Learning an Invariant Hilbert Space for Domain Adaptation

Samitha Herath$^{1,2}$, Mehrtash Harandi$^{1,2}$ and Fatih Porikli$^1$
$^1$Australian National University, $^2$DATA61-CSIRO
Canberra, Australia
{samitha.herath, mehrtash.harandi}@data61.csiro.au, fatih.porikli@anu.edu.au

Abstract

This paper introduces a learning scheme to construct a Hilbert space (i.e., a vector space along its inner product) to address both unsupervised and semi-supervised domain adaptation problems. This is achieved by learning projections from each domain to a latent space along the Mahalanobis metric of the latent space to simultaneously minimizing a notion of domain variance while maximizing a measure of discriminatory power. In particular, we make use of the Riemannian optimization techniques to match statistical properties (e.g., first and second order statistics) between samples projected into the latent space from different domains. Upon availability of class labels, we further deem samples sharing the same label to form more compact clusters while pulling away samples coming from different classes. We extensively evaluate and contrast our proposal against state-of-the-art methods for the task of visual domain adaptation using both handcrafted and deep-net features. Our experiments show that even with a simple nearest neighbor classifier, the proposed method can outperform several state-of-the-art methods benefiting from more involved classification schemes.

1. Introduction

This paper presents a learning algorithm to address both unsupervised [21, 16, 49] and semi-supervised [27, 14, 29] domain adaptation problems. Our goal here is to learn a latent space in which domain disparities are minimized. We show such a space can be learned by first matching the statistical properties of the projected domains (e.g., covariance matrices), and then adapting the Mahalanobis metric of the latent space to the labeled data, i.e., minimizing the distances between pairs sharing the same class label while pulling away samples with different class labels. We develop a geometrical solution to jointly learn projections onto the latent space and the Mahalanobis metric there by making use of the concepts of Riemannian geometry.

Thanks to deep learning, we are witnessing rapid growth in classification accuracy of the imaging techniques if substantial amount of labeled data is provided [35, 48, 25, 26]. However, harnessing the attained knowledge into a new application with limited labeled data (or even without having labels) is far beyond clear [33, 37, 19, 8, 51]. To make things even more complicated, due to the inherent bias of datasets [50, 47], straightforward use of large amount of auxiliary data does not necessarily assure improved performances. For example, the ImageNet [43] data is hardly useful for an application designed to classify images captured by a mobile phone camera. Domain Adaptation (DA) is the science of reducing such undesired effects in transferring knowledge from the available auxiliary resources to a new problem.

The most natural solution to the problem of DA is by identifying the structure of a common space that minimizes a notion of domain mismatch. Once such a space is obtained, one can design a classifier in it, hoping that the classifier will perform equally well across the domains as the domain mismatched is minimized. Towards this end, several studies assume that either 1. a subspace of the target domain is the right space to perform DA and learn how the source domain should be mapped onto it [45, 29], or 2. subspaces obtained from both source and target domains are equally important for classification, hence trying to either learn their evolution [22, 21] or similarity measure [46, 52, 14].

Objectively speaking, a common practice in many solutions including the aforementioned methods, is to simplify the learning problem by separating the two elements of it. That is, the algorithm starts by fixing a space (e.g., source subspace in [16, 29]), and learns how to transfer the knowledge from domains accordingly. A curious mind may ask why should we resort to a predefined and fixed space in the first place.

In this paper, we propose a learning scheme that avoids such a separation. That is, we do not assume that a space or a transformation, apriori is known and fixed for DA. In

---

$^1$In DA terminology target domain refers to the data directly related to the task. Source domain data is used as the auxiliary data for knowledge transferring.
essence, we propose to learn the structure of a Hilbert space (i.e., its metric) along the transformations required to map the domains onto it jointly. This is achieved through the following contributions,

(i) We propose to learn the structure of a latent space, along its associated mappings from the source and target domains to address both problems of unsupervised and semi-supervised DA.

(ii) Towards this end, we propose to maximize a notion of discriminatory power in the latent space. At the same time, we seek the latent space to minimize a notion of statistical mismatch between the source and target domains (see Fig. 1 for a conceptual diagram).

(iii) Given the complexity of the resulting problem, we provide a rigorous mathematical modeling of the problem. In particular, we make use of the Riemannian geometry and optimization techniques on matrix manifolds to solve our learning problem.

(iv) We extensively evaluate and contrast our solution against several baseline and state-of-the-art methods in addressing both unsupervised and semi-supervised DA problems.

