is set close to the dimension d of each subspace. This problem can be solved using a greedy algorithm like orthogonal matching pursuit (OMP) [23]. Once the sparse coefficients corresponding to all $\mathbf{D}_k$'s are found, they are concatenated to compute the domain invariant sparse representation of a source data sample. Having computed the domain shift invariant representations of the labeled source data, we use the linear SVM to learn a classification model. Given a novel test sample, similar to the source samples, we compute the concatenated sparse representation on the intermediate dictionaries $\mathbf{D}_k$'s and predict the label using the learned SVM model.

6. Experiments

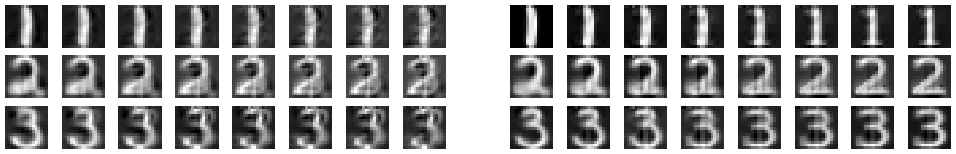
In order to evaluate the proposed method, we first analyze it on the digit dataset where we can visualize the intermediate subspaces. Furthermore, to get more quantitative comparisons, we perform experiments for object recognition tasks considering different datasets as different domains.

6.1. Domain Adaptation Between Hand-Written and Computer-Typed Digits

In order to visualize the intermediate subspaces, we evaluate our algorithm on three class digit data containing digits $1, 2$ and 3. For the source domain, we use the USPS digit dataset [1] and, for the target domain we use the computer generated digits of different fonts, as shown in Figure 2. For each digit in the target domain, we use 18 different fonts of normal, italic and boldface types, resulting in 54 samples per digit. Size of each digit image is 16×16 and we use 10-dimensional subspace for each class to represent them. Hence, the dimension of Grassmann manifold is $(256, 10)$. To visualize the intermediate subspaces, we reconstruct one digit from each class on the intermediate subspaces and compare our results with [13] in Figure 4(a) and 4(b). It clearly demonstrates that parallel transport of subspaces, can represent the data better than having a single subspace in the source and the target domain.

6.2. Object Recognition Across Datasets

For a quantitative evaluation of our method, we use four image datasets: Caltech, Amazon, DSLR, and Webcam. Each dataset can be thought of as a domain and it is generally true that classifiers trained on one dataset do not perform well when the test image is from a different dataset [25]. The Caltech dataset (also known as Caltech-256 [14]) has images of 256 object categories downloaded from Google images. The Amazon dataset contains images from online merchants (www.amazon.com) which are taken with studio light settings. The third domain, Digital SLR



(a) Single Subspace (b) Union of Subspaces

Figure 4. Reconstruction of a digit from each class on intermediate subspaces using (a) Single subspace model in source and target domain [13], and (b) the proposed union of subspaces model. First row shows the reconstruction of a randomly chosen digit 1 on the intermediate subspaces. Similarly, second and third row are the reconstruction results for digits 2 and 3, respectively.

(DSLR), has been prepared by capturing images with a high resolution (4288×2848) camera in realistic environment with natural lighting. Finally, the Webcam domain consists of images captured using a simple low resolution (640×480) webcam. The last three domains have been prepared by [25] and recently used by many visual domain adaptation papers [13, 12, 21] for evaluating their algorithms. Following the standard setting, we use 10 categories common across four domains. The example images from these categories are shown in Figure 5. Following settings in [21], we report our performance on eight different pairs of source and target domain combinations. For the webcam and the DSLR, 8 samples per category are used, while for the Amazon and the Caltech source domains number of samples per class is set to 20. For each image, 64 dimensional SURF features are computed at interest points found using the SURF detector. Next, using k-means clustering, a codebook of size 800 was generated using features of randomly chosen subset of Amazon dataset. Each image is represented using 800 dimensional histogram on this codebook. Based on this representation, we compare the proposed method with two single subspace-based methods proposed by Gopalan *et al.* [13] and Gong *et al.* [12], and a recently proposed dictionary learning-based method in [21]. As shown in Table 1, our method outperforms these methods. The subspace dimension for each class was empirically chosen between 5 and 15 and the number of intermediate subspaces were between 6 and 12 for all the source and target pairs. We believe that the main reason for the improved performance of our method is due to the multiple intermediate union of subspaces and sparse representation of the data. Constructing separate subspace for each class followed by sparse approximation of a test sample on the concatenation of all the subspaces is a popular idea in dictionary learning literature and has demonstrated a significant performance improvement in many computer vision tasks [31, 28, 20, 29].

7. Conclusion

We have proposed an unsupervised domain adaptation method based on the parallel transport of the source subspace for each class on the Grasmann manifold. The underlying assumption of our method what that the data lies in a union of subspaces in both source as well as target domains. Extensive experiments on the real datasets demonstrate the effectiveness of our approach on object recognition.

