Robust Domain Adaptation on the L1-Grassmannian Manifold

Sriram Kumar  Andreas Savakis

Rochester Institute of Technology, Rochester, New York
{sk3100, andreas.savakis}@rit.edu

Abstract

Domain adaptation aims to remedy the loss in classification performance that often occurs due to domain shifts between training and testing datasets. This problem is known as the dataset bias attributed to variations across datasets. Domain adaptation methods on Grassmann manifolds are among the most popular, including Geodesic Subspace Sampling and Geodesic Flow Kernel. Grassmann learning facilitates compact characterization by generating linear subspaces and representing them as points on the manifold. However, Grassmanian construction is based on PCA which is sensitive to outliers. This motivates us to find linear projections that are robust to noise, outliers, and dataset idiosyncrasies. Hence, we combine L1-PCA and Grassmann manifolds to perform robust domain adaptation. We present empirical results to validate improvements and robustness for domain adaptation in object class recognition across datasets.

1. Introduction

In practical machine learning problems, the data presented at test time can be quite different from the data used to train the classifier due to variations in pose or illumination, sensor variability and changes in the environment. For example, in Figure 1 sample images from three different domains are shown. The three domains have the same object categories, but contain visually dissimilar images due to domain shift. The process of adaptation comes naturally to humans, but it is hard to achieve in computer vision. This may be attributed to the fact that the model learned during training is biased on the particular training dataset [1]. Torralba and Efros [1] pointed out that each dataset has a distinct inherent bias or idiosyncrasy that causes the classifier to learn a biased model. This often results in poor cross-dataset generalization. The problem we address here is visual domain adaptation (DA), which falls under the category of transductive transfer learning, where the train and test data have the same object categories but the domain-shift is unknown [2].

One of the main problems in visual domain adaptation is how to select proper features, given that the nature of the domain-shift is unknown. Ben-David et al. [3] presented a theoretical analysis indicating the choice of features representing the domains is such that the divergence between the distributions in the feature space is minimized. There is need for a robust system that is able to perform reasonably well on any dataset without being idiosyncratic towards any particular one. Although this is desirable, it is hard to achieve.

There have been several approaches proposed to tackle this problem [4]. Subspace based methods try to find a latent space that is domain invariant, and then project the data from different domains onto this space where classification is performed. This paper introduces robustness to domain adaptation by incorporating recent work on L1-PCA. The particular methods explored are Geodesic Subspace Sampling (GSS) [5] and Geodesic Flow Kernel (GFK) [6].

The paper is organized as follows. Section 2 reviews the related work in visual domain adaptation. Section 3 reviews Grassmannian geometry and Section 4 overviews the PCA based Grassmann approaches used in this paper for evaluation. In Section 4 we present a robust approach for subspace generation during Grassmann manifold construction based on L1-PCA. We outline the experiment setup and related results in Section 5 and conclude in Section 6.
2. Related Work

Visual domain adaptation can be broadly categorized into semi-supervised and unsupervised. The former exploits the presence of target labels at hand. Successful approaches include transformative learning [19] and metric learning [2] based approaches. Although having access to few labelled samples from the target domain helps in improving performance, it is often difficult to acquire and in many applications not available. Unsupervised domain adaptation strategies often make use of linear representations such as principal component analysis (PCA) for domain representations [5, 6, 20-22] in a lower dimensional subspace. Although dimensionality reduction finds a low dimensional space common to both the domains, it does not guarantee the reduction in the divergence mismatch between the two domains.

Dimensionality reduction based DA methods are presented in [21, 22]. These approaches try to find a latent space by minimizing the mismatch in the distribution between the two domains using maximum mean discrepancy (MMD), a non-parametric method to compare two statistical distributions by mapping the data points to reproducible kernel Hilbert space. A limitation of [22] is the computation of the kernel matrix via semi-definite programming which can be computationally challenging.