2. Proposed Method

In this work, we are interested in learning an Invariant Latent Space (ILS) to reduce the discrepancy between domains. We first define our notations. Bold capital letters denote matrices (e.g., $X$) and bold lower-case letters denote column vectors (e.g., $x$). $I_n$ is the $n \times n$ identity matrix. $S_n^{++}$ and $\text{St}(n, p)$ denote the SPD and Stiefel manifolds, respectively, and will be formally defined later. We show the source and target domains by $X_s \subset \mathbb{R}^n$ and $X_t \subset \mathbb{R}^t$. The training samples from the source and target domains are shown by $\{x_i^s, y_i^s\}_{i=1}^{n_s}$ and $\{x_i^t\}_{i=1}^{n_t}$, respectively. For now, we assume only source data is labeled. Later, we discuss how the proposed learning framework can benefit from the labeled data.

Our idea in learning an ILS is to determine the transformations $\mathbb{R}^{s \times p} \ni W_s : X_s \to \mathcal{H}$ and $\mathbb{R}^{t \times p} \ni W_t : X_t \to \mathcal{H}$ from the source and target domains to a latent p-dimensional space $\mathcal{H} \subset \mathbb{R}^p$. We furthermore want to equip the latent space with a Mahalanobis metric, $M \in S^{++}_p$, to reduce the discrepancy between projected source and target samples (see Fig. 1 for a conceptual diagram). To learn $W_s$, $W_t$ and $M$ we propose to minimize a cost function in the form

$$\mathcal{L} = \mathcal{L}_d + \lambda \mathcal{L}_u.$$  

In Eq. 1, $\mathcal{L}_d$ is a measure of dissimilarity between labeled samples. The term $\mathcal{L}_u$ quantifies a notion of statistical dif-

2Our implementation is available on https://sherath@bitbucket.org/sherath/ils.git.

ference between the source and target samples in the latent space. As such, minimizing $\mathcal{L}$ leads to learning a latent space where not only the dissimilarity between labeled samples is reduced but also the domains are matched from a statistical point of view. The combination weight $\lambda$ is envisaged to balance the two terms. The subscripts "d" and "u" in Eq. 1 stand for "Discriminative" and "Unsupervised". The reason behind such naming will become clear shortly. Below we detail out the form and properties of $\mathcal{L}_d$ and $\mathcal{L}_u$.

2.1. Discriminative Loss

The purpose of having $\mathcal{L}_d$ in Eq. 1 is to equip the latent space $\mathcal{H}$ with a metric to 1. minimize dissimilarities between samples coming from the same class and 2. maximizing the dissimilarities between samples from different classes.

Let $Z = \{z_j\}_{j=1}^{n_t}$ be the set of labeled samples in $\mathcal{H}$. In unsupervised domain adaptation $z_j = W_s^T x_j^s$ and $n = n_s$.

In the case of semi-supervised domain adaptation, $Z = \{W_s^T x_j^s\}_{j=1}^{n_s} \cup \{W_t^T x_j^t\}_{j=1}^{n_t}$

where we assume $n_t$ labeled target samples are provided (out of available $n_t$ samples). From the labeled samples in $\mathcal{H}$, we create $N_p$ pairs in the form $(z_1,k, z_2,k)$, $k = 1, 2, \cdots, N_p$ along their associated label $y_k \in \{-1, 1\}$. Here, $y_k = 1$ iff label of $z_1,k$ is similar to that of $z_2,k$ and $-1$ otherwise. That is the pair $(z_1,k, z_2,k)$ is similar if $y_k = 1$ and dissimilar otherwise.

To learn the metric $M$, we deem the distances between the similar pairs to be small while simultaneously making the distances between the dissimilar pairs large. In particular, we define $\mathcal{L}_d$ as,

$$\mathcal{L}_d = \frac{1}{N_p} \sum_{k=1}^{N_p} \ell_\beta(M, y_k, z_1,k - z_2,k, 1) + r(M),$$  

with

$$\ell_\beta(M, y, x, u) = \frac{1}{\beta} \log \left( 1 + \exp \left( \beta y (x^T M x - u) \right) \right).$$  

Here, $\ell_\beta$ is the generalized logistic function tailored with large margin structure (see Fig. 2) having a margin of $u$.

First note that the quadratic term in Eq. 3 (i.e., $x^T M x$) measures the Mahalanobis distance between $z_1,k$ and $z_2,k$ if used according to Eq. 2. Also note that $\ell_\beta(\cdot, \cdot, x, \cdot) = \ell_\beta(\cdot, -x, \cdot)$, hence how samples are order in the pairs is not important.

