A Manifold Approach to Learning Mutually Orthogonal Subspaces

Stephen Giguere   SGIGUERE@CS.UMASS.EDU
Francisco Garcia   FMGARCIA@CS.UMASS.EDU
College of Information and Computer Sciences, University of Massachusetts, Amherst, Massachusetts 01003

Sridhar Mahadevan   SRIDHAR.MAHADEVAN@SRI.COM
Stanford Research Institute, Menlo Park, California 94025

Abstract

Although many machine learning algorithms involve learning subspaces with particular characteristics, optimizing a parameter matrix that is constrained to represent a subspace can be challenging. One solution is to use *Riemannian optimization methods* that enforce such constraints implicitly, leveraging the fact that the feasible parameter values form a manifold. While Riemannian methods exist for some specific problems, such as learning a single subspace, there are more general subspace constraints that offer additional flexibility when setting up an optimization problem but have not been formulated as a manifold.

We propose the *partitioned subspace (PS) manifold* for optimizing matrices that are constrained to represent one or more subspaces. Each point on the manifold defines a partitioning of the input space into mutually orthogonal subspaces, where the number of partitions and their sizes are defined by the user. As a result, distinct groups of features can be learned by defining different objective functions for each partition. We illustrate the properties of the manifold through experiments on multiple dataset analysis and domain adaptation.

1. Introduction

The process of designing a model and learning its parameters by numerically optimizing a loss function is a cornerstone of machine learning. In this setting, it is common to place constraints on the model's parameters to ensure that they are valid, to promote desirable characteristics such as sparsity, to incorporate domain knowledge, or to make learning more efficient. Because they can significantly impact the difficulty of an optimization problem, it is useful to understand the properties of particular types of constraints and to develop optimization techniques that preserve them.

Recently, there has been interest in developing *Riemannian optimization* methods that enforce certain constraints by leveraging the geometry of the parameters that satisfy them, called the *feasible set*. Specifically, if the feasible set forms a smooth manifold in the original parameter space, then Riemannian optimization can be applied. Unlike other strategies for enforcing constraints, such as augmenting the loss function with penalty terms or projecting the parameters back onto the feasible set, this approach eliminates the need to deal with constraints explicitly by performing optimization on the constraint manifold directly. In many cases, using Riemannian optimization can simplify algorithms, provide convergence guarantees on learning, and ensure that constraints are satisfied exactly rather than approximately.

In this work, we investigate Riemannian optimization methods for enforcing *subspace constraints*, which we define as any constraint that forces a matrix of parameters to represent one or more subspaces. There are two commonly used subspace constraints that have well-established Riemannian optimization methods provided in (Edelman, Arias, and Smith, 1998). The first is applicable when optimizing over matrices that define k-dimensional bases in an n-dimensional ambient space. In this case, the feasible set consists of all $n \times k$ matrices Y that satisfy $Y^T Y = I$, which corresponds to the *Stiefel manifold*. If the optimization is instead taken over all distinct k-dimensional subspaces, then in addition to the constraint that $Y^T Y = I$, two matrices must be considered identical if they have the same span. This second condition is important because it implies that during optimization, estimates must be updated to not only change the parameter matrix, but to do so in a way that changes the subspace it represents. The feasible set for this constraint corresponds to the *Grassmannian manifold*, which has proven useful for a large number of applications including background separation in video, human activity analysis, subspace tracking, and others (He, Balzano, and Lui, 2011; Turaga and Chellappa, 2009; He, Balzano, and Szlam, 2012).
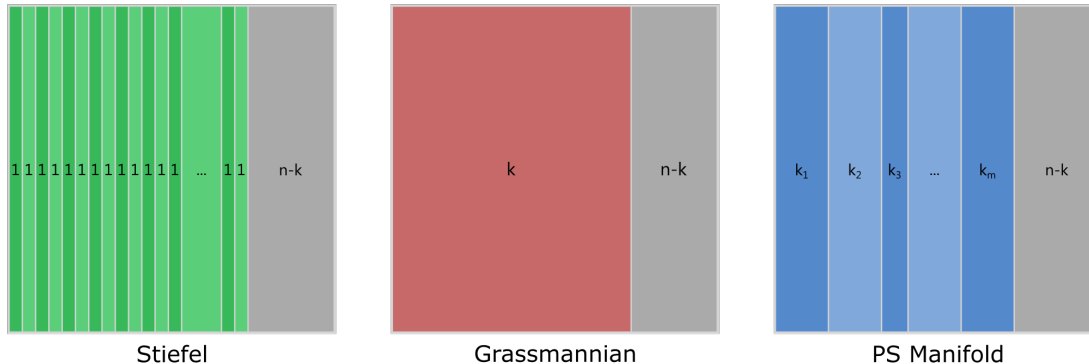
Figure 1. Visualization of Q matrices showing the subspaces encoded by the Stiefel, Grassmannian, and partitioned subspace manifolds. Within each matrix, the colored segments represent mutually orthogonal subspaces. The values in each segment denote the dimensionality of the subspace. All three matrices are the same size and have k columns of interest, but the properties of their respective manifolds cause the subspaces to be partitioned differently. While the Stiefel manifold represents k, 1-dimensional subspaces and the Grassmannian represents a single, k-dimensional subspace, the PS manifold allows the number of partitions and their sizes to be chosen as needed for a given problem.

