STOCHASTIC CANONICAL CORRELATION ANALYSIS: A RIEMANNIAN APPROACH

Anonymous authors
Paper under double-blind review

ABSTRACT

We present an efficient stochastic algorithm (RSG+) for canonical correlation analysis (CCA) derived via a differential geometric perspective of the underlying optimization task. We show that exploiting the Riemannian structure of the problem reveals natural strategies for modified forms of manifold stochastic gradient descent schemes that have been variously used in the literature for numerical optimization on manifolds. Our developments complement existing methods for this problem which either require $O(d^3)$ time complexity per iteration with $O(\frac{1}{\sqrt{t}})$ convergence rate (where d is the dimensionality) or only extract the top 1 component with $O(\frac{1}{t})$ convergence rate. In contrast, our algorithm achieves $O(d^2k)$ runtime complexity per iteration for extracting top k canonical components with $O(\frac{1}{t})$ convergence rate. We present our theoretical analysis as well as experiments describing the empirical behavior of our algorithm, including a potential application of this idea for training fair models where the label of protected attribute is missing or otherwise unavailable.

1 INTRODUCTION

Canonical correlation analysis (CCA) is a popular method for evaluating correlations between two sets of variables. It is commonly used in unsupervised multi-view learning, where the multiple views of the data may correspond to image, text, audio and so on Rupnik & Shawe-Taylor (2010); Chaudhuri et al. (2009); Luo et al. (2015). Classical CCA formulations have also been extended to leverage advances in representation learning, for example, Andrew et al. (2013) showed how the CCA can be interfaced with deep neural networks enabling modern use cases. Many results over the last few years have used CCA or its variants for problems including measuring representational similarity in deep neural networks Morcos et al. (2018), speech recognition Couture et al. (2019), etc.

The goal in CCA is to find linear combinations within two random variables $\mathbf{X}$ and $\mathbf{Y}$ which have maximum correlation with each other. Formally, the CCA problem is defined in the following way. Given a pair of random variables, a d_x-variate random variable $\mathbf{X}$ and a d_y-variate random variable $\mathbf{Y}$, with unknown joint probability distribution, find the projection matrices $U \in \mathbf{R}^{d_x \times k}$ and $V \in \mathbf{R}^{d_y \times k}$, with $k \leq \min\{d_x, d_y\}$, such that the correlation is maximized:

$$\text{maximize trace}\left(U^T \mathbb{E}_{\mathbf{X},\mathbf{Y}}\left[X^T Y\right] V\right) \text{ s.t. } U^T \mathbb{E}_{\mathbf{X}}\left[X^T X\right] U = I_k, V^T \mathbb{E}_{\mathbf{Y}}\left[Y^T Y\right] V = I_k \quad (1)$$

Here, X, Y are samples of $\mathbf{X}$ and $\mathbf{Y}$ respectively. The objective function in equation 1 is the expected cross-correlation in the projected space and the constraints specify that different canonical components should be decorrelated. Let $C_X = \mathbb{E}_{\mathbf{X}}[X^T X]$ and $C_Y = \mathbb{E}_{\mathbf{Y}}[Y^T Y]$ be the covariance matrices, and $C_{XY} = \mathbb{E}_{(\mathbf{X},\mathbf{Y})}[X^T Y]$ denote the cross-covariance. Let us define the whitened covariance $T := C_X^{-1/2} C_{XY} C_Y^{-1/2}$ and Φ_k (and Ψ_k) contains the top-k left (and right) singular vectors of T. It is known Golub & Zha (1992) that the optimal solution of equation 1 is achieved at $U^* = C_X^{-1/2}\Phi_k$, $V^* = C_Y^{-1/2}\Psi_k$.

In practice, we may be given two views of N samples as $X \in \mathbf{R}^{N \times d_x}$ and $Y \in \mathbf{R}^{N \times d_y}$. A natural approach to solving CCA is based on on the following sequence of steps. We first compute the *empirical* covariance and cross-covariance matrices, namely, $\widetilde{C}_{\mathbf{X}} = {}^1\!/_N X^T X$, $\widetilde{C}_{\mathbf{Y}} = {}^1\!/_N Y^T Y$ and $\widetilde{C}_{\mathbf{XY}} = {}^1\!/_N X^T Y$. We then calculate the empirical whitened cross-covariance matrix $\widetilde{T}$, finally, compute U^*, V^* by applying a k-truncated SVD to $\widetilde{T}$.

Runtime and memory considerations. The above procedure is simple but is only feasible when the data matrices are small. But in most modern applications, not only are the datasets large but also the dimension d (let $d = \max\{d_x, d_y\}$) of each sample can be quite high, especially if representations are being learned using deep neural network models. As a result, the computational footprint of the foregoing algorithm can be quite high. This has motivated the study of stochastic optimization routines for solving CCA. Observe that in contrast to the typical settings where stochastic optimization schemes are most effective, the CCA objective does not decompose over samples in the dataset. Many efficient strategies have been proposed in the literature: for example, Ge et al. (2016); Wang et al. (2016) present Empirical Risk Minimization (ERM) models which optimize the empirical objective. More recently, Gao et al. (2019); Bhatia et al. (2018); Arora et al. (2017) describe proposals that optimize the population objective. To summarize the approaches succinctly, if we are satisfied with identifying the top 1 component of CCA, effective schemes are available by utilizing either extensions of the Oja's rule Oja (1982) to generalized eigenvalue problem Bhatia et al. (2018) or the alternating SVRG algorithm Gao et al. (2019)). Otherwise, a stochastic approach must make use of an explicit whitening operation which involves a cost of d^3 for each iteration Arora et al. (2017).

Observation. Most approaches either directly optimize equation 1 or instead a reparametrized or regularized form Ge et al. (2016); Allen-Zhu & Li (2016); Arora et al. (2017). Often, the search space for U and V corresponds to the entire $\mathbf{R}^{d\times k}$ (ignoring the constraints for the moment). But if the formulation could be cast in a form which involved approximately writing U and V as a product of several matrices with nicer properties, we may be able to obtain specialized routines which are tailored to exploit those properties. Such a reformulation is not difficult to derive – where the matrices involved in expressing U and V can be identified as objects that live in well studied structured/geometric spaces. Then, utilizing the geometry of the space and borrowing relevant tools from differential geometry leads to an efficient approximate algorithm for top-k CCA which optimizes the population objective in a streaming fashion.