To better understand the behavior of the function $\ell_\beta$, assume the function is fed with a similar pair, i.e. $y_k = 1$. For

3For now we keep the margin at $u = 1$ and later will use this to explain the soft-margin extension.
the sake of discussion, also assume \( \beta = 1 \). In this case, \( \ell_\beta \) is decreased if the distance between \( z_{1,k} \) and \( z_{2,k} \) is reduced. For a dissimilar pair (i.e., \( y_k = -1 \)), the opposite should happen to have a smaller objective. That is, the Mahalanobis distance between the samples of a pair should be increased.

The function \( \ell_\beta(\cdot, \cdot, x, \cdot) \) can be understood as a smooth and differentiable form of the hinge-loss function. In fact, \( \ell_\beta(\cdot, \cdot, x, \cdot) \) asymptotically reaches the hinge-loss function if \( \beta \to \infty \). The smooth behavior of \( \ell_\beta(\cdot, \cdot, x, \cdot) \) is not only welcomed in the optimization scheme but also avoids samples in the latent space to collapse into a single point.

Along the general practice in metric learning, we regularize the metric \( M \) by \( r(M) \). The divergences derived from the \( \log \det(\cdot) \) function are familiar faces for regularizing Mahalanobis metrics in the literature [13, 45].

Among possible choices, we make use of the Stein divergence [11] in this work. Hence,

\[
r(M) = \frac{1}{p} \delta_s(M, I_p). \tag{4}
\]

Where,

\[
\delta_s(P, Q) = \log \det \left( \frac{P + Q}{2} \right) - \frac{1}{2} \log \det (PQ), \tag{5}
\]

for \( P, Q \in S_{++} \). Our motivation in using the Stein divergence stems from its unique properties. Among others, Stein divergence is symmetric, invariant to affine transformation and closely related to geodesic distances on the SPD manifold [11, 24, 9].

**Soft Margin Extension**

For large values of \( \beta \), the cost in Eq. 2 seeks the distances of similar pairs to be less than 1 while simultaneously it deems the distances of dissimilar pairs to exceed 1. This hard-margin in the design of \( \ell_\beta(\cdot, \cdot, x, \cdot) \) is not desirable. For example, with a large number of pairs, it is often the case to have outliers. Forcing outliers to fit into the hard margins can result in overfitting. As such, we propose a soft-margin extension of Eq. 3. The soft-margins are implemented by associating a non-negative slack variable \( \epsilon_k \) to a pair according to

\[
L_d = \frac{1}{N_p} \sum_{k=1}^{N_p} \ell_\beta(M, y_k, z_{1,k} - z_{2,k}, 1 + y_k \epsilon_k) + r(M) + \frac{1}{N_p} \sqrt{\sum \epsilon_k}, \tag{6}
\]

where a regularizer on the slack variables is also envisaged.

**2.2. Matching Statistical Properties**

A form of incompatibility between domains is due to their statistical discrepancies. Matching the first order
statistics of two domains for the purpose of adaptation is studied in [40, 2, 29]. In our framework, matching domain averages can be achieved readily. In particular, let \( \bar{x}_s^t = x_s^t - m_s \) and \( \bar{x}_t^t = x_t^t - m_t \) be the centered source and target samples with \( m_s \) and \( m_t \) being the mean of corresponding domains. It follows easily that the domain means in the latent space are zero and hence matching is achieved.

To go beyond first order statistics, we propose to match the second order statistics (i.e., covariance matrices) as well. The covariance of a domain reflects the relationships between its features. Hence, matching covariances of source and target domains in effect improves the cross feature relationships. We capture the mismatch between source and target covariances in the latent space using the Stein divergence to measure their differences. This leads us to define \( \mathcal{L}_u \) as

\[
\mathcal{L}_u = \frac{1}{p} \delta_1(\Sigma_s^TW_s^TW_s, W_t^T\Sigma_tW_t), \tag{7}
\]

with \( \Sigma_s^t \in S_n^{++} \) and \( \Sigma_t \in S_+^{2t} \) being the covariance matrices of the source and target domains, respectively. We emphasize that matching the statistical properties as discussed above is an unsupervised technique, enabling us to address unsupervised DA.

### 2.3. Classification Protocol

Upon learning \( W_s, W_t, M \), training samples from the source and target (if available) domains are mapped to the latent space using \( W_sM^2 \) and \( W_tM^2 \), respectively. For a query from the target domain \( x_t^q \), \( M^2W_t^T x_t^q \) is its latent space representation which is subsequently classified by a nearest neighbor classifier.