While the constraints satisfied by optimizing on the Stiefel or Grassmannian manifolds are useful in practice, there are more general subspace constraints that are not captured by these manifolds and offer significant flexibility when setting up an optimization problem. To see this, first consider that a point on a Stiefel manifold can be viewed as a collection of k, 1-dimensional subspaces that are constrained to be mutually orthogonal. Similarly, a point on a Grassmannian manifold represents 1, k-dimensional subspace. It is therefore natural to consider a general constraint in which both the number of subspaces and their sizes can be specified by the user. These relationships are shown in figure 1. Importantly, subspace constraints of this form allow different set of features to be learned according to different criteria. In analogy to optimization on the Stiefel and Grassmannian manifolds, we aim to optimize over distinct sets of m mutually orthogonal subspaces with sizes $[k_1, k_2, ..., k_m]$, which we refer to as *partitions*.

In this paper, we introduce a novel manifold that generalizes the Stiefel and Grassmannian manifolds to implicitly enforce the general subspace constraints described above. Individual points on our proposed manifold define partitions of n-dimensional space into mutually orthogonal subspaces of particular sizes. In addition, we derive update rules for performing Riemannian optimization on the manifold. This allows features of the original space to be grouped in useful ways by defining separate objective functions on each subspace. For example, given two datasets X_0 and X_1, the linear features that best describe the two can easily be partitioned into a pair of subspaces containing features unique to each dataset and a subspace of features that are shared between them. Because of these characteristics, we refer to this manifold as the *partitioned subspace*

(PS) manifold. Finally, we provide several examples using both real and synthetic data to illustrate how the manifold can be applied in practice, and to establish intuition for its properties.

Related work The problem of learning parameters subject to subspace constraints has been investigated widely and for a variety of applications. In general, these applications use only simple subspace constraints, which effectively cause the feasible set to be either the Grassmannian manifold or Stiefel manifold. Optimization subject to general subspace constraints has been investigated much less. Kim, Kittler, and Cipolla (2010) proposed an algorithm for face recognition that is similar to our work in that they also learn a set of mutually-orthogonal subspaces. However, their approach learns subspaces by incrementally updating and reorthogonalizing the subspace matrix, whereas we optimize on the constraint manifold itself using Riemannian optimization. Our work presents, to our knowledge, the first formulation of general subspace constraints as a manifold.

The rest of this paper is organized as follows. In section 2, we describe background concepts and establish notation. Next, we define the partitioned subspace manifold and provide details necessary to use it for optimization in section 3. Finally, section 4 illustrates the various properties of the manifold and analyzes them through experiments on both real and synthetic data.

2. Background

Before defining the partitioned subspace manifold, we briefly introduce some necessary background concepts from differential geometry.

2.1. Riemannian Manifolds

A *Riemannian manifold*, or simply a manifold, can be described as a continuous set of points that appears locally Euclidean at every location. More specifically, a manifold is a topological space and a collection of differentiable, one-to-one mappings called *charts*. At any point on a d-dimensional manifold, there is a chart that maps a neighborhood containing that point to the Euclidean space $\mathbb{R}^d$ (Absil, Mahony, and Sepulchre, 2009). This property allows a *tangent space* to be defined at every point, which is an Euclidean space consisting of directions that point along the manifold. Tangent spaces are particularly important for optimization because they characterize the set of update directions that can be followed without leaving the manifold. While the most familiar manifold is Euclidean space, a large number of other, more exotic manifolds can be defined. The manifolds discussed in this paper are examples of *matrix manifolds*, meaning that they consist of points that can be represented as matrices.

Given a manifold, it is often useful to define a *quotient manifold* by specifying an equivalence relation that associates sets of points on the original manifold with individual points on the quotient (Edelman, Arias, and Smith, 1998). Because each point on the quotient is an equivalence class of points on the parent manifold, we use the notation $[p]$ to refer to a point on the quotient that contains p, where p is an element of the parent manifold. Quotient manifolds inherit a great deal of structure from their parent manifold. In particular, constraints that define the parent manifold also apply to the quotient.

To illustrate this, consider the Stiefel and Grassmannian manifolds, which can be defined as quotients of the group of orthogonal matrices. Later in section 3, the derivation for the partitioned subspace manifold will follow a similar pattern. The orthogonal group, denoted $\mathcal{O}_n$, consists of all $n \times n$ matrices Q that satisfy the condition $Q^T Q = I$. Note that in this work, we use Q to refer to an element of this set. From this definition, we see that each point $[Q]_{St}$ on a Stiefel manifold is a set of matrices from the orthogonal group whose first k columns match those of Q. Similarly, each point $[Q]_{Gr}$ on a Grassmannian manifold corresponds to those orthogonal group matrices whose first k columns span the same subspace. These relationships can be expressed mathematically as follows:

$$\text{Stiefel points } [Q]_{St} = \{QE\}_{E \in \mathcal{E}_{St}}$$
$$\text{Grassmannian points } [Q]_{Gr} = \{QE\}_{E \in \mathcal{E}_{Gr}}$$

where

$$\mathcal{E}_{St} = \begin{bmatrix} I_k & \\ & \mathcal{O}_{n-k} \end{bmatrix}, \text{ and}$$
$$\mathcal{E}_{Gr} = \begin{bmatrix} \mathcal{O}_k & \\ & \mathcal{O}_{n-k} \end{bmatrix}$$

Because they encode the relationship between matrices that map to the same point on the quotient, we refer to the sets $\mathcal{E}_{St}$ and $\mathcal{E}_{Gr}$ as the *equivalence sets* for their respective manifolds.