Contributions. (a) First, we re-parameterize the top-k CCA problem as an optimization problem on specific matrix manifolds, and show that it is equivalent to the original formulation in equation 1. (b) Informed by the geometry of the manifold, we derive stochastic gradient descent algorithms for solving the re-parameterized problem with $O(d^2k)$ cost per iteration and provide convergence rate guarantees. (c) This analysis provides a direct mechanism to obtain an upper bound on the number of iterations needed to guarantee an ϵ error w.r.t. the population objective for the CCA problem. (d) The algorithm works in a streaming manner so it easily scales to large datasets and we do not need to assume access to the full dataset at the outset. (e) We present empirical evidence for both the standard CCA model and the DeepCCA setting Andrew et al. (2013), describing advantages and limitations.

2 Stochastic CCA: Reformulation, Algorithm and Analysis

In order to develop an alternative efficient formulation of Stochastic CCA, we will make use of the following manifolds: (a) $\mathsf{St}(p, n)$: the manifold consists of $n \times p$ column orthonormal matrices (b) $\mathsf{Gr}(p, n)$: the manifold consists of p-dimensional subspaces in $\mathbf{R}^n$ (c) $\mathsf{SO}(n)$, the manifold/group consists of $n \times n$ special orthogonal matrices. A more detailed exposition of the geometric properties and operations for these manifolds are given in Appendix. Note that this setting is completely distinct from performing CCA on *manifold valued* data Kim et al. (2014).

Let us recall the objective function for CCA as given in equation 1. We denote $X \in \mathbf{R}^{N\times d_x}$ as the matrix consisting of the samples $\{\mathbf{x}_i\}$ drawn from a zero mean random variable $\mathbf{X} \sim \mathcal{X}$ and $Y \in \mathbf{R}^{N\times d_y}$ denotes the matrix consisting of samples $\{\mathbf{y}_i\}$ drawn from a zero mean random variable $\mathbf{Y} \sim \mathcal{Y}$. *Note:* For notational and formulation simplicity, we assume that $d_x = d_y = d$ in the remainder of the paper although the results hold for general d_x and d_y.

Let C_X, C_Y be the covariance matrices of $\mathbf{X}$, $\mathbf{Y}$. Also, let C_{XY} be the cross-correlation matrix between $\mathbf{X}$ and $\mathbf{Y}$. Then, we can write the CCA objective as

$$\max_{U,V} \quad F = \text{trace}\left(U^T C_{XY} V\right) \quad \text{subject to} \quad U^T C_X U = I_k \quad V^T C_Y V = I_k \tag{2}$$

Here, $U \in \mathbf{R}^{d\times k}$ ($V \in \mathbf{R}^{d\times k}$) is the matrix consisting of $\{\mathbf{u}_j\}$ ($\{\mathbf{v}_j\}$) , where ($\{\mathbf{u}_j\}, \{\mathbf{v}_j\}$) are the canonical directions. The constraints in equation 2 are called *whitening constraints*.

Let us define matrices $\widetilde{U}, \widetilde{V} \in \mathbf{R}^{d\times k}$ which lie on the Stiefel manifold, $\mathsf{St}(k, d)$. Also, let $S_u, S_v \in \mathbf{R}^{k\times k}$ denote upper triangular matrices and $Q_u, Q_v \in \mathsf{SO}(k)$. We can rewrite the above equation and the constraint as follows.

A Reformulation for CCA

$$\max_{\substack{\widetilde{U},\widetilde{V},S_u,S_v,Q_u,Q_v \\ U=\widetilde{U}Q_uS_u;\ V=\widetilde{V}Q_vS_v}} \quad \widetilde{F} = \text{trace}\left(U^T C_{XY} V\right) \tag{3a}$$

$$\text{subject to} \quad U^T C_X U = I_k;\ V^T C_Y V = I_k \tag{3b}$$

$$\widetilde{U}, \widetilde{V} \in \mathsf{St}(k, d);\ Q_u, Q_v \in \mathsf{SO}(k);\ S_u, S_v \text{ is upper triangular}$$

Here, we will maximize equation 3a with respect to $\widetilde{U}$, $\widetilde{V}$, S_u, S_v, Q_u, and Q_v satisfying equation 3b.

Remark 1. *Observe that in equation 2, while U and V should decorrelate C_X and C_Y respectively, the optimization/search strategy is unrestricted and treats them as arbitrary matrices. In contrast, in equation 3a–equation 3b, we decompose U (and V) as $\widetilde{U}Q_uS_u$ (and $\widetilde{V}Q_vS_v$) where each component in this decomposition is a structured matrix. We will use this geometry explicitly to help the optimization.*

Intuition behind the decomposition $U = \tilde{U}Q_uS_u$: First, we note that CCA seeks to maximize the total correlation under the constraint that different components are decorrelated. The difficult part in the optimization is to ensure decorrelation, which leads to a higher complexity in existing streaming CCA algorithms. Our **key observation** is the following: Recall that by doing principal component analysis (PCA), the resultant projection matrix will exactly satisfy the decorrelation condition (projection matrix consists of the eigen-vectors of X^TX). Interestingly, there are already several algorithms which perform PCA in a streaming fashion and achieve a low complexity (d^2k runtime complexity per iteration). A natural question to ask is: Can we utilize streaming PCA algorithm to help us obtain an efficient streaming CCA algorithm? It turns out that the subspace of CCA solution and that of PCA can be related by a rotation matrix and an upper triangular matrix. This is the rationale behind the form of decomposition shown above. We will shortly describe the rationale in more detail.

Below, we provide a high-level reasoning behind why substituting equation 2 with equation 3a–equation 3b is sensible. We will operationalize this idea to describe a general scheme whose behavior for the CCA problem can be analyzed without getting into specific details of the optimization itself.

2.1 How to use the reformulation in equation 3?

We explain how equation 3a–equation 3b can be used to derive an efficient solution scheme. We first start by stating some mild assumptions needed for the algorithm and the subsequent analysis.

Assumptions: (a) The random variables $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \Sigma_x)$ and $\mathbf{Y} \sim \mathcal{N}(\mathbf{0}, \Sigma_y)$ with $\Sigma_x \preceq cI_d$ and $\Sigma_y \preceq cI_d$ for some $c > 0$. **(b)** The samples X and Y drawn from $\mathcal{X}$ and $\mathcal{Y}$ respectively have zero mean. **(c)** For a given $k \leq d$, Σ_x and Σ_y have non-zero top-k eigen values.