### 3. Optimization

The objective of our algorithm is to learn the transformation parameters \( (W_s, W_t) \), the metric \( M \) and slack variables \( \epsilon_1, \epsilon_2, ..., \epsilon_N_p \) (see Eq. 6 and Eq. 7). Inline with the general practice of dimensionality reduction, we propose to have orthogonality constraints on \( W_s \) and \( W_t \). That is \( W_s^TW_s = W_t^TW_t = I_p \). We include an experiment elaborating how orthogonality constraint improves the discriminatory power of the proposed framework in the supplementary material.

The problem depicted in Eq. 1 is indeed a non-convex and constrained optimization problem. One may resort to the method of Projected Gradient Descent (PGD) [7] to minimize Eq. 1. In PGD, optimization is proceed by projecting the gradient-descent updates onto the set of constraints. For example, in our case, we can first update \( W_s \) by ignoring the orthogonality constraint on \( W_s \) and then project the result onto the set of orthogonal matrices using eigen-decomposition. As such, optimization can be performed by alternatingly updating \( W_s, W_t \), the metric \( M \) and slack variables using PGD.

In PGD to perform the projection, the set of constraints needs to be closed though in practice one can resort to open sets. For example, the set of SPD matrices is open though one can project a symmetric matrix onto this set using eigen-decomposition.

Empirically, PGD showed an erratic and numerically unstable behavior in addressing our problem. This can be attributed to the non-linear nature of Eq. 1, existence of open-set constraints in the problem or perhaps the combination of both. To alleviate the aforementioned difficulty, we propose a more principle approach to minimize Eq. 1 by making use of the Riemannian optimization technique. We take a short detour and briefly describe the Riemannian optimization methods below.

### Optimization on Riemannian manifolds.

Consider a non-convex constrained problem in the form

\[
\begin{align*}
\text{minimize } & f(x) \\
\text{s.t. } & x \in \mathcal{M},
\end{align*}
\tag{8}
\]

where \( \mathcal{M} \) is a Riemannian manifold, i.e., informally, a smooth surface that locally resembles a Euclidean space. Optimization techniques on Riemannian manifolds (e.g., Conjugate Gradient) start with an initial solution \( x^{(0)} \in \mathcal{M} \), and iteratively improve the solution by following the geodesic identified by the gradient. For example, in the case of Riemannian Gradient Descent Method (RGDM), the updating rule reads

\[
x^{(t+1)} = \tau_{x^{(t)}}(-\alpha \text{ grad } f(x^{(t)})), \tag{9}
\]

with \( \alpha > 0 \) being the algorithm’s step size. Here, \( \tau_x(\cdot) : T_{x}\mathcal{M} \to \mathcal{M} \), is called the retraction and moves the solution along the descent direction while assuring that the new solution is on the manifold \( \mathcal{M} \), i.e., it is within the constraint set. \( T_x\mathcal{M} \) is the tangent space of \( \mathcal{M} \) at \( x \) and can be thought of as a vector space with its vectors being the gradients of all functions defined on \( \mathcal{M} \).

\[\text{strictly speaking and in contrast with the exponential map, a retraction only guarantees to pull a tangent vector on the geodesic locally, i.e., close to the origin of the tangent space. Reactions, however, are typically easier to compute than the exponential map and have proven effective in Riemannian optimization [1].}\]
Due to space limitation, we defer more details on Riemannian optimization techniques to the supplementary. As for now, it suffices to say that to perform optimization on the Riemannian manifolds, the form of Riemannian gradient, retraction and the gradient of the objective with respect to its parameters (shown by $\nabla$) are required. The constraints in Eq. 1 are orthogonality (transformations $W_s$ and $W_t$) and p.d. for metric $M$. The geometry of these constraints are captured by the Stiefel [30, 23] and SPD [24, 10] manifolds, formally defined as

**Definition 1 (The Stiefel Manifold)** The set of $(n \times p)$-dimensional matrices, $p \leq n$, with orthonormal columns endowed with the Frobenius inner product\(^7\) forms a compact Riemannian manifold called the Stiefel manifold $\text{St}(p, n) [1]$.  

$$\text{St}(p, n) \triangleq \{ W \in \mathbb{R}^{n \times p} : W^TW = I_p \}.$$  

**Definition 2 (The SPD Manifold)** The set of $(p \times p)$-dimensional real, SPD matrices endowed with the Affine Invariant Riemannian Metric (AIRM) [42] forms the SPD manifold $S^p_{++}$.