Manifold alignment based methods find a projection that exploits the local geometry by preserving the local neighborhood information [23, 24]. In [27], adaptation is performed by aligning the basis vectors of the source domain to the target domain by learning a transformation that minimizes the Bregman divergence. Metric learning and canonical correlation analysis (CCA) methods have been explored for DA in [2, 25, 26]. In [25], they assume the existence of a linear predictor for both the domains. A robust approach based on low rank reconstruction that is similar to manifold learning was proposed in [26].

While the above approaches restrict to the use of source and target domain representation alone, Grassmannian based approaches exploit the intermediate representations [5, 6, 20]. In [5], intermediate subspaces are sampled from the geodesic curve on the manifold and combined to create a domain invariant space, while [6] and [20] integrate the subspaces on the geodesic between the source and target domain to learn a transformation matrix. In [21], a lower dimensional representation is learnt on the Grassmann manifold that minimizes the MMD in a Hilbert space.

Subspace based methods in general try to find a latent space that is domain invariant, and then project data from different domains onto this space, where classification is performed. This process is summarized in Figure 2. Under the paradigm of domain adaptation, the training data is called the source domain $D_S$ and the test data is called the target domain $D_T$. Domain adaptation deals with learning a classifier that performs well on the target domain, which is sampled from a different distribution than the source domain and often has few or no labeled samples.

Let $X_S$ and $X_T$ denote the samples present in the source and target domain respectively. $Y_S$ and $Y_T$ are the class labels corresponding to the conditional probabilities $P(Y_S|X_S)$ and $P(Y_T|X_T)$. Under transductive transfer learning, $P(Y_S|X_S) \approx P(Y_T|X_T)$ while $P(X_S) \neq P(X_T)$ [4]. This kind of scenario occurs frequently in computer vision problems such as face and object recognition in the wild. In the subsequent sections we discuss the techniques used for analysis.

\[ \theta = \theta_1, \ldots, \theta_m \]

Figure 3. Mapping from Euclidean space to Grassmann space. The subspaces $Y_1$ and $Y_2$ are mapped as points on the Grassmann manifold. The angle $\theta$ is a metric used to define the similarity between the two subspaces spanned by $Y_1$ and $Y_2$.

3. Grassmannian Domain Adaptation

3.1. Grassmannian Geometry

Grassmann manifolds are a special class of Riemannian manifolds. The Grassmann manifold $G(m, D)$ is defined as the set of all $m$-dimensional linear subspaces in $\mathbb{R}^D$ [7]. A simple visualization of the Grassmann manifold is shown in Figure 3. Let $Y_1$ and $Y_2$ be the representations of linear subspaces corresponding to two different image sets. These subspaces are mapped as two different points on the Grassmann manifold. Various metrics on the Grassmannian based on the angle $\theta$ between subspaces have been considered [8].
Sample points along the flow by substituting different value for \( t \) between 0 and 1, to obtain intermediate representations. d) Project the source and target data onto these intermediate subspaces and perform one nearest neighbor.

**Algorithm 1:** Computing the direction matrix via inverse exponential map [10]

1. Given the source \( P_S \) and target \( P_T \) subspaces, and the tangent vector \( B = \begin{bmatrix} 0 & A^T \\ -A^T & 0 \end{bmatrix} \).
2. Find the orthogonal completion \( Q \) of \( P_S \).
3. Perform economic CS decomposition of \( Q^T P_T \) given by, \( Q^T P_T = \begin{bmatrix} X_A \\ Y_A \end{bmatrix} = \begin{bmatrix} U_1 & 0 \\ 0 & D_2 \end{bmatrix} \begin{bmatrix} \Gamma(1) \\ -\Sigma(1) \end{bmatrix} V_1^T \).
4. Compute \( \theta_i \) given by \( \arccos(\gamma_i) \) and \( \sigma_i = \sin(\theta_i) \). Construct a diagonal matrix \( \Theta \), using \( \theta \) as its diagonal elements.
5. Compute the direction matrix \( A = \bar{U}_2 \Theta U_1^T \).