A high-level solution to optimize $\widetilde{F}$ in equation 3: Consider the following general scheme,

> **(a)** Let us initialize $\widetilde{U}, \widetilde{V} \in \mathsf{St}(k, d)$ as the top-k eigen vectors of $C_X = (1/N)X^TX$ and $C_Y = (1/N)Y^TY$ respectively; **(b)** Let us initialize Q_u and Q_v to be random $\mathsf{SO}(k)$ matrices; **(c)** Let us set up S_u and S_v to be diagonal matrices with the diagonal entries to be the square root of the inverse of the top-k eigen values; **(d)** *Observe that with this initialization, the constraints in equation 3b are satisfied.* With the above initialization as a feasible solution for U and V, we may optimize equation 3a while satisfying equation 3b. The specific details of how this is done (described in the next section) is not critical at this point as long as we assume that a suitable numerical optimization scheme exists and can be implemented.

With the component matrices, we can construct the solution as $U = \widetilde{U}Q_uS_u$ and $V = \widetilde{V}Q_vS_v$.

Why the solution makes sense? We now show how the presented solution, assuming access to an effective numerical procedure, approximates the CCA problem presented in equation 2. We formally state the result in the following theorem with a sketch of proof (appendix includes the full proof).

Theorem 1. *Using the hypothesis, assumptions and notations as above, the approximation error $E = \|F - \widetilde{F}\|$ is bounded and goes to zero while the whitening constraints in equation 3b are satisfied.*

Sketch. We project the data matrix X and Y on to the solution spaces namely $\widetilde{U}Q_uS_u$ and $\widetilde{V}Q_vS_v$. Let the projected solutions be denoted by $\widetilde{X}_u = X\widetilde{U}Q_uS_u$ and $\widetilde{Y}_v = Y\widetilde{V}Q_vS_v$. We must compare it with the projected solutions for the original CCA model, $X_u = XU$ and $Y_v = YV$. Using Prop. 1 (see below), we can show that $\widetilde{X}_u, X_u, \widetilde{Y}_v, Y_v$ are sub-Gaussian. Now, we can upper bound $\|\widetilde{X}_u - X_u\|$ and $\|\widetilde{Y}_v - Y_v\|$ using Prop. 2 (see below) which completes the proof. □

Recall that a random variable $\mathbf{X}$ is called sub-Gaussian if the norm given by $\|\mathbf{X}\|_\star := \inf\{d \geq 0 | \mathbf{E}_{\mathbf{X}}\left[\exp\left(\text{trace}(X^TX)/d^2\right)\right] \leq 2\}$ is finite.

Proposition 1. *Let $U \in \mathbf{R}^{d\times k}$, then $\mathcal{X}U$ is sub-Gaussian.*

Proposition 2 (Reiß et al. (2020))**.** *Let $\mathbf{X}$ be a random variable which follows a sub-Gaussian distribution. Let $\widehat{X}$ be the approximation of $X \in \mathbf{R}^{N\times d}$ (samples drawn from $\mathcal{X}$) with the top-k principal vectors. Let $\widetilde{C}_X$ be the covariance of $\widehat{X}$. Also, assume that λ_i is the ith eigen value of C_X for $i = 1, \cdots, d-1$ and $\lambda_i \geq \lambda_{i+1}$ for all i. Then, the PCA reconstruction error, denoted by $\mathcal{E}_k = \|X - \widehat{X}\|$ (in Frobenius norm sense) can be upper bounded as follows*

$$\mathcal{E}_k \leq \min\left(\sqrt{2k}\|\Delta\|_2, \frac{2\|\Delta\|_2^2}{\lambda_k - \lambda_{k+1}}\right), \quad \textit{where } \Delta = C_X - \widetilde{C}_X.$$

Now, the only unresolved issue is an optimization scheme for equation 3a satisfying the constraints in equation 3b.

2.2 How to numerically optimize (3a) satisfying constraints in (3b)?

Overview. We now describe how to maximize the formulation in equation 3a–equation 3b with respect to $\widetilde{U}$, $\widetilde{V}$, Q_u, Q_v, S_u and S_v. We will first compute top-k principal vectors to get $\widetilde{U}$ and $\widetilde{V}$. Then, we will simply use a gradient update rule to solve for Q_u, Q_v, S_u and S_v. Since all these matrices are "structured", care must be taken to ensure that the matrices remain on their respective manifolds. Making use of the geometry of the manifolds will offer desirable properties. We re-purpose a Riemannian stochastic gradient descent (RSGD) to do this, so call our algorithm *RSG+*. While this will be sufficient, other more sophisticated Riemannian optimization techniques can be substituted in. A concise review of different Riemannian optimization methods is available in Absil et al. (2007).

The algorithm block is presented in Algorithm 1 (a direct implementable block for the algorithm including the expression for gradient is presented in the Appendix A.3). The algorithm consists of four main blocks denoted by different colors, namely **(a)** the Red block deals with update $\widetilde{U}$ and $\widetilde{V}$ from the gradient for top-k principal vectors; **(b)** the Green block describes computation of gradient of equation 3; **(c)** the Gray block describes computation of gradient to update the canonical directions and finally **(d)** the Blue block performs a batch update of the canonical directions.

Gradient calculations. The gradient update for $\widetilde{U}, \widetilde{V}$ is divided into two parts **(a)** The (Red block) gradient updates "principal" directions (denoted by $\nabla_{\widetilde{U}}\widetilde{F}_{\text{pri}}$ and $\nabla_{\widetilde{V}}\widetilde{F}_{\text{pri}}$), which is specifically to designed to satisfy the *whitening constraint*. Note that this requires the Riemannian gradient computation on the Grassmannian **(b)** The (green block) gradient from the objective function in equation 3, denoted by $\nabla_{\widetilde{U}}\widetilde{F}_{\text{can}}$ and $\nabla_{\widetilde{V}}\widetilde{F}_{\text{can}}$. This step requires first calculating the Euclidean gradient and then projecting on to the tangent space of $\mathsf{St}(k, d)$.