$$S^p_{++} \triangleq \{ M \in \mathbb{R}^{p \times p} : v^TMv > 0, \forall v \in \mathbb{R}^p - \{0_p\} \}. \quad (11)$$

Updaing $W_s$, $W_t$ and $M$ and slacks can be done alternatively using Riemannian optimization. As mentioned above, the ingredients for doing so are 1. the Riemannian tools for the Stiefel and SPD manifolds along 2. the form of gradients of the objective with respect to its parameters. To do complete justice, in Table. 1 we provide the Riemannian metric, form of Riemannian gradient and retraction for the Stiefel and SPD manifolds. In Table. 2, the gradient of Eq. 1 with respect to $W_s$, $W_t$ and $M$ and slacks is provided. The detail of derivations can be found in the supplementary material. A tiny note about the slacks worth mentioning. To preserve the non-negativity constraint on $\epsilon_k$, we define $\epsilon_k = e^v_k$ and optimize on $v_k$ instead. This in turn makes optimization for the slacks unconstrained.

**Remark 1** From a geometrical point of view, we can make use of the product topology of the parameter space to avoid alternative optimization. More specifically, the set

$$\mathcal{M}_{\text{prod.}} = \text{St}(p, s) \times \text{St}(p, t) \times S^p_{++} \times \mathbb{R}^{N_p}, \quad (12)$$

can be given the structure of a Riemannian manifold using the concept of product topology [1].

**Remark 2** In Fig. 5, we compare the convergence behavior of PGD, alternating Riemannian optimization and optimization using the product geometry. While optimization on

\(^7\)Note that the literature is divided between this choice and another form of Riemannian metric. See [15] for details.
We compare our perfor-
mean discrepancy (MMD) \[HeMap \[transfer method jointly . T wo representative works are the
parameters and slack variables. Without less of generality we only
perform the training. The manifold alignment algorithm
experiments. To this end, we propose to set \( \beta \) to the reciprocal of the standard deviation of the similar pair distances.
5.2. Semi-supervised Setting
In our semi-supervised experiments, we follow the standard setup on the Office+Caltech10 dataset with the train/test splits provided by [28]. The Office+Caltech10 dataset contains images collected from 4 different sources and 10 object classes. The corresponding domains are Amazon, Webcam, DSLR, and Caltech. We use a subspace of dimension 20 for DA-SL algorithms. We employ SURF [3] for the handcrafted feature experiments. We extract VGG-Net features with the network model of [48] for the deep-net feature experiments\(^8\). We compare our performance with the following benchmarks:
\textbf{1-NN-t and SVM-t} : Basic Nearest Neighbor (1-NN) and linear SVM classifiers trained only on the target domain.
\textbf{HFA [14]} : This method employs latent space learning based on the max-margin framework. As in its original implementation, we use the RBF kernel SVM for its evaluation.
\textbf{MMDT [27]} : This method jointly learns a transformation between the source and target domains along a linear SVM for classification.
\textbf{CDLS [29]} : This is the cross-domain landmark search algorithm. We use the parameter setting (\( \delta = 0.5 \) in the notation of [29]) recommended by the authors.
\(^8\)The same SURF and VGG-FC6 features are used for the unsupervised experiments as well.

5.1. Implementation Details
Since the number of dissimilar pairs is naturally larger than the number of similar pairs, we randomly sample from the different pairs to keep the sizes of these two sets equal. We initialize the projection matrices \( W_s \), \( W_t \) with PCA, following the transductive protocol [21, 16, 27, 29]. For the semi-supervised setting, we initialize \( M \) with the Mahalanobis metric learned on the similar pair covariances [31], and for the unsupervised setting, we initialize it with the identity matrix. For all our experiments we have \( \lambda = 1 \). We include an experiment showing our solution’s robustness to \( \lambda \) in the supplementary material. We use the toolbox provided by [6] for our implementations.

\textbf{Remark 4} To have a simple way of determining \( \beta \) in Eq. 3, we propose a heuristic which is shown to be effective in our experiments. To this end, we propose to set \( \beta \) to the reciprocal of the standard deviation of the similar pair distances.

5.2. Semi-supervised Setting
In our semi-supervised experiments, we follow the standard setup on the Office+Caltech10 dataset with the train/test splits provided by [28]. The Office+Caltech10 dataset contains images collected from 4 different sources and 10 object classes. The corresponding domains are Amazon, Webcam, DSLR, and Caltech. We use a subspace of dimension 20 for DA-SL algorithms. We employ SURF [3] for the handcrafted feature experiments. We extract VGG-Net features with the network model of [48] for the deep-net feature experiments\(^8\). We compare our performance with the following benchmarks:

\textbf{1-NN-t and SVM-t} : Basic Nearest Neighbor (1-NN) and linear SVM classifiers trained only on the target domain.