Representative elements As defined above, points on the Stiefel and Grassmannian manifolds are equivalence classes $[Q]$ of orthogonal matrices $Q \in \mathcal{O}_n$. However, in practice, these points are represented by single, $n \times k$ matrices Y. This discrepancy occurs because in general, only the subspace spanned by the first k columns is of interest. Given a point $[Q]$ on these manifolds, we refer to Y as a *representative element* defined by:

$$Y = Q' \begin{bmatrix} I_k \\ 0 \end{bmatrix}, \ s.t. \ Q' \in [Q]$$

2.2. Riemannian Optimization

Optimization on manifolds requires specialized *Riemannian optimization* methods to ensure that the learned parameters remain valid. While many standard optimization methods have been derived on manifolds (Edelman, Arias, and Smith, 1998), we focus on Riemmannian gradient descent because of its simplicity. As in standard gradient descent, the goal is to use the gradient of a differentiable convex function f with respect to the matrix Q, denoted G_Q, to construct a sequence of estimates $Q' = Q - \alpha G_Q$ that converges to the minimum of f, where the parameter α controls the size of the update at each step. However, in Riemannian gradient descent, the estimates are constrained to lie on a manifold.

Riemannian gradient descent proceeds by iterating three steps, as illustrated in Figure 2. The first step is to compute the gradient at the current location, which is given by $G_Q = \nabla_Q f$.

Next, the update direction is modified to reflect the geometry of the underlying manifold. Intuitively, if some component of the update direction is orthogonal to the manifold's surface, then any movement at all along that direction will immediately leave the manifold. To correct for this, the update direction is projected onto the tangent space at the current location. We denote the projected update direction as $\Delta = \pi_{T_Q M}(G_Q)$, where $\pi_{T_Q M}(\cdot)$ projects its input onto the tangent space of manifold M at Q.

The final step of Riemmannian gradient descent computes a new location by moving along the manifold in the direction Δ. This is accomplished using the *exponential map*, which maps elements of a tangent space onto the manifold by moving in a straight line along its surface. For matrix manifolds, the exponential map is given by the matrix exponential function $\exp(\cdot)$, and the updated location is given by the following formula, as shown in (Edelman, Arias, and
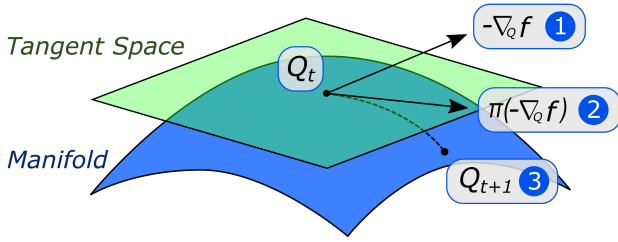
Figure 2. The three steps for Riemannian gradient descent. First, the negative gradient direction is computed. Next, that direction is projected onto the tangent space at the current estimate. Finally, a new estimate is computed using an appropriate retraction. These steps are repeated until the estimates converge.

Smith, 1998):

$$Q' = Q \exp(\alpha Q^T \Delta) \tag{1}$$

Unfortunately, computing updates in this way is inefficient due to the cost of evaluating the matrix exponential. Instead, it is common to use a *retraction*, which behaves similarly to the exponential map but may distort the magnitude of the update. Depending on the manifold, these alternatives may be comparably inexpensive. For example, updates on quotient manifolds of the orthogonal group can be performed using the *Q-factor retraction*, which maps an update Δ at the point Q to the location $Q' = \text{qr}(Q + \Delta)$, where $\text{qr}(\cdot)$ denotes the Q matrix of the QR decomposition of its input (Absil, Mahony, and Sepulchre, 2009).

The steps of computing the standard gradient, projecting onto the tangent space at the current point Q, and updating the location using an appropriate retraction are repeated until convergence.

3. The Partitioned Subspace Manifold

Having defined the necessary basic concepts, we now describe the partitioned subspace (PS) manifold, emphasizing the intuition behind its derivation. We also derive the components needed to apply Riemannian optimization on the manifold in section 3.2. Note that in this section, we use $[Q]$ to denote $[Q]_{PS}$ for notational convenience.