The gradient updates for Q_u, Q_v, S_u, S_v are given in the green block, denoted by $\nabla_{Q_u}\widetilde{F}_{\text{can}}$, $\nabla_{Q_v}\widetilde{F}_{\text{can}}$, $\nabla_{S_u}\widetilde{F}_{\text{can}}$ and $\nabla_{S_v}\widetilde{F}_{\text{can}}$. Note that unlike the previous step, this gradient only has components from objective function in equation 3. As before, this step requires first computing the Euclidean gradient and then projecting on to the tangent space of the underlying Riemannian manifold.

Finally, we get the gradient to update the canonical directions by combining gradients from principal directions and Eqs. equation 3, which is shown in gray block. With these gradients we can perform

a batch update as shown in the blue block. As described in the previous section and with the help of convergence result presented in Propositions equation 3, equation 4, these solutions can be shown (under some assumptions) to approximately optimize the CCA objective in equation 2.

Proof of convergence of Algorithm 1. Now, we give a proof of convergence for the algorithm above. We present two results stating the asymptotic proof of convergence for the top-k principal vectors and the canonical directions as stated in the algorithm.

Proposition 3 (Chakraborty et al. (2020))**.** *(Asymptotically) If the samples, X, are drawn from a Gaussian distribution, then the gradient update rule presented in 4 in Algorithm 1 returns an orthonormal basis – the top-k principal vectors of the covariance matrix C_X.*

Proposition 4. *(Bonnabel (2013)) Consider a connected Riemannian manifold $\mathcal{M}$ with injectivity radius bounded from below by $I > 0$. Assume that the sequence of step sizes (γ_l) satisfy the condition (a) $\sum \gamma_l^2 < \infty$ (b) $\sum \gamma_l = \infty$. Suppose $\{A_l\}$ lie in a compact set $K \subset \mathcal{M}$. We also suppose that $\exists D > 0$ such that, $g_{A_l}\left(\nabla_{A_l}\widetilde{F}, \nabla_{A_l}\widetilde{F}\right) \leq D$. Then $\nabla_{A_l}\widetilde{F} \to 0$ and $l \to \infty$.*

Notice that in our problem, the manifold $\mathcal{M}$ can be $\mathsf{Gr}(p,n)$, $\mathsf{St}(p,n)$ or $\mathsf{SO}(p)$. Hence all the assumptions in Proposition 4 are satisfied if we guarantee the step sizes satisfy the aforementioned condition. One example of the step sizes that satisfy the property is $\gamma_l = \frac{1}{l+1}$.

2.3 Convergence rate and complexity of the RSG+ algorithm

In this section, we describe the convergence rate and complexity of the algorithm proposed in Algorithm 1. Observe that the key component of Algorithm 1 is a Riemannian gradient update. Let A_t be the generic entity needed to be updated in the algorithm using the Riemannian gradient update $A_{t+1} = \mathsf{Exp}_{A_t}\left(-\gamma_t \nabla_{A_t}\widetilde{F}\right)$, where γ_t is the step size at time step t. Also assume $\{A_t\} \subset \mathcal{M}$ for a Riemannian manifold $\mathcal{M}$. The following proposition states that under certain assumptions, the Riemannian gradient update has a convergence rate of $O\left(\frac{1}{t}\right)$.

Proposition 5. *(Nemirovski et al. (2009)Bécigneul & Ganea (2018)) Let $\{A_t\}$ lie inside a geodesic ball of radius less than the minimum of the injectivity radius and the strong convexity radius of $\mathcal{M}$. Assume $\mathcal{M}$ to be a geodesically complete Riemannian manifold with sectional curvature lower bounded by $\kappa \leq 0$. Moreover, assume that the step size $\{\gamma_t\}$ diverges and the squared step size converges. Then, the Riemannian gradient descent update given by $A_{t+1} = \mathsf{Exp}_{A_t}\left(-\gamma_t \nabla_{A_t}\widetilde{F}\right)$ with a bounded $\nabla_{A_t}\widetilde{F}$, i.e., $\|\nabla_{A_t}\widetilde{F}\| \leq C < \infty$ for some $C \geq 0$, converges in the rate of $O\left(\frac{1}{t}\right)$.*

To evaluate the time complexity of our model, we must look at the main compute-heavy steps needed. The basic modules are Exp and Exp^{-1} maps for $\mathsf{St}(k,d)$, $\mathsf{Gr}(k,d)$ and $\mathsf{SO}(k)$ manifolds (see Table 3). Observe that the complexity of these modules is influenced by the complexity of svd needed for the Exp map for the St and Gr manifolds. Since performing singular value decomposition of a $d \times k$ matrix takes $O(d^2k)$ time, the time complexity is $O(d^2k)$.

Algorithm 1: Riemannian SGD based algorithm (RSG+) to compute canonical directions

Input: $X \in \mathbf{R}^{N \times d_x}, Y \in \mathbf{R}^{N \times d_y}, k > 0$
Output: $U \in \mathbf{R}^{d_x \times k}, V \in \mathbf{R}^{d_y \times k}$

Initialize $\widetilde{U}, \widetilde{V}, Q_u, Q_v, S_u, S_v$;
Partition data X, Y into batches of size B. Let j^{th} batch be denoted by X_j and Y_j ;
for $j \in \{1, \cdots, \lfloor \frac{N}{B} \rfloor\}$ **do**

Gradient for top-k principal vectors: calculating $\nabla_{\widetilde{U}}\widetilde{F}_{\text{pri}}, \nabla_{\widetilde{V}}\widetilde{F}_{\text{pri}}$
1. Partition X_j (Y_j) into L ($L = \lfloor \frac{B}{k} \rfloor$) blocks of size $d_x \times k$ ($d_y \times k$);
2. Let the lth block be denoted by Z_l^x (Z_l^y);
3. Orthogonalize each block and let the orthogonalized block be denoted by $\hat{Z}_l^x$ ($\hat{Z}_l^y$);
4. Let the subspace spanned by each $\hat{Z}_l^x$ (and $\hat{Z}_l^y$) be $\hat{\mathcal{Z}}_l^x \in \mathsf{Gr}(k, d_x)$ (and $\hat{\mathcal{Z}}_l^y \in \mathsf{Gr}(k, d_y)$);
5. Update $\nabla_{\widetilde{U}}\widetilde{F}_{\text{pri}}$ and $\nabla_{\widetilde{V}}\widetilde{F}_{\text{pri}}$ based on $\hat{\mathcal{Z}}_l^x$ and $\hat{\mathcal{Z}}_l^y$ respectively.