**Algorithm 2:** Computing the Exponential Map and sampling along the geodesic [10]

1. Given a point on the Grassmann manifold \( P_S \), and the tangent vector \( B = \begin{bmatrix} 0 & A^T \\ -A^T & 0 \end{bmatrix} \).
2. Find the orthogonal completion \( Q \) of \( P_S \).
3. Compute the compact SVD of the direction matrix \( A = \bar{U}_2 \Theta U_1^T \).
4. Compute the diagonal matrices \( \Gamma(t) \) and \( \Sigma(t) \) such that \( \gamma_i = \cos(t\theta_i) \) and \( \sigma_i = \sin(t\theta_i) \), where \( \theta \)'s are the diagonal elements of \( \Theta \).
5. Compute \( \phi(t) = Q \left( U_1 \Gamma(t) \right) \), for different values of \( t \in [0,1] \).

Despite its promising results and intuitive idea, GSS had some limitations. The first one is that there is no proper way to sample the intermediate points. Next is how to select the number of subspaces to retain for each domain. These parameters need to be tuned. Hence, the number of samples is determined empirically through cross-validation, which is cumbersome and time consuming.

### 3.3. Geodesic Flow Kernel (GFK)

Gong et al [6], proposed a new approach which integrates all the intermediate subspaces between the source and target subspaces. The geodesic distance is the shortest distance connecting between two points on the Grassmann manifold. One can think of the geodesic as the line connecting the two subspaces in the manifold. The intermediate points on the geodesic line represent the intermediate subspaces between the two.
A pictorial representation of the GFK process is shown in Figure 5. We denote the source and target subspace as $P_S$ and $P_T$. Intuitively, this works as all the subspaces between the source and target domains are utilized unlike the previous approach where one samples intermediate points along the geodesic between $P_S$ and $P_T$. Their approach consisted of two major steps: (a) construct a geodesic curve between the two subspaces, (b) find a kernel that integrates all the intermediate subspaces, in this case it is the Geodesic Flow Kernel.

The geodesic curve is a generalization of the straight line on curved surfaces. If we try to parameterize the curve by $t \in [0,1]$, and we define a subspace as $\Phi(t)$, in which at $t = 0$ the subspace is represents the source domain, $P_S$ and at $t = 1$ the subspace is represents the target domain, $P_T$. There are infinite number of subspaces between the source and target domains. Projection of feature vector on a subspace is given by $\Phi(t)^T x$. Such a projection enables representations that is invariant and unbiased to both source and target domains. If $x_i$ and $x_j$ represent the features from source and target domain respectively, the flow kernel can be obtained as follows:

$$
\langle x_i^{\Phi} | x_j^{\Phi} \rangle = \int_0^1 (\Phi(t)^T x_i) (\Phi(t)^T x_j) \, dt
= \int_0^1 x_i^T (\Phi(t) \Phi(t)^T) x_j \, dt
= x_i G_K x_j,
$$

where, $G_K \in \mathbb{R}^{D \times D}$ is a positive definite matrix. By doing this, the feature vectors are projected on infinite number of subspaces implicitly that are lying in between. This $G_K$ is essentially the kernel matrix and accomplishes the “kernel trick”. The elements of $G_K$ are the pairwise dot products between the subspaces. The closed form solution for $G_K$ is given by

$$
G_K = [P_S U_1 \quad R_S U_2] \begin{bmatrix}
A_1 & A_2 \\
A_2 & A_3
\end{bmatrix}
[U_1^T P_S^T] [U_2^T R_S^T],
$$

where $A_1$, $A_3$ are the diagonal elements and $A_2$ is the off diagonal elements and are defined as follows:

$$
\lambda_{1l} = 1 + \frac{\sin 2\theta_l}{2\theta_l}, \quad \lambda_{2l} = \frac{\cos 2\theta_l - 1}{2\theta_l}, \quad \lambda_{3l} = 1 - \frac{\sin 2\theta_l}{2\theta_l}
$$

Although these methods work well, they are often susceptible to noise and are not robust [6, 11]. This is due the fact that the subspaces are generated using the $L_2$-norm constraint. Hence, we use the $L_1$-norm constraint to generate our subspaces. The robustness of $L_1$-based PCA has been shown to work well for face recognition [16]. We show that by combining the $L_1$-norm for generating subspaces in the Grassmann framework, we obtain better recognition rates.