3.1. Formulation

The PS manifold is defined as a quotient of the orthogonal group, similarly to how the Stiefel and Grassmannian manifolds were defined in section 2.1. Here, the difference will be the use of a generalized equivalence relation. For the Stiefel manifold, the equivalence relation equates two orthogonal group matrices if they are related by a rotation of their last $n - k$ columns. The Grassmannian equivalence relation, on the other hand, equates two matrices if they are related by both a rotation of their last $n - k$ columns and a rotation of their first k columns. The PS manifold equivalence relation generalizes this pattern by equating two matrices if they are related by rotation of the first k_1 columns, or the next k_2 columns, etc. Intuitively, rotations that only involve basis vectors for a single partition will not change the subspace they span, whereas rotations that mix basis vectors from different partitions will. Based on this, we define the partitioned subspace manifold as follows.

Definition 1 *Let $k_1, k_2, ..., k_m$ be subspace partition sizes, where m is the number of partitions. Furthermore, let n be the dimensionality of the ambient space, and let $k = \sum_{i=1}^{m} k_i \leq n$ denote the total subspace size. We define the partitioned subspace manifold as a quotient of the group of orthogonal matrices, with points on the manifold characterized by:*

$$PS \text{ points } [Q]_{PS} = \{QE\}_{E \in \mathcal{E}_{PS}} \tag{2}$$

where

$$\mathcal{E}_{PS} = \begin{bmatrix} \mathcal{O}_{k_1} & & & \\ & \ddots & & \\ & & \mathcal{O}_{k_m} & \\ & & & \mathcal{O}_{n-k} \end{bmatrix} \tag{3}$$

Similarly to the Stiefel and Grassmannian manifolds, the last $n - k$ columns are generally ignored, and points on the PS manifold are represented by $n \times k$ matrices $Y = Q' \begin{bmatrix} I_k & 0 \end{bmatrix}^T$ for some $Q' \in [Q]_{PS}$. It is straightforward to show that the partitioned subspace manifold is equivalent to a Grassmannian manifold when $m = 1$ and $k_1 = k$, and equivalent to the Stiefel manifold when $m = k$ and $k_j = 1$ for $j = 1, 2, ..., m$.

Next, we characterize the tangent space for the PS manifold, which will be required for Riemannian optimization. Let Q be any element of an equivalence class defining a point on the PS manifold. Because the manifold is a quotient of $\mathcal{O}_n$, it is subject to the orthogonality constraint $Q^T Q = I$. As shown by Edelman, Arias, and Smith (1998), this implies that elements Δ of the tangent space at $[Q]$ satisfy the skew-symmetry condition, $Q^T \Delta = -\Delta^T Q$. It is straightforward to verify that $\Delta = QA$ satisfies this condition for all skew-symmetric A. However, this does not fully characterize the tangent space at $[Q]$ because there are directions of this form that lead only to points that are in the same equivalence class $[Q]$. These directions are exactly the ones for which movement in that direction is the same as multiplication by an element of $\mathcal{E}_{PS}$:

$$Q \exp(\alpha Q^T \Delta) = QE, \quad E \in \mathcal{E}_{PS}$$

Solving this equation reveals that updating Q in the direction Δ will not change its equivalence class on the PS man-

ifold if Δ is of the form:

$$\Delta \;=\; Q \begin{bmatrix} A_{k_1} & & & \\ & \ddots & & \\ & & A_{k_m} & \\ & & & A_{n-k} \end{bmatrix},$$

where each A_i is a $k_i \times k_i$ skew-symmetric matrix. Note that to reach this conclusion, we rely on the observation that $Q^T\Delta = A$, and that $\exp(A) \in \mathcal{O}$ because A is skew-symmetric (Edelman, Arias, and Smith, 1998). Given these observations, tangent spaces on the partitioned subspace manifold are characterized as follows. Let $B_{i,j}$ denote an arbitrary $k_i \times k_j$ matrix, and let $B_{\perp,j}$ and $B_{i,\perp}$ denote arbitary $(n-k) \times k_j$ and $k_i \times (n-k)$ matrices, respectively. The tangent space at a point $[Q]$ on the partitioned subspace manifold is the set of all $n \times n$ matrices Δ given by:

$$\Delta \;=\; Q \begin{bmatrix} 0 & -B_{2,1}^T & \cdots & -B_{m,1}^T & -B_{\perp,1}^T \\ B_{2,1} & 0 & \cdots & -B_{m,2}^T & -B_{\perp,2}^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ B_{m,1} & B_{m,2} & \cdots & 0 & -B_{\perp,m}^T \\ B_{\perp,1} & B_{\perp,2} & \cdots & B_{\perp,m} & 0 \end{bmatrix} \quad (4)$$

3.2. Optimization on the PS Manifold

To apply Riemannian optimization on the partitioned subspace manifold, we must derive the tangent space projection operator and identify an appropriate retraction.

The projection of an arbitrary update direction Z onto the tangent space at $[Q]$ can be derived as follows. First, we project Z onto the set of matrices of the form QA, where A is skew-symmetric. The result is given by $Q\ \text{skew}(Q^T Z)$, where $\text{skew}(X) = \frac{1}{2}(X - X^T)$ (Edelman, Arias, and Smith, 1998). Next, we replace the diagonal blocks of this matrix with zeros to obtain the projection of Z onto matrices of the form given in equation 4.