Gradient from equation 3: calculating $\nabla_{\widetilde{U}}\widetilde{F}_{\text{can}}, \nabla_{\widetilde{V}}\widetilde{F}_{\text{can}}, \nabla_{Q_u}\widetilde{F}_{\text{can}}, \nabla_{Q_v}\widetilde{F}_{\text{can}}, \nabla_{S_u}\widetilde{F}_{\text{can}}, \nabla_{S_v}\widetilde{F}_{\text{can}}$
Calculation of the Riemannian gradients of $\widetilde{U}$, $\widetilde{V}$, Q_u, Q_v, S_u and S_v from equation 3, i.e., objective from CCA.

Gradient to update canonical directions: calculating $\nabla_A \widetilde{F}$ where, $A \in \{\widetilde{U}, \widetilde{V}, Q_u, Q_v, S_u, S_v\}$
The final gradients of $\widetilde{U}$ and $\widetilde{V}$ is a combination of gradient from objective for principal vectors and CCA;
On the other hand, the gradients for Q_u, Q_v, S_u, S_v is from only CCA objective;

Batch update of canonical directions
$A = \mathsf{Exp}_A\left(-\gamma_j \nabla_A \widetilde{F}\right)$ where, A is a generic entity: $A \in \{\widetilde{U}, \widetilde{V}, Q_u, Q_v, S_u, S_v\}$;

end
$U = \widetilde{U} Q_u S_u$ and $V = \widetilde{V} Q_v S_v$;

3 Experiments

We first evaluate RSG+ for extracting top-k canonical components on three benchmark datasets and show that it performs favorably compared with Arora et al. (2017). Then we show that our RSG+ can also fit naturally into feature learning on DeepCCA setting Andrew et al. (2013), and has the ability to scale to large feature dimensions where the non-stochastic method fails to. Finally we show that our RSG+ can be used to

improve fairness of deep neural networks without the need for labels of protected attributes during training.

3.1 CCA ON FIXED DATASETS

Datasets and baseline. We conduct experiments on three benchmark datasets (MNIST LeCun et al. (2010), Mediamill Snoek et al. (2006) and CIFAR-10 Krizhevsky (2009)) to evaluate the performance of RSG+ to extract top-k canonical components. To our best knowledge, Arora et al. (2017) is the only previous work which stochastically optimizes the population objective in a streaming fashion and can extract top-k components, so we compare our RSG+ with the matrix stochastic gradient (MSG) method proposed in Arora et al. (2017) (There are two methods proposed in Arora et al. (2017) and we choose MSG because it performs better in the experiments of Arora et al. (2017)). The details about the three datasets and how we process them are as follows:

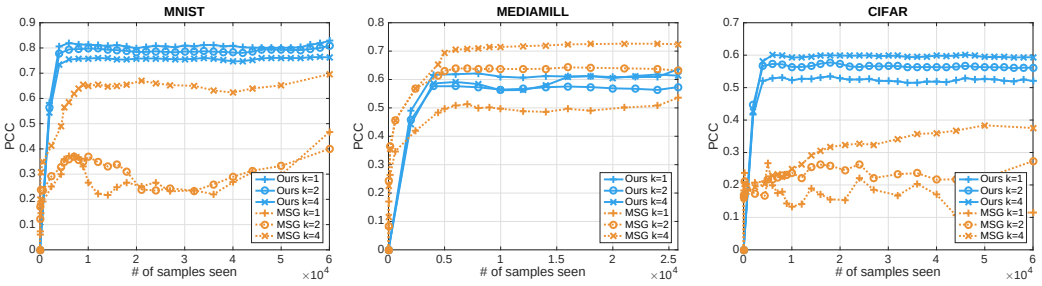
MNIST LeCun et al. (2010): MNIST contains grey-scale images of size 28×28. We use its full training set containing 60K images. Every image is split into left/right half, which are used as the two views. **Mediamill** Snoek et al. (2006): Mediamill contains around 25.8K paired features of videos and corresponding commentary of dimension $120, 101$ respectively. **CIFAR-10** Krizhevsky (2009): CIFAR-10 contains 60K 32×32 color images. Like MNIST, we split the images into left/right half and use them as two views.

Evaluation metric. We choose to use Proportion of Correlations Captured (PCC) which is widely used Ma et al. (2015); Ge et al. (2016), partly due to its efficiency, especially for relatively large datasets. Let $\hat{U} \in R^{d_x \times k}, \hat{V} \in R^{d_y \times k}$ denote the estimated subspaces returned by RSG+, and $U^* \in R^{d_x \times k}, V^* \in R^{d_y \times k}$ denote the true canonical subspaces (all for top-k). The PCC is defined as PCC $= \frac{\text{TCC}(X\hat{U}, Y\hat{V})}{\text{TCC}(XU^*, YV^*)}$, where TCC is the sum of canonical correlations between two matrices.

Performance. See A.4 for the implementation deails. The performance in terms of PCC as a function of # of seen samples (coming in a streaming way) are shown in Fig. 1, and the runtime is reported in A.5 . Our RSG+ captures more correlation than MSG Arora et al. (2017) while being $5 - 10$ times faster. One case where our RSG+ underperform Arora et al. (2017) is when the top-k eigenvalues are dominated by the top-l eigenvalues with $l < k$ (Fig. 1b): on Mediamill dataset, the top-4 eigenvalues of the covariance matrix in view 1 are: $8.61, 2.99, 1.15, 0.37$. The first eigenvalue is dominantly large compared with the rest and our RSG+ performs better for $k = 1$ and worse than Arora et al. (2017) for $k = 2, 4$. We also plot the runtime of RSG+ under different data dimension (set $d_x = d_y = d$) and number of total samples sampled from joint gaussian distribution in A.5.

3.2 CCA FOR DEEP FEATURE LEARNING

Background and motivation. A deep neural network (DNN) extension of CCA was proposed by Andrew et al. (2013) and has become popular in the multi-view representation learning tasks. The idea is to learn a deep neural network as the mapping from original data space to a latent space where the canonical correlations are maximized. We refer the reader to Andrew et al. (2013) for details of the task. Since deep neural networks are usually trained using SGD on mini-batches, this requires getting estimate of CCA objective at every iteration in a streaming fashion, thus our RSG+ can be a natural choice here. We conduct experiments on a noisy version of MNIST dataset to evaluate RSG+.