\textbf{HFA [14]} : This method employs latent space learning based on the max-margin framework. As in its original implementation, we use the RBF kernel SVM for its evaluation.

\textbf{MMDT [27]} : This method jointly learns a transformation between the source and target domains along a linear SVM for classification.

\textbf{CDLS [29]} : This is the cross-domain landmark search algorithm. We use the parameter setting (\( \delta = 0.5 \) in the notation of [29]) recommended by the authors.

\(^8\)The same SURF and VGG-FC6 features are used for the unsupervised experiments as well.

5.1. Implementation Details
Since the number of dissimilar pairs is naturally larger than the number of similar pairs, we randomly sample from the different pairs to keep the sizes of these two sets equal. We initialize the projection matrices \( W_s \), \( W_t \) with PCA, following the transductive protocol [21, 16, 27, 29]. For the semi-supervised setting, we initialize \( M \) with the Mahalanobis metric learned on the similar pair covariances [31], and for the unsupervised setting, we initialize it with the identity matrix. For all our experiments we have \( \lambda = 1 \). We include an experiment showing our solution’s robustness to \( \lambda \) in the supplementary material. We use the toolbox provided by [6] for our implementations.

\textbf{Remark 4} To have a simple way of determining \( \beta \) in Eq. 3, we propose a heuristic which is shown to be effective in our experiments. To this end, we propose to set \( \beta \) to the reciprocal of the standard deviation of the similar pair distances.

5.2. Semi-supervised Setting
In our semi-supervised experiments, we follow the standard setup on the Office+Caltech10 dataset with the train/test splits provided by [28]. The Office+Caltech10 dataset contains images collected from 4 different sources and 10 object classes. The corresponding domains are Amazon, Webcam, DSLR, and Caltech. We use a subspace of dimension 20 for DA-SL algorithms. We employ SURF [3] for the handcrafted feature experiments. We extract VGG-Net features with the network model of [48] for the deep-net feature experiments\(^8\). We compare our performance with the following benchmarks:

\textbf{1-NN-t and SVM-t} : Basic Nearest Neighbor (1-NN) and linear SVM classifiers trained only on the target domain.

\textbf{HFA [14]} : This method employs latent space learning based on the max-margin framework. As in its original implementation, we use the RBF kernel SVM for its evaluation.

\textbf{MMDT [27]} : This method jointly learns a transformation between the source and target domains along a linear SVM for classification.

\textbf{CDLS [29]} : This is the cross-domain landmark search algorithm. We use the parameter setting (\( \delta = 0.5 \) in the notation of [29]) recommended by the authors.

\(^8\)The same SURF and VGG-FC6 features are used for the unsupervised experiments as well.
Table 3 and Table 4 report the performances using the handcrafted SURF and the VGG-FC6 layer features, respectively. For the SURF features our solution achieves the best performance in 7 out 12 cases, and for the VGG-FC6 features, our solution tops in 9 sets. We notice the 1-NN-t baseline performs the worst for both SURF and the VGG-FC6 features. Hence, it is clear that the used features do not favor the nearest neighbor classifier. We observe that Caltech and Amazon domains contain the largest number of test instances. Although the performances of all tested methods decrease on these domains, particularly on Caltech, our method achieves the top rank in almost all domain transformations.

5.3. Unsupervised Setting

In the unsupervised domain adaptation problem, only labeled data from the source domain is available [16, 21]. We perform two sets of experiments for this setting. (1) We evaluate the object recognition performance on the Office+Caltech10 dataset. Similar to the semi-supervised settings, we use the SURF and VGG-FC6 features. Our results demonstrate that the learned transformations by our method are superior domain representations. (2) We analyze our performance when the domain discrepancy is gradually increased. This experiment is performed on the PIEFace dataset. We compare our method with the following benchmarks:

1-NN-s and SVM-s: Basic 1-NN and linear SVM classifiers trained only on the source domain.

GFK-PLS [21]: The geodesic flow kernel algorithm where partial least squares (PLS) implementation is used to initialize the source subspace. Results are evaluated on kernel-NNs.

SA [16]: This is the subspace alignment algorithm. Results are evaluated using 1-NN.

CORAL [49]: The correlation alignment algorithm that uses a linear SVM on the similarity matrix formed by correlation matching.