Although deriving expressions for the $B_{i,j}$ blocks in equation 4 is tedious due to notation, it is straightforward and simplifies to the following. Let Z_i and Q_i denote the k_i columns of Z and Q corresponding to the jth subspace partition. For an arbitrary matrix Z, the projection onto the tangent space of the PS manifold at a point Q is given by

$$\pi_{T_Q PS}(Z) = \frac{1}{2}\left[\,\pi_1 \mid \pi_2 \mid \ldots \mid \pi_m \mid \pi_{n-k}\,\right], \quad (5)$$

where

$$\pi_i = (Z_i - QZ^T Q_i) + (Q_i Z_i^T Q_i - Q_i Q_i^T Z_i). \quad (6)$$

With the tangent space projection defined, we now consider retractions on the partitioned subspace manifold. Ideally

---

Algorithm 1 Riemannian gradient descent on the partitioned subspace manifold

---

Input: loss function $f(Q)$, initial estimate Q_0,
    partition sizes $\{k_i\}_{i=1}^m$, step size α
for $i = 0, 1, 2, \ldots$ **do**
    compute $G_Q \leftarrow \nabla_Q f(Q_i)$
    compute $\Delta \leftarrow \pi_{T_{Q_i} PS}(G_Q)$ (Eqns. 5-6)
    compute $Q_{i+1} \leftarrow \text{qr}(Q_i - \alpha\Delta)$
    if $Q_i \approx Q_{i+1}$ **then**
        return Q_{i+1}

---

the exponential map would be used, but the computational cost associated with it makes it impractical. Fortunately, because our manifold is a quotient of the group of orthogonal matrices, the *Q-factor retraction* can be applied to efficiently compute updates.

Having derived the tangent space projection and an appropriate retraction, Riemannian gradient descent can be applied as described in section 3. Our approach is summarized in Algorithm 1.

4. Applications

To illustrate how the PS manifold can be used in practice, we applied it to the problems of multiple dataset analysis and domain adaptation. Specifically, our intention throughout these applications was not to establish new state-of-the-art methods, but rather to provide proof of concept and to demonstrate the properties of the PS manifold.

4.1. Multiple Dataset Analysis

First, we use the PS manifold to extract features that describe a collection of datasets, grouped according to whether they are unique to one dataset in particular (the *per-dataset features*) or shared between several (the *shared features*). Given a collection of D datasets $\mathcal{X} = \{X_i\}_{i=1}^D$, we define $D+1$ partitions, where the first D partitions will capture the per-dataset features and the last partition will capture the shared features. We note that in general, additional partitions could be defined to model more complex relationships, e.g., to learn features unique to distinct pairs or triplets of datasets. For simplicity, we consider only per-dataset and shared features in this section, and leave the problem of learning more general partitioning schemes as a promising direction for future work.

4.1.1. PROBLEM FORMULATION

Our approach is motivated by the *principal components analysis (PCA)*, which computes the k-dimensional subspace that best reconstructs a single input dataset (Dunteman, 1989). The subspace returned by PCA can be viewed

as the solution to the following optimization problem:

$$Y_{PCA} = \underset{Y \in Gr(n,k)}{\arg\min} \; ||X - XYY^T||_F^2 \,,$$

where $Gr(n, k)$ denotes the Grassmannian manifold of k-dimensional subspaces of $\mathbb{R}^n$, and $||\cdot||_F$ denotes the Frobenius norm. The expression $||X - XYY^T||_F^2$ calculates the reconstruction error between the original dataset X and its projection onto Y, which is given by XYY^T. Based on this optimization problem, we propose *Multiple-Dataset PCA* to solve the following related problem:

Definition 2 *Let X_i denote one of D datasets in a collection $\mathcal{X}$, and let $PS(n, k_{pd}, k_{sh})$ be a partitioned subspace manifold consisting of D per-dataset partitions of size k_{pd} and one shared partition of size k_{sh}. Furthermore, given a point $[Q]$ on the PS manifold, let Q_i and Q_{sh} denote the columns of Q that span the ith per-dataset partition and the shared partition, respectively. Finally, define $S_i = [\; Q_i \mid Q_{sh} \;]$ to be the union of the per-dataset partition for X_i and the shared partition. We define the Multi-Dataset PCA subspace to be the point on the PS manifold given by $Y_{MD\text{-}PCA} = Q_{MD\text{-}PCA} \begin{bmatrix} I_k & 0 \end{bmatrix}^T$, where $Q_{MD\text{-}PCA}$ is equal to:*

$$Q_{MD\text{-}PCA} = \underset{Q \in PS(n, k_{pd}, k_{sh})}{\arg\min} \sum_{i=1}^{D} \frac{||X_i - X_i S_i S_i^T||_F^2}{|X_i|}. \quad (7)$$

Given Algorithm 1 for performing Riemannian gradient descent on the partitioned subspace manifold, the MD-PCA algorithm is simple to state: apply Riemannian gradient descent using the gradient of the loss function in Equation 7 at each step. For MD-PCA, the gradient can be computed using the equations below.

$$\frac{\partial}{\partial Q_i} f(Q) = -2 \frac{X_i^T X_i Q_i}{|X_i|} \quad \text{and}$$

$$\frac{\partial}{\partial Q_{sh}} f(Q) = -2 \left(\sum_{i=1}^{D} \frac{X_i^T X_i}{|X_i|} \right) Q_{sh}$$