Acknowledgments

This work was partially supported by a MURI from the Office of Naval Research under the Grant 1141221258513.

References

[1] USPS Handwriten Digit Database. http://www-i6.informatik.rwth-aachen.de/ keysers/usps.html.

[2] M. Aharon, M. Elad, and A. Bruckstein. K-SVD : An algorithm for designing of overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[3] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *EMNLP*, 2006.

[4] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[5] O. Chapelle, B. Schölkopf, A. Zien, et al. *Semi-supervised learning*, volume 2. MIT press Cambridge, 2006.

[6] M. Chen, K. Weinberger, and J. Blitzer. Co-training for domain adaptation. In *NIPS*, 2011.

[7] Y. Chikuse. *Statistics on Special Manifolds*. Lecture notes in statistics. Springer, 2003.

[8] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[9] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *CVPR*, 2009.

[10] K. Gallivan, A. Srivastava, X. Liu, and P. V. Dooren. Efficient algorithms for inferences on grassmann manifolds. In *In IEEE Workshop on Statistical Signal Processing*, 2003.

[11] B. Gong, K. Grauman, and F. Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *ICML*, 2013.

[12] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.

Figure 5. Example images each of the 10 classes from four different domains. Each column corresponds to a class. From top to bottom the rows Amazon, Caltech, DSLR and webcam datasets, respectively.

Source Domain	Target Domain	K-SVD [2]	[21]	SGF [13]	GFK [12]	Proposed Method
Caltech	Amazon	20.5 ± 0.8	45.4 ± 0.3	36.8 ± 0.5	40.5 ± 0.7	$\mathbf{49.4 \pm 1.7}$
Caltech	DSLR	19.8 ± 1.0	42.3 ± 0.4	32.6 ± 0.7	41.1 ± 1.3	$\mathbf{48.2 \pm 2.8}$
Amazon	Caltech	20.2 ± 0.9	40.4 ± 0.5	35.3 ± 0.5	37.9 ± 0.4	$\mathbf{41.4 \pm 1.3}$
Amazon	Webcam	16.9 ± 1.0	37.9 ± 0.9	31.0 ± 0.7	35.7 ± 0.9	$\mathbf{40.4 \pm 1.4}$
Webcam	Caltech	13.2 ± 0.6	36.3 ± 0.3	21.7 ± 0.4	29.3 ± 0.4	$\mathbf{37.1 \pm 1.2}$
Webcam	Amazon	14.2 ± 0.7	38.3 ± 0.3	27.5 ± 0.5	35.5 ± 0.7	$\mathbf{38.7 \pm 1.0}$
DSLR	Amazon	14.3 ± 0.3	$\mathbf{39.1 \pm 0.5}$	32.0 ± 0.4	36.1 ± 0.4	38.2 ± 1.5
DSLR	Webcam	46.8 ± 0.8	$\mathbf{86.2 \pm 1.0}$	66.0 ± 0.5	79.1 ± 0.7	84.8 ± 1.4

Table 1. Comparison of unsupervised domain adaptation methods for object recognition task.

[13] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *ICCV*, 2011.

[14] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. In *Technical Report, Caltech*, 2007.

[15] J. J. Heckman. Sample selection bias as a specification error. *Econometric*, 47(1):153–161, 1979.

[16] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–450, 2002.

[17] J. Jiang. A literature survey on domain adaptation of statistical classifiers. *Tech report*, 2008.

[18] H. Karcher. Riemannian center of mass and mollifier smoothing. In *Communications on Pure and Applied Mathematics*, 1977.

[19] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR*, 2011.

[20] J. Mairal, F. Bach, J. Pnce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. 2008.

[21] J. Ni, Q. Qiu, and R. Chellappa. Subspace interpolation via dictionary learning for unsupervised domain adaptation. In *CVPR*, 2013.

[22] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[23] Y. C. Pati, R. Rezaiifar, Y. C. P. R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers*, pages 40–44, 1993.

[24] X. Pennec. Statistical computing on manifolds: From riemannian geometry to computational anatomy. In *Emerging Trends in Visual Computing*, 2008.

[25] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, volume 6314, pages 213–226, 2010.

[26] A. Sharma, A. Kumar, H. Daume, and D. Jacobs. Generalized multiview analysis: A discriminative latent space. In *CVPR*, pages 2160–2167, 2012.

[27] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[28] A. Shrivastava, H. V. Nguyen, V. M. Patel, and R. Chellappa. Design of non-linear discriminative dictionaries for image classification. In *ACCV*, 2012.

[29] A. Shrivastava, J. K. Pillai, V. M. Patel, and R. Chellappa. Learning discriminative dictionaries with partially labeled data. In *ICIP*, 2012.

[30] P. K. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa. Statistical computations on grassmann and stiefel manifolds for image and video-based recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(11):2273–2286, 2011.

[31] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. 2008.

[32] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *International Conference on Machine Learning*, pages 114–121, 2004.