5.3.1 Office+Caltech10 (Unsupervised)

We follow the original protocol provided by [21] on Office+Caltech10 dataset. Note that several baselines, determine the best dimensionality per domain to achieve their maximum accuracies on SURF features. We observed that a dimensionality in the range [20,120] provides consistent results for our solution using SURF features. For VGG features we empirically found the dimensionality of 20 suits best for the compared DA-SL algorithms.

Table. 5 and Table. 6 present the unsupervised setting results using the SURF and VGG-FC6 features. For both feature types, our solution yields the best performance in 8 domain transformations out of 12.

5.3.2 PIE-Multiview Faces

The PIE Multiview dataset includes face images of 67 individuals captured from different views, illumination conditions, and expressions. In this experiment, we use the views C27 (looking forward) as the source domain and C09 (looking down), and the views C05, C37, C02, C25 (looking towards left in an increasing angle, see Fig. 5) as target domains. We expect the face inclination angle to reflect the complexity of transfer learning. We normalize the images to 32×32 pixels and use the vectorized gray-scale images as features. Empirically, we observed that the GFK [21] and SA [16] reach better performances if the features are normalized to have unit ℓ2 norm. We therefore use ℓ2 normalized features in our evaluations. The dimensionality of the subspaces for all the subspace based methods (i.e., [21, 16]) including ours is 100.

Table. 7 lists the classification accuracies with increasing angle of inclination. Our solution attains best scores for 4 views and the second best for the C09. With the increasing

---

Figure 4. The accuracy gain on Office-Caltech dataset for GFK [21] and SA [16] when their initial PCA subspaces are replaced with PLS and our Ws transformation matrices.

**Learned Transformations as Subspace Representations:**

We consider both GFK [21] and SA [16] as DA-SL algorithms. Both these methods make use of PCA subspaces to adapt the domains. Nevertheless, there is no strong reason to assume the PCA subspaces favorably capture the domain structure for transfer learning. Gong et al., [21] show that their performance improves when employing PLS to define the source subspace. However, this subspace learning is disjoint to their domain adaptation technique. We notice that, a more suitable initialization would be to use a subspace representation learned along with a domain adaptation framework. We empirically show this by using our learned source transformation matrix Ws as the source subspace initialization for [21] and [16].

Figure 4 compares the accuracy gains over PCA spaces by using PLS and our Ws initialization. It is clear that the highest classification accuracy gain is obtained by our Ws initialization. This proves that Ws is capable to learn a more favorable subspace representation for DA.
Table 3. Semi-supervised domain adaptation results using SURF features on Office+Caltech10 [21] dataset with the evaluation setup of [27]. The best score (in bold blue), the second best (in blue).

| Method    | A→W | A→D | A→C | A→A | W→A | W→D | W→C | D→A | D→W | D→C | C→A | C→W | C→D |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1-NN-t    | 81.0| 79.1| 67.8| 76.1| 77.9| 65.2| 77.1| 81.7| 65.6| 78.3| 80.2| 77.7|     |
| SVM-t     | 89.1| 88.2| 77.3| 86.5| 87.7| 76.3| 87.3| 88.3| 76.3| 87.5| 87.8| 84.9|     |
| HFA [14]  | 87.9| 87.1| 75.5| 85.1| 87.3| 74.4| 85.9| 86.9| 74.8| 86.2| 86.0| 87.0|     |
| MMDT [27] | 82.5| 77.1| 78.7| 84.7| 85.1| 73.6| 83.6| 86.1| 71.8| 85.9| 82.8| 77.9|     |
| CDLS [29] | 91.2| 86.9| 78.1| 87.4| 88.5| 78.2| 88.1| 90.7| 77.9| 88.0| 89.7| 86.3|     |
| ILS (1-NN)| 90.7| 87.7| 83.3| 88.8| 94.5| 82.8| 88.7| 95.5| 81.4| 89.7| 91.4| 86.9|     |

Table 4. Semi-supervised domain adaptation results using VGG-FC6 features on Office+Caltech10 [21] dataset with the evaluation setup of [27]. The best (in bold blue), the second best (in blue).