4.1.2. EXPERIMENTS

To evaluate MD-PCA, we used The Office+Caltech10 set, which is a standard object recognition benchmark containing four datasets of processed image data (Gong et al., 2012). Each dataset is drawn from a different source, which is either *Amazon* (A), *DSLR* (D), *Caltech* (C) or *Webcam* (W). The original images are encoded as 800-bin histograms of SURF features that have been normalized and z-scored to have zero mean and unit standard deviation in each dimension, as described in (Gong et al., 2012). Each image is associated with one of ten class labels, but these were not used in our experiments because MD-PCA, like PCA, is an unsupervised learning approach.
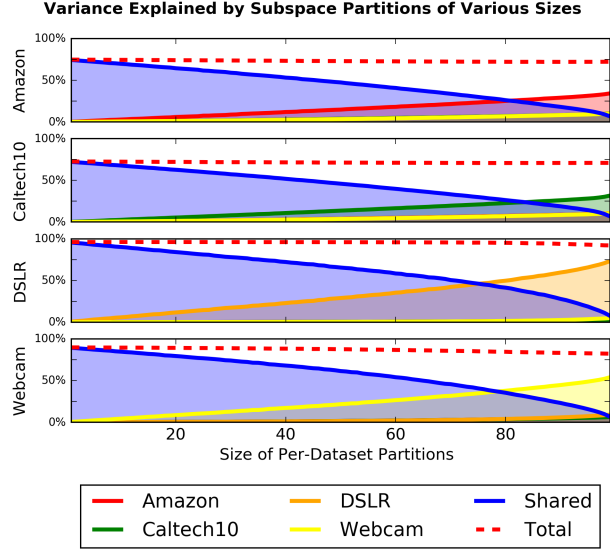

Figure 3. Results of MD-PCA for various sizes of k_{pd}. For each trial, k_{sh} was set to $400 - 4k_{pd}$ so that the total number of components learned was fixed at 400. For each dataset, the majority of its variance is explained by the shared partition and its per-dataset partition. Note that the total variance explained is relatively insensitive to the per-dataset partition sizes, suggesting that k_{pd} can be set liberally in practice.

To evaluate the sensitivity of the MD-PCA results with respect to the chosen partition sizes, we performed an exhaustive sweep over values for k_{pd}. Specifically, k_{pd} was varied between 1 and 99, and the size of the shared partition was set to $400 - 4k_{pd}$. Figure 3 illustrates the result of this analysis.

4.1.3. REMARKS

MD-PCA highlights several of the benefits and considerations of using the PS manifold in practice.

First, the mutual orthogonality constraints between the subspace partitions guarantee that a feature in one partition is not contained in any other, i.e., the dataset-specific features truly are unique. Because the manifold enforces mutual-orthogonality exactly, there is never redundancy in the features that might otherwise be introduced due to the constraints only being satisfied approximately.

Second, these same constraints cause common features to be included in Q_{sh}, and unique features to be included in the corresponding subspace Q_i, despite the lack of explicit discriminative terms in the loss function. Intuitively, if a feature is important for reconstructing X_i, then the ith term in Equation 7 will encourage it to be contained in either Q_i or Q_{sh}. If that feature is also important for reconstructing a different dataset X_j, then the jth term will behave sim-

ilarly, and the pressure to include it in Q_{sh} will dominate the pressure to include it in Q_i or Q_j. On the other hand, if the feature is unique to X_i, then the loss terms for the other datasets will encourage Q_{sh} to be populated with other features, so that the pressure to include it in Q_i will be uncontested while the pressure to include it in Q_{sh} will be neutralized. This behavior arises because the mutual orthogonality constraints captured by the PS manifold prevent a feature from being in multiple partitions simultaneously. If the partitions were not constrained in this way, for example if they were optimized as points on separate Grassmannian manifolds, then shared features would tend to mix with the per-dataset features, introducing redundancy and increasing the overall reconstruction error.

Finally, the use of the PS manifold simplifies the presentation of the algorithm by eliminating the steps that would otherwise have to be taken to enforce the mutual orthogonality constraints.

On the other hand, there are limitations to this approach. Foremost, while MD-PCA extracts and partitions features for multiple datasets, those datasets must represent the data in a consistent way. In particular, if samples from two datasets have different dimensionality, as is often the case with image datasets for example, then MD-PCA cannot be applied without first processing the datasets to map them to a common feature space.

Also, MD-PCA requires the user to specify not only the total subspace size, as in PCA, but also the size of each partition. Figure 3 shows how the variance explained by the subspace partitions changes as the per-dataset partition size is varied. In general, the total variance explained for each dataset does not change much as the partition size changes, until the per-dataset partitions become large enough to cause the shared partition to be very small. This observation implies that in practice, the size of the per-dataset partitions can be set liberally as long as the shared partition remains reasonably large. We hypothesize that the decrease in explained variance occurs when the shared partition becomes too small to contain all of the common features, which causes one dataset to "steal" the feature into its partition while the reconstruction error suffers for the other datasets.