4. Robust Grassmann Learning

Subspace generation is an important part of constructing the Grassmann manifold. Traditional PCA is based on $L_2$-norm and as such it is susceptible to noisy projections. This affects the recognition process and a robust alternative is needed. By using $L_1$-norm based PCA, we can extract robust projections and thus create a robust Grassmann manifold. Below we review the $L_2$-norm and $L_1$-norm approaches for PCA subspace generation.

4.1. $L_2$ - Principal Component Analysis

Principal Component Analysis provides a signal decomposition using a linear combination of orthogonal basis vectors. The principal components are computed by finding the eigenvectors of the covariance matrix. The contribution of each component for a particular signal can reconstruct the original signal from the principal components. This is achieved by reducing the squared loss between the original signal and its reconstruction. For example if we define the matrix $X \in \mathbb{R}^{D \times N}$ and $D \gg N$ such that each column represents a signal after vectorising. The principal components can be found by solving:

$$
(XX^T)V = VA
$$

where $V$ is the matrix of eigenvectors of $(XX^T)$ and $A$ is a diagonal matrix, containing the eigenvalues associated with each eigenvector.

4.2. $L_1$ - Principal Component Analysis

In this section we begin with $L_2$-PCA and move into the formulation of $L_1$-PCA. Again, the data matrix is $X = [x_1, x_2, ..., x_N], X \in \mathbb{R}^{D \times N}$, where $D$ is the dimension of the data and $N$ is the number of samples. Assuming the data is zero mean, in $L_2$-PCA formulation we try to
minimize the projection error which is equivalent to maximizing the variance in the data.

\[ E_2 = \|X - RV\|_2^2 \]  

(6)

where, \( E \) is the reconstruction error, \( R \in \mathbb{R}^{D \times m} \) and \( V \in \mathbb{R}^{m \times N} \) given by

\[ V = R^TX \]  

(7)

\[ R_{L_2} = \arg \max_R \|X^TR\|_2, \]  

(8)

where \( R \) is the matrix containing columns denoting the principal components. The problem with this approach is that the error value is sensitive to outliers [12]. Even a single outlier can affect the direction of the principal components. In order to make it robust, we may formulate the problem under the \( L_1 \) norm. Under the \( L_1 \) formulation, Equations 6 and 8 become

\[ E_1 = \|X - RV\|_1 \]  

(9)

\[ R_{L_1} = \arg \max_R \|X^TR\|_1. \]  

(10)

As a consequence of this reformulation, Equations 9 and 10 are no longer equivalent under \( L_1 \) constraints. The optimal solution to the above \( L_1 \) problem is \( NP \)-hard [13] and suboptimal approaches have been developed [14]. To find one optimal principal component, Equation 10 can be rewritten as,

\[ r_{L_1} = \arg \max_{r \in \mathbb{R}^{D \times 1}, r^TR=1} \|X^Tr\|_1. \]  

(11)

It has been shown [13] that the optimal solution for a single \( L_1 \) principal component is given by,

\[ r_{L_1} = \frac{Xb_{\text{opt}}}{\|Xb_{\text{opt}}\|_2}, \]  

(12)

where

\[ b_{\text{opt}} = \arg \max_{b \in \{\pm 1\}^N} \|xb\|_2 \]  

(13)

The vector \( b_{\text{opt}} \) is a binary vector having length \( N \) and entries either \( 1 \) or \( -1 \). It has also been shown in [13] that \( \|X^Tr_{L_1}\|_1 = \|Xb_{\text{opt}}\|_2 \). Thus, after finding \( b_{\text{opt}} \) it is straightforward to obtain \( r_{L_1} \), the \( L_1 \) principal component. A fast computation of the principal component was proposed in [15] and is discussed next.