(a) on MNIST (b) on Mediamill (c) on CIFAR

Figure 1: Performance on three datasets in terms of PCC as a function of # of seen samples.

Dataset. We follow Wang et al. (2015a) to construct a noisy version of MNIST: View 1 is a randomly sampled image which is first rescaled to $[0, 1]$ and then rotated by a random angle from $[-\frac{\pi}{4}, \frac{\pi}{4}]$. View 2 is randomly sampled from the same class as view 1. Then we add independent uniform noise from $[0, 1]$ to each pixel. Finally the image is truncated into $[0, 1]$ to form the view 2.

Implementation details. We use a simple 2-layer MLP with ReLU nonlinearity as the DNN, where the hidden dimension in the middle is 512 and the output feature dimension is $d \in \{100, 500, 1000\}$. After the network is trained on the CCA objective, we use a linear Support Vector Machine (SVM) to measure the classification accuracy on output latent features. Andrew et al. (2013) uses the closed form CCA objective on the current batch directly, which costs $O(d^3)$ memory and time for every iteration.

Table 1: Results of feature learning on MNIST. N/A means fail to give a result on our hardware and pytorch platform

Accuracy(%)	$d = 100$	$d = 500$	$d = 1000$
DeepCCA	80.57	N/A	N/A
Ours	79.79	84.09	86.39

Performance. Table 1 shows that we get similar performance when $d = 100$ and can scale to large latent dimensions $d = 1000$ while the batch method Andrew et al. (2013) encounters numerical difficulty on our GPU resources and Pytorch Paszke et al. (2019) platform in the process of doing eigen-decomposition of $d \times d$ matrix when $d = 500$, and becomes intractable with our GPU resources when d is larger than 1000.

3.3 CCA FOR FAIRNESS

Background and motivation. Fairness is becoming an important issue to consider in the design and deployment of learning algorithms. A common strategy to make an algorithm fair is to remove the influence of one or more protected (or discriminatory) attributes or variables when training the models, see Lokhande et al. (2020). Most current methods assume that the labels of protected attributes are known during training but this may not always be possible. The CCA model enables us to consider a slightly different setting, where we may not have per-sample protected attributes which may be sensitive or otherwise hard to obtain for third-parties Price & Cohen (2019). On the other hand, we assume that a model trained to predict the protected attribute labels has been trained and is provided. For example, if the protected attribute is gender, we only assume that a well trained classifier which is trained to predict gender from the samples is available rather than sample-wise gender values themselves. We next demonstrate that fairness of the model, using standard measures, can be improved by putting constraints on correlation value from CCA.

Table 2: Fairness results on CelebA. We applied CCA on three different layers in Resnet-18 respectively. See appendix A.8 for the exact positions of conv $0, 1, 2$.

	Accuracy(%)	DEO(%)	DDP(%)
Unconstrained	76.3	22.3	4.8
Ours-conv0	76.5	17.4	**1.4**
Ours-conv1	**77.7**	**15.3**	3.2
Ours-conv2	75.9	22.0	2.8

Dataset. CelebA Wang et al. (2015b) consists of 200K celebrity face images from the internet. There are up to 40 labels, each of which is binary-valued. Here, we closely follow Lokhande et al. (2020) to focus on the *attactiveness* attribute (which we want to train a classifier to predict) and the *gender* is treated as the protected attribute since it may lead to an unfair classifier for this dataset according to Lokhande et al. (2020).

Method. Our strategy is inspired by Morcos et al. (2018) which showed that canonical correlations can reveal the similarity in neural networks: when two networks (same architecture) are trained using different labels/schemes for example, canonical correlations can indicate how similar their features are. Our observation is the following. Consider a classifier that is trained on gender (the protected attribute), and another classifier that is trained on *attractiveness*, if the features extracted by the latter model share a high similarity with the one trained to predict gender, then it is more likely that the latter model is influenced by features in the image pertinent to gender, which will lead to an unfairly biased trained model. We show that by imposing a loss on the canonical correlation between the network being trained (but we lack per-sample protected attribute information) and a well trained classifier pre-trained on the protected attributes, we can get a more fair model. This may enable training fairer models in settings which would otherwise be difficult.

Implementation details. To simulate the case where we only have a pretrained network on protected attributes, we train a Resnet-18 He et al. (2016) on *gender* attribute, and when we train the classifier

to predict *attractiveness*, we add a loss using the canonical correlations between these two networks on intermediate layers: $L_{\text{total}} = L_{\text{cross-entropy}} + L_{\text{CCA}}$ where the first term is the standard cross entropy term and the second term is the canonical correlation. See A.7 for more details of training/evaluation.

Results. We choose two commonly used error metrics for fairness: difference in Equality of Opportunity Hardt et al. (2016) (DEO), and difference in Demographic Parity Yao & Huang (2017) (DDP). See appendix A.6 for more detailed explaination of the two metrics. We conduct experiments by applying the canonical correlation loss on three different layers in Resnet-18. In Table 2, we can see that applying canonical correlation loss generally improves the DEO and DDP metrics (lower is better) over the standard model (trained using cross entropy loss only). Specifically, applying the loss on early layers like conv0 and conv1 gets better performance than applying at a relatively late layer like conv2. Another promising aspect of our approach is that is can easily handle the case where the protected attribute is a continuous variable (as long as a well trained regression network on the protected attribute is given) while other methods like Lokhande et al. (2020); Zhang et al. (2018) need to first discretize the variable and then enforce constraints which can be much more involved.

4 Related Work

Stochastic CCA: There has been much interest in designing scalable and provable algorithms for CCA: Ma et al. (2015) proposed the first stochastic algorithm for CCA, while only local convergence is proven for non-stochastic version. Wang et al. (2016) designed algorithm which uses alternating SVRG combined with shift-and-invert pre-conditioning, with global convergence. These stochastic methods, together with Ge et al. (2016) Allen-Zhu & Li (2016), which reduce CCA problem to generalized eigenvalue problem and solve it by performing efficient power method, all belongs to the methods that try to solve empirical CCA problem, it can be seen as ERM approxiamtion of the priginal population objective, which requires solving numerical optimization of the empirical CCA objective on a fixed data set. These methods usually assume the access to the full dataset in the beginning, which is not very suitable for many practical applications where data tend to come in a streaming way. Recently, there are increasingly interest in considering population CCA problem Arora et al. (2017) Gao et al. (2019). The main difficulty in population setting is we have limited knowledge about the objective unless we know the distribution of $\mathbf{X}$ and $\mathbf{Y}$. Arora et al. (2017) handles this problem by deriving an estimation of gradient of population objecitve whose error can be properly bounded so that applying proximal gradient to a convex relexed objective will provably converge. Gao et al. (2019) provides tightened analysis of the time complexity of the algorithm in Wang et al. (2016), and provides sample complexity under certain distribution. The problem we are trying to solve in this work is the same as that in Arora et al. (2017); Gao et al. (2019): to optimize the population objective of CCA in a streaming fashion.