4.2. Domain Adaptation

To further illustrate using the partitioned subspace manifold, we applied it to the problem of domain adaptation. Here, the goal is to leverage a labeled dataset X_s sampled from a *source distribution*, to improve accuracy when predicting an unlabeled target dataset X_t, sampled from a different *target distribution*. In this work, we consider the problem of performing classification in the target domain. Typical strategies for domain adaptation include project-

ing X_s and X_t onto a common space on which a classifier is then trained (Fernando et al., 2013; Gong et al., 2012; Gopalan, Li, and Chellappa, 2011), or finding a transformation of the datasets such that a classifier generalizes well to both domains (Kulis, Saenko, and Darrell, 2011; Chen et al., 2012).

The unique properties of the partitioned subspace manifold make it well suited to learning representations that generalize between the source and target. In the following, we provide an illustrative example of using the PS manifold in this way.

4.2.1. PROBLEM FORMULATION

In our approach, we wish to learn a set of subspace partitions that captures the common features between the source and target domains, while providing a representation that makes it easy to distinguish between classes. To accomplish this, we hypothesize that there are class-specific subspaces on which the source and target distributions are equivalent, so that a classifier trained on the projected data will generalize well across domains. Secondly, we constrain the subspaces to be mutually orthogonal to promote better discrimination between classes after projection. The partitioned subspace manifold makes it straightforward to specify and implement this objective.

We note that while the strategy of identifying domain- or label-specific features is not new to domain adaptation, the use of explicitly partitioned subspaces to capture those features has, to the best of our knowledge, not been investigated.

First, we define Q to be a partitioned subspace, where each partition will contain features that describe source samples with a given label y. To encourage the subspace to capture the structure of the source and target domains, we minimize the reconstruction error of approximating X_s and X_t by their projections onto Q. In addition, we promote discriminative subspaces by maximizing the magnitude of each source sample when projected onto its corresponding class partition, and minimizing the magnitude when projected onto the remaining partitions. Combining these, we define the *Class Discriminative Transfer (CDT) subspace* as follows.

Definition 3 *Let X_s be a source dataset with labels Y_s, where y_x denotes the label for sample x, and let X_t be a target dataset. Let L denote total number of unique labels. For each label y_i, we define a per-class partition Q_{y_i} of size k_{pc} to contain features that describe source samples with that label. Let $Q = [\ Q_{y_1} \mid Q_{y_2} \mid \dots \mid Q_{y_L} \mid Q_\perp\ ]$ be an orthonormal matrix containing these partitions in its first k columns, where $k = k_{pc}L$ is the combined size of the partitions. Finally, let $Q_{\bar{y}}$ be the union of the class-*

Table 1. Domain adaptation results with `Amazon` and `Caltech10` as source domains.

DA Method	A → C	A → D	A → W	C → A	C → D	C → W
ORIGFEAT	22.6	22.2	23.5	20.8	22.0	19.4
GFK(PCA, PCA)	35.6	34.9	34.4	36.9	35.2	33.9
GFK(PLS, PCA)	37.9	35.2	35.7	40.4	41.1	35.8
SA-SVM	**39.9**	**38.8**	39.6	46.1	39.4	38.9
PS-NB	37.7	36.3	**40.0**	**46.4**	**43.4**	**40.3**
PS-SVM	35.1	31.4	24.2	41.7	40.9	29.3

Table 2. Domain adaptation results with `DSLR` and `Webcam` as source domains.

DA Method	D → A	D → C	D → W	W → A	W → C	W → D
ORIGFEAT	27.7	24.8	53.1	20.7	16.1	37.3
GFK(PCA, PCA)	32.6	30.1	74.9	31.3	27.3	70.7
GFK(PLS, PCA)	36.2	32.7	79.1	35.5	29.3	71.2
SA-SVM	**42.0**	35.0	**82.3**	**39.3**	31.8	77.9
PS-NB	38.0	**35.8**	81.0	39.2	**31.8**	**78.1**
PS-SVM	32.0	31.8	53.8	30.6	23.2	77.9

specific partitions that do not correspond to label y. We define the Class Discriminative Transfer (CDT) subspace to be $Y_{CDT} = Q_{CDT} \begin{bmatrix} I_k & 0 \end{bmatrix}^T$, *where* Q_{CDT} *is equal to:*

$$Q_{CDT} = \underset{Q \in PS(n, k_{pc})}{\arg\min} f(Q), \tag{8}$$

and

$$f(Q) = ||X_s - X_s Q Q^T||_F^2 + ||X_t - X_t Q Q^T||_F^2$$
$$- \lambda \sum_{x \in X_s} \left(||x Q_{y_x} Q_{y_x}^T||_F^2 - ||z Q_{\bar{y}_x} Q_{\bar{y}_x}^T||_F^2 \right) \tag{9}$$

The first two terms in Equation 9 minimize the reconstruction error of the projected source and target data, while the third term encourages subspaces that allow samples from different classes to be easily distinguished. As a result, a set of partitions is learned that accurately reconstructs the source and target datasets while grouping the features into partitions that are useful for classification.