4.2.1 Fast Computation of Eigenvectors  

Kundu et al [15] introduced a new approach for fast computation of one principal component. We briefly explain the method in this section. A binary vector is initialized from the column of covariance matrix of the input data. The bits that contribute negatively to the projection energy are flipped and the process is repeated until the optimal binary vector is found. The quadratic form of Equation 12 is given by,

\[ b^TX^TXb = \text{Trace}(X^TX) + \sum_{i} 2b_i \left\{ \sum_{j \neq i} b_j (X^TX)_{i,j} \right\} \]  

(14)

where \( i, j \in 1, 2, ..., N \).

In [15], the expression \( \alpha \) is defined, that finds the bits that contribute negatively to each location.

\[ \alpha_i \equiv \pm 4b_i \sum_{j \neq i} b_j (X^TX)_{i,j}. \]  

(15)

The bits that negatively contribute are found through this process and are flipped to obtain \( b_{\text{opt}} \). After finding \( b_{\text{opt}} \) we can determine the corresponding eigenvector using Equation 12.

This approach was extended to obtaining multiple eigenvectors with a greedy strategy that preserves orthogonality [14]. The pseudo code for the greedy approach is presented below in Algorithm 3. After we find the first principal component, we remove its contribution from the data and find the next principal component using the bit flipping method. This guarantees orthogonality of the principal components [14] and the process is repeated until all principal components are found.

**Algorithm 3: Greedy search for \( L_1 \)-PCA [14]**

```
For j = 2:m
    // m is the number of principal components
    x_i^j = x_i^{j-1} - r_j-1(r_j^{T}x_i^{j-1}) \forall i \in \{1, ..., N\}
    X^j = [x_1^j, x_2^j, ..., x_N^j]
```

4.3. \( L_1 \)-Grassmann manifold  

During Grassmann manifold construction, PCA is the common method for subspace generation. The apparent problem is that the method lacks robustness to outliers and noise, while blindly trying to find the projection along the direction maximum variance. This leads to finding projections that may point towards noise, which is not desirable [11, 14].

In order to obtain a robust subspace that is not susceptible to noise or outliers, we utilize the \( L_1 \)-PCA to obtain the subspaces \( P_S \) and \( P_T \) from source and target domain data \( X_S \) and \( X_T \) respectively. The \( L_1 \)-PCA has shown promising results in face recognition with noise [16]. We maintain that by integrating \( L_1 \)-PCA approach for Grassmann framework, we might get a more robust subspace. The \( L_1 \)-Grassmannian [11] approach for subspace mapping will be robust to the noise that may occur while mapping the subspace on the Grassmann manifold.
5. Experimental Results

In this section we start by discussing the datasets used and report empirical results on two subspace based visual domain adaptation methods, namely Geodesic Subspace Sampling [5] and Geodesic Flow Kernel [6]. We perform object recognition experiments and show the validity of the proposed robust Grassmann manifold approach.

5.1. Object Class Recognition

For this experiment, we use the Office and Caltech-256 datasets. The office dataset contains images taken from Webcam, DSLR camera and Amazon. In each domain there are a total of 31 categories, such as headphones, monitor, laptop, cycle, etc. In addition, images from Caltech–256 database were used. The categories which overlapped with the other three domains were selected from the Caltech database.

We used the precomputed features provided in [2]. In short, SURF (Speeded-Up Robust Features) features, were extracted from images in the Amazon dataset and a codebook was learnt with 800 codewords by selecting a subset of features and by performing K-means clustering. Finally, each image was represented by a histogram of 800 codewords. This representation was used to represent images in each domain. The features were normalized to have zero mean and unit standard deviation.