| Method    | A→W | A→D | A→C | A→A | W→A | W→D | W→C | D→A | D→W | D→C | C→A | C→W | C→D |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1-NN-t    | 81.0| 79.1| 67.8| 76.1| 77.9| 65.2| 77.1| 81.7| 65.6| 78.3| 80.2| 77.7|     |
| SVM-t     | 89.1| 88.2| 77.3| 86.5| 87.7| 76.3| 87.3| 88.3| 76.3| 87.5| 87.8| 84.9|     |
| HFA [14]  | 87.9| 87.1| 75.5| 85.1| 87.3| 74.4| 85.9| 86.9| 74.8| 86.2| 86.0| 87.0|     |
| MMDT [27] | 82.5| 77.1| 78.7| 84.7| 85.1| 73.6| 83.6| 86.1| 71.8| 85.9| 82.8| 77.9|     |
| CDLS [29] | 91.2| 86.9| 78.1| 87.4| 88.5| 78.2| 88.1| 90.7| 77.9| 88.0| 89.7| 86.3|     |
| ILS (1-NN)| 90.7| 87.7| 83.3| 88.8| 94.5| 82.8| 88.7| 95.5| 81.4| 89.7| 91.4| 86.9|     |

Table 5. Unsupervised domain adaptation results using SURF features on Office+Caltech10 [21] dataset with the evaluation setup of [21].The best (in bold blue), the second best (in blue).

| Method    | A→W | A→D | A→C | A→A | W→A | W→D | W→C | D→A | D→W | D→C | C→A | C→W | C→D |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1-NN-s    | 23.1| 22.3| 20.0| 14.7| 31.3| 12.0| 23.0| 51.7| 19.9| 21.0| 19.0| 23.6|     |
| SVM-s     | 25.6| 33.4| 35.9| 30.4| 67.7| 23.4| 34.6| 70.2| 31.2| 43.8| 30.5| 40.3|     |
| GFK-PLS [21]| 35.7| 35.1| 37.9| 35.5| 71.2| 29.3| 36.2| 79.1| 32.7| 40.4| 35.8| 41.1|     |
| SA [10]   | 38.6| 37.6| 35.9| 34.3| 80.3| 32.5| 38.0| 83.6| 32.4| 39.0| 36.8| 59.6|     |
| CORAL [49]| 88.7| 38.3| 40.3| 37.8| 84.9| 34.8| 38.1| 85.8| 34.2| 47.2| 39.2| 40.7|     |
| ILS (1-NN)| 40.6| 41.0| 37.1| 38.6| 72.4| 32.6| 38.9| 79.1| 36.9| 48.6| 42.0| 44.1|     |

Table 6. Unsupervised domain adaptation results using VGG-FC6 features on Office+Caltech10 [21] dataset with the evaluation setup of [21].The best (in bold blue), the second best (in blue).

| Method    | A→W | A→D | A→C | A→A | W→A | W→D | W→C | D→A | D→W | D→C | C→A | C→W | C→D |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1-NN-s    | 60.9| 52.3| 70.1| 62.4| 83.9| 57.5| 57.0| 86.7| 48.0| 81.9| 65.9| 55.6|     |
| SVM-s     | 63.1| 51.7| 74.2| 69.8| 89.4| 64.7| 58.7| 91.8| 55.5| 86.7| 74.8| 61.5|     |
| GFK-PLS [21]| 74.1| 63.5| 77.7| 77.9| 92.9| 71.3| 69.9| 92.4| 64.0| 86.2| 76.5| 66.5|     |
| SA [10]   | 76.0| 64.9| 77.6| 76.6| 90.4| 70.7| 69.0| 90.5| 62.3| 83.9| 76.0| 66.2|     |
| CORAL [49]| 74.8| 67.1| 79.8| 81.2| 92.6| 75.2| 75.8| 94.6| 64.7| 89.4| 77.6| 67.6|     |
| ILS (1-NN)| 82.4| 72.5| 78.9| 85.9| 87.4| 77.0| 79.2| 94.2| 66.5| 87.6| 84.4| 73.0|     |

Conclusion

In this paper, we proposed a solution for both semi-supervised and unsupervised Domain Adaptation (DA) problems. Our solution learns a latent space in which domain discrepancies are minimized. We showed that such a latent space can be obtained by 1. minimizing a notion of discriminative power over the available labeled data while simultaneously 2. matching statistical properties across the domains. To determine the latent space, we modeled the learning problem as a minimization problem on Riemannian manifolds and solved it using optimization techniques on matrix manifolds.

Empirically, we showed that the proposed method outperformed state-of-the-art DA solutions in semi-supervised and unsupervised settings. With the proposed framework we see possibilities of extending our solution to large scale datasets with stochastic optimization techniques, multiple source DA and for domain generalization [20, 18]. In terms of algorithmic extensions we look forward to use dictionary learning [32] and higher order statistics matching.
References


[31] M. Koestinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof. Large scale metric learning from equivalence.