Riemannian Optimization: Riemannian optimization is a generalization of standard Euclidean optimization methods to smooth manifolds, which takes the following form: Given $f : \mathcal{M} \to \mathbb{R}$, solve $\min_{x \in \mathcal{M}} f(x)$, where $\mathcal{M}$ is a Riemannian manifold. One advantage is that it provides a nice way to express many constrained optimization problems as unconstrained problems. Applications include matrix and tensor factorizationIshteva et al. (2011),Tan et al. (2014), PCAEdelman et al. (1998), CCAYger et al. (2012), and so on. Yger et al. (2012) rewrites CCA formulation as Riemannian optimization on Stiefel manifold. In our work, we further explore the ability of Riemannian optimization framework, decomposing the linear space spanned by canonical vectors into products of several matrices which lie in several different Riemannian manifolds.

5 Conclusions

In this work, we presented a stochastic approach (RSG+) for the CCA model based on the observation that the solution of CCA can be decomposed into a product of matrices which lie on certain structured spaces. This affords specialized numerical schemes and makes the optimization more efficient. The optimization is based on Riemannian stochastic gradient descent and we provide a proof for its $O(\frac{1}{t})$ convergence rate with $O(d^2k)$ time complexity per iteration. In experimental evaluations, we find that our RSG+ behaves favorably relative to the baseline stochastic CCA method in capturing the correlation in the datasets. We also show the use of RSG+ in the DeepCCA setting showing feasibility when scaling to large dimensions as well as in an interesting use case in training fair models.

REFERENCES

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. Riemannian geometry of grassmann manifolds with a view on algorithmic computation. *Acta Applicandae Mathematica*, 80(2):199–220, 2004.

Pierre-Antoine Absil, Robert E. Mahony, and Rodolphe Sepulchre. Optimization algorithms on matrix manifolds. 2007.

Zeyuan Allen-Zhu and Yuanzhi Li. Doubly accelerated methods for faster cca and generalized eigendecomposition. In *ICML*, 2016.

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International conference on machine learning*, pp. 1247–1255, 2013.

Raman Arora, Teodor Vanislavov Marinov, Poorya Mianjy, and Nati Srebro. Stochastic approximation for canonical correlation analysis. In *Advances in Neural Information Processing Systems*, pp. 4775–4784, 2017.

Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. *arXiv preprint arXiv:1810.00760*, 2018.

Kush Bhatia, Aldo Pacchiano, Nicolas Flammarion, Peter L Bartlett, and Michael I Jordan. Gen-oja: Simple & efficient algorithm for streaming generalized eigenvector computation. In *Advances in Neural Information Processing Systems*, pp. 7016–7025, 2018.

Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

William M Boothby. *An introduction to differentiable manifolds and Riemannian geometry*. Academic press, 1986.

Rudrasis Chakraborty, Liu Yang, Soren Hauberg, and Baba Vemuri. Intrinsic grassmann averages for online linear, robust and nonlinear subspace learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Kamalika Chaudhuri, Sham M Kakade, Karen Livescu, and Karthik Sridharan. Multi-view clustering via canonical correlation analysis. In *Proceedings of the 26th annual international conference on machine learning*, pp. 129–136, 2009.

Heather D. Couture, Roland Kwitt, J. S. Marron, Melissa A. Troester, Charles M. Perou, and Marc Niethammer. Deep multi-view learning via task-optimal cca. *ArXiv*, abs/1907.07739, 2019.

Michele Donini, Luca Oneto, Shai Ben-David, John S Shawe-Taylor, and Massimiliano Pontil. Empirical risk minimization under fairness constraints. In *Advances in Neural Information Processing Systems*, pp. 2791–2801, 2018.

Alan Edelman, Tomás A. Arias, and Steven Thomas Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20:303–353, 1998.

Chao Gao, Dan Garber, Nathan Srebro, Jialei Wang, and Weiran Wang. Stochastic canonical correlation analysis. *Journal of Machine Learning Research*, 20(167):1–46, 2019.

Rong Ge, Chi Jin, Praneeth Netrapalli, Aaron Sidford, et al. Efficient algorithms for large-scale generalized eigenvector computation and canonical correlation analysis. In *International Conference on Machine Learning*, pp. 2741–2750, 2016.

Gene H. Golub and Hongyuan Zha. The canonical correlations of matrix pairs and their numerical computation. 1992.

Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pp. 3315–3323, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Mariya Ishteva, Pierre-Antoine Absil, Sabine Van Huffel, and Lieven De Lathauwer. Best low multilinear rank approximation of higher-order tensors, based on the riemannian trust-region scheme. *SIAM J. Matrix Anal. Appl.*, 32:115–135, 2011.

Tetsuya Kaneko, Simone Fiori, and Toshihisa Tanaka. Empirical arithmetic averaging over the compact stiefel manifold. *IEEE Transactions on Signal Processing*, 61(4):883–894, 2012.

Hyunwoo J Kim, Nagesh Adluru, Barbara B Bendlin, Sterling C Johnson, Baba C Vemuri, and Vikas Singh. Canonical correlation analysis on Riemannian manifolds and its applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 251–267, Zürich, Switzerland, September 2014. URL `http://pages.cs.wisc.edu/~hwkim/projects/riem-cca/`.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Vishnu Suresh Lokhande, Aditya Kumar Akash, Sathya N Ravi, and Vikas Singh. Fairalm: Augmented lagrangian method for training fair models with little regret. *arXiv preprint arXiv:2004.01355*, 2020.

Yong Luo, Dacheng Tao, Kotagiri Ramamohanarao, Chao Xu, and Yonggang Wen. Tensor canonical correlation analysis for multi-view dimension reduction. *IEEE transactions on Knowledge and Data Engineering*, 27(11):3111–3124, 2015.