We perform object recognition experiments on the Office and Caltech-256 datasets. Sample images are shown in Figure 1. We denote the four domains as A, C, W and D for Amazon, Caltech-256, Webcam and DSLR respectively. There are in total 12 possible combinations of datasets for the Domain Adaptation tasks.

We report the recognition rates for unsupervised object recognition in Tables 1 and 2. We compared GSS and GFK, the two Grassman manifold based techniques, with standard subspace generation using \( L_2 \)-PCA and robust subspace generation using \( L_1 \)-PCA. We report the average accuracies over the trials based on one nearest neighbor and SVM. For SVM classification, we used the kernelized version where we precomputed the kernel and cross-validated for the parameter C. For SVM computation, we used the libsvm package [17]. For GSS, we used Partial Least Squares [18] to learn discriminative classifier.

6. Analysis of Results

In the unsupervised learning setting, the labels of images in the source domain are known but those of the images in the target domain are unknown. For each domain we generate \( L_1 \)-norm constrained basis vectors such that, the number of eigenvectors is same as the dimension of the data. After this, we performed domain adaptation using Geodesic Subspace Sampling (GSS) and Geodesic Flow Kernel (GFK). We performed 20 random trials, for which we randomly choose a subset of samples from the source domain and performed the domain adaptation on the target domain. We empirically chose the optimal value for subspace dimensions. For the subspace generation, we used \( L_1 \)-PCA and \( L_2 \)-PCA for both the source and target domains and evaluated the two approaches. In specific, for the webcam-DSLR pair the accuracy of the GFK approach improved from 78.79% to 87.19%.

6.1. Finding the optimal \( L_1 \) subspace dimensionality

In order to estimate the optimal subspace dimension for each of the 12 domain pairs, we varied the subspace dimensions from 5 to 200. The variation of accuracy against the subspace dimension for four domain pairs based on 1-NN approach is shown in Figure 6 and 7. Similar results were obtained for both GSS and GFK. The optimal dimension was found to be around 20-30.
were analyzed and evaluated.

Geodesic Subspace Sampling and Grassmannian manifold proves the recognition rate. Two popular Grassmann manifold based domain adaptation techniques, Geodesic Subspace Sampling and Geodesic Flow Kernel, were analyzed and evaluated.

The $L_1$ approaches achieved improvements over the standard methods across multiple domain pairs. In particular, the $L_1$ Grassmannian approaches boosted the accuracy significantly when the domain pairs were visually similar.

| Table 1. Recognition accuracy (in %) with unsupervised Domain Adaptation using NN classification. Datasets: A: AMAZON, C: CALTECH, D: DSLR, W: WEBCAM |
| --- | --- | --- | --- | --- | --- | --- |
| $C$→$A$ | $34.15\pm1.8$ | $34.93\pm2.2$ | $36.75\pm2.1$ | $37.88\pm2.0$ |
| $C$→$W$ | $28.47\pm4.3$ | $28.71\pm3.7$ | $34.19\pm4.7$ | $34.37\pm4.3$ |
| $C$→$D$ | $32.61\pm4.5$ | $34.04\pm4.3$ | $35.25\pm3.2$ | $38.12\pm3.8$ |
| $A$→$C$ | $33.01\pm2.2$ | $33.54\pm1.6$ | $35.66\pm1.3$ | $34.62\pm1.6$ |
| $A$→$W$ | $30.10\pm3.7$ | $32.41\pm4.3$ | $36.08\pm2.9$ | $37.76\pm3.5$ |
| $A$→$D$ | $29.52\pm3.8$ | $32.48\pm3.9$ | $36.37\pm4.2$ | $33.66\pm2.9$ |