4.2.2. EXPERIMENTS

We evaluated our approach using the Office+Caltech10 dataset described in Subsection 4.1.2. Unlike the multiple dataset analysis application, here we use the provided labels to conduct supervised training on the source domain and to evaluate test accuracy on the target domain. Experiments were conducted for every possible pair of source and target domains, leading to 12 configurations.

To measure the effectiveness of using the CST subspace for domain adaptation, we compared against two commonly used approaches that are related to ours. The first, called the *Geodesic Flow Kernel (GFK)* method, computes one subspace to represent the source data and another to represent the target. A similarity between samples is then computed by integrating their inner product after projection onto all intermediate subspaces between the source and target. The similarities are then used as input to a 1-NN classifier (Gong et al., 2012). We include results for two versions of GFK: in the configuration GFK(PCA,PCA), the source and target subspaces are learned using PCA, while in GFK(PLS,PCA), partial least squares (PLS) is used to learn a source subspace that captures covariance between the source data and the available labels (Wold, Sjöström, and Eriksson, 2001).

The second approach we compare to is *Subspace Alignment (SA)*, which first computes source and target subspaces in a similar way to GFK. However, SA then projects each dataset onto its respective subspace and determines the optimal linear transformation to align the projected source data to the projected target data. After the transformation is applied, a support vector machine classifier is trained (Fernando et al., 2013). Finally, we compared our approach to the simple baseline of training an SVM on the original SIFT features without applying domain adaptation, which we call *OrigFeat* in our results. To ensure a fair comparison, we used the same evaluation protocols used for GFK and SA for our experiments (Gong et al., 2012; Fernando et al., 2013).

For our experiments, we conducted trials using partition sizes k_{cs} of 20 and 80 dimensions. Note that because the Office+Caltech10 datasets consist of samples with 800 features across 10 classes, the maximum class-specific partition size is 80. In all cases, the 80-dimensional subspace partitions performed best. The value of λ was fixed to be 2.0 for all experiments. After learning the CST subspace, we evaluated classification accuracy using Support Vector Machines (SVM) and Naive Bayes (NB). Tables 1 and 2

show the mean accuracy on all transfer tasks.

4.2.3. REMARKS

When comparing accuracy using SVM as a classifier, the partitioned subspace manifold is able to achieve comparable performance to GFK and SA. This suggests that the different partitions are able to learn discriminative subspaces that generalize to the target domain. Our approach also seems to be particularly well suited to using Naive Bayes for classification. In this case, the predictions are not made according to distance between the data, but according to the probability distributions of each feature.

5. Conclusion

In this work, we presented a formulation for the *Partitioned Subspace Manifold*, which captures the geometry of mutually orthogonal subspaces, and provided the necessary components for optimizing on the manifold using Riemannian optimization techniques. The ability to specify and easily optimize under general subspace constraints offers a significant amount of flexibility in designing new loss functions. To illustrate this, we proposed and analyzed multiple-dataset PCA, an extension of the principle component analysis which is able to group features from multiple datasets according to whether they are unique to a specific dataset or shared between them. In addition, we showed how the PS manifold can be used to learn class-specific features for domain adaptation and proposed a method that achieves comparable accuracy several standard approaches. We are actively interested in extending this formulation to allow subspace partitions to be adaptively resized so that they can be more effective in online settings, where the optimal partition sizes may change.

References

Absil, P.-A.; Mahony, R.; and Sepulchre, R. 2009. *Optimization algorithms on matrix manifolds*. Princeton University Press.

Chen, M.; Xu, Z.; Weinberger, K.; and Sha, F. 2012. Marginalized denoising autoencoders for domain adaptation. In Langford, J., and Pineau, J., eds., *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12. New York, NY, USA: ACM. 767–774.

Dunteman, G. H. 1989. *Principal components analysis*. Number 69. Sage.

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

Fernando, B.; Habrard, A.; Sebban, M.; and Tuytelaars, T. 2013. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE International Conference on Computer Vision*, 2960–2967.

Gong, B.; Shi, Y.; Sha, F.; and Grauman, K. 2012. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2066–2073. IEEE.

Gopalan, R.; Li, R.; and Chellappa, R. 2011. Domain adaptation for object recognition: An unsupervised approach. In *2011 international conference on computer vision*, 999–1006. IEEE.

He, J.; Balzano, L.; and Lui, J. 2011. Online robust subspace tracking from partial information. *arXiv preprint arXiv:1109.3827*.

He, J.; Balzano, L.; and Szlam, A. 2012. Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 1568–1575. IEEE.

Kim, T.-K.; Kittler, J.; and Cipolla, R. 2010. On-line learning of mutually orthogonal subspaces for face recognition by image sets. *IEEE Transactions on Image Processing* 19(4):1067–1074.

Kulis, B.; Saenko, K.; and Darrell, T. 2011. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, 1785–1792. Washington, DC, USA: IEEE Computer Society.

Turaga, P., and Chellappa, R. 2009. Locally time-invariant models of human activities using trajectories on the grassmannian. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2435–2441. IEEE.

Wold, S.; Sjöström, M.; and Eriksson, L. 2001. Plsregression: a basic tool of chemometrics. *Chemometrics and intelligent laboratory systems* 58(2):109–130.