Zhuang Ma, Yichao Lu, and Dean P. Foster. Finding linear structure in large datasets with scalable canonical correlation analysis. In *ICML*, 2015.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems*, pp. 5727–5736, 2018.

Arkadi Nemirovski, Anatoli B. Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. Optimization*, 19:1574–1609, 2009.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

W Nicholson Price and I Glenn Cohen. Privacy in the age of medical big data. *Nature medicine*, 25 (1):37–43, 2019.

Markus Reiß, Martin Wahl, et al. Nonasymptotic upper bounds for the reconstruction error of pca. *Annals of Statistics*, 48(2):1098–1123, 2020.

Jan Rupnik and John Shawe-Taylor. Multi-view canonical correlation analysis. In *Conference on Data Mining and Data Warehouses (SiKDD 2010)*, pp. 1–4, 2010.

Cees GM Snoek, Marcel Worring, Jan C Van Gemert, Jan-Mark Geusebroek, and Arnold WM Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th ACM international conference on Multimedia*, pp. 421–430, 2006.

Raghav Subbarao and Peter Meer. Nonlinear mean shift over riemannian manifolds. *International journal of computer vision*, 84(1):1, 2009.

Mingkui Tan, Ivor Wai-Hung Tsang, Li Wang, Bart Vandereycken, and Sinno Jialin Pan. Riemannian pursuit for big matrix recovery. In *ICML*, 2014.

Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International Conference on Machine Learning*, pp. 1083–1092, 2015a.

Weiran Wang, Raman Arora, Karen Livescu, and Nathan Srebro. Stochastic optimization for deep cca via nonlinear orthogonal iterations. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 688–695. IEEE, 2015b.

Weiran Wang, Jialei Wang, Dan Garber, and Nati Srebro. Efficient globally convergent stochastic optimization for canonical correlation analysis. In *Advances in Neural Information Processing Systems*, pp. 766–774, 2016.

Sirui Yao and Bert Huang. Beyond parity: Fairness objectives for collaborative filtering. In *Advances in Neural Information Processing Systems*, pp. 2921–2930, 2017.

Florian Yger, Maxime Berar, Gilles Gasso, and Alain Rakotomamonjy. Adaptive canonical correlation analysis based on matrix manifolds. In *ICML*, 2012.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 335–340, 2018.

A APPENDIX

A.1 A BRIEF REVIEW OF RELEVANT DIFFERENTIAL GEOMETRY CONCEPTS

To make the paper self-contained, we briefly review certain differential geometry concepts. We only include a condensed description – needed for our algorithm and analysis – and refer the interested reader to Boothby (1986) for a comprehensive and rigorous treatment of the topic.

Riemannian Manifold: A Riemannian manifold, $\mathcal{M}$, (of dimension m) is defined as a (smooth) topological space which is locally diffeomorphic to the Euclidean space $\mathbf{R}^m$. Additionally, $\mathcal{M}$ is equipped with a Riemannian metric g which can be defined as $g_X : T_X\mathcal{M} \times T_X\mathcal{M} \to \mathbf{R}$, where $T_X\mathcal{M}$ is the tangent space at X of $\mathcal{M}$, see Fig. 2.

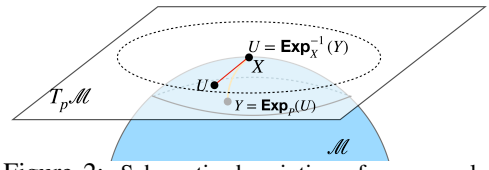


Figure 2: Schematic description of an exemplar manifold ($\mathcal{M}$) and the visual illustration of Exp and Exp^{-1} map.

If $X \in \mathcal{M}$, the Riemannian Exponential map at X, denoted by $\mathsf{Exp}_X : T_X\mathcal{M} \to \mathcal{M}$ is defined as $\gamma(1)$ where $\gamma : [0, 1] \to \mathcal{M}$. We can find γ by solving the following differential equation: $\gamma(0) = X, (\forall t_0 \in [0, 1]) \left.\frac{d\gamma}{dt}\right|_{t=t_0} = U$. In general Exp_X is not invertible but the inverse $\mathsf{Exp}_X^{-1} : \mathcal{U} \subset \mathcal{M} \to T_X\mathcal{M}$ is defined only if $\mathcal{U} = \mathcal{B}_r(X)$, where r is called the *injectivity radius* Boothby (1986) of $\mathcal{M}$. This concept will be useful to define the mechanics of gradient descent on the manifold.

In our reformulation, we will shortly make use of the following manifolds, specifically, when decomposing U and V into a product of several matrices. (a) $\mathsf{St}(p, n)$: the manifold consists of $n \times p$ column orthonormal matrices (b) $\mathsf{Gr}(p, n)$: the manifold consists of p-dimensional subspaces in $\mathbf{R}^n$ (c) $\mathsf{SO}(n)$, the manifold/group consists of $n \times n$ special orthogonal matrices. We covered the exponential map and the Riemannian metric above, and their explicit formulation for manifolds listed above is provided for easy reference in Table equation 3.

	$g_X(U, V)$	$\mathsf{Exp}_X(U)$	$\mathsf{Exp}_X^{-1}(Y)$
$\mathsf{St}(p, n)$ Kaneko et al. (2012)	trace $\left(U^T V\right)$	$\widetilde{U}\widetilde{V}^T$, $\widetilde{U}S\widetilde{V}^T = \text{svd}(X + U)$	$(Y - X) - X(Y - X)^T X$
$\mathsf{Gr}(p, n)$ Absil et al. (2004)	trace $\left(\Pi_*^{-1}(U)^T \Pi_*^{-1}(V)\right)$	$\widehat{U}\widehat{V}^T$, $\widehat{U}\widehat{S}\widehat{V}^T = \text{svd}(\bar{X} + U)$	$\bar{Y}\left(\bar{X}^T\bar{Y}\right)^{-1} - \bar{X}$, $X = \Pi(\bar{X}), Y = \Pi(\bar{Y})$
$\mathsf{SO}(n)$ Subbarao & Meer (2009)	trace $\left(X^T U X^T V\right)$	$X\mathsf{expm}\left(X^T U\right)$	$X\mathsf{logm}\left(X^T Y\right)$

Table 3: Explicit