| Table 2. Recognition accuracy (in %) with unsupervised Domain Adaptation using SVM classification. Datasets: A: AMAZON, C: CALTECH, D: DSLR, W: WEBCAM |
| --- | --- | --- | --- | --- | --- |
| $W$→$C$ | $25.45\pm1.6$ | $26.53\pm1.7$ | $29.93\pm1.2$ | $27.80\pm0.6$ |
| $W$→$A$ | $31.96\pm2.1$ | $32.53\pm1.9$ | $30.27\pm1.5$ | $31.64\pm1.5$ |
| $W$→$D$ | $73.47\pm3.0$ | $75.54\pm2.5$ | $78.79\pm2.2$ | $87.19\pm2.8$ |
| $D$→$C$ | $28.05\pm1.7$ | $30.00\pm1.8$ | $29.64\pm1.3$ | $29.39\pm1.0$ |
| $D$→$A$ | $28.19\pm3.4$ | $30.36\pm2.3$ | $33.06\pm1.8$ | $35.25\pm1.6$ |
| $D$→$W$ | $65.10\pm3.5$ | $68.20\pm2.7$ | $74.49\pm2.3$ | $78.17\pm2.4$ |

In Figure 7, it can be observed that as the dimensionality increases above a certain point, the performance decreases and finally tapers off. This is attributed to the fact that higher dimensions begin to overfit to the source domain and do not work as well for targets domains that appear visually similar such as webcam and DSLR benefit the most from the proposed approach.

The results in Tables 1 and 2 shown in bold indicate improvement when using the $L_1$ approaches. In order to validate the significance of the proposed approaches we conducted two-way ANOVA (analysis of variance) across different domain pairs. The results which are underlined in Tables 1 and 2 indicate statistically significant with 95% confidence. This shows that the results obtained with the $L_1$ approach are significant and not due to chance.

<table>
<thead>
<tr>
<th>Method</th>
<th>$W$→$C$</th>
<th>$W$→$A$</th>
<th>$W$→$D$</th>
<th>$D$→$C$</th>
<th>$D$→$A$</th>
<th>$D$→$W$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_2$-GSS [5]</td>
<td>$46.57\pm3.8$</td>
<td>$39.83\pm4.1$</td>
<td>$42.39\pm4.9$</td>
<td>$40.49\pm2.3$</td>
<td>$40.49\pm3.0$</td>
<td>$38.41\pm3.5$</td>
</tr>
<tr>
<td>$L_1$-GSS (ours)</td>
<td>$46.69\pm3.2$</td>
<td><strong>41.00±3.8</strong></td>
<td>$42.32±4.2$</td>
<td><strong>41.03±1.6</strong></td>
<td>$40.61±2.8$</td>
<td>$38.98±3.7$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>$L_2$-GFK [6]</th>
<th>$L_1$-GFK (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$→$C$</td>
<td>$30.51±1.7$</td>
<td>$31.24±1.4$</td>
</tr>
<tr>
<td>$W$→$A$</td>
<td>$34.87±2.4$</td>
<td><strong>37.71±2.1</strong></td>
</tr>
<tr>
<td>$W$→$D$</td>
<td><strong>74.81±3.3</strong></td>
<td>$79.36±2.8$</td>
</tr>
<tr>
<td>$D$→$C$</td>
<td>$32.89±2.0$</td>
<td><strong>34.82±2.4</strong></td>
</tr>
<tr>
<td>$D$→$A$</td>
<td>$30.27±1.5$</td>
<td>$36.47±2.5$</td>
</tr>
<tr>
<td>$D$→$W$</td>
<td>$72.93±3.3$</td>
<td><strong>75.68±2.5</strong></td>
</tr>
</tbody>
</table>

7. Conclusion

In this paper we explored robust domain adaptation methods by learning the $L_1$-Grassmannian. Our method of generating robust subspaces is attributed to the $L_1$-norm and improves the recognition rate. Two popular Grassmann manifold based domain adaptation techniques, Geodesic Subspace Sampling and Geodesic Flow Kernel, were analyzed and evaluated.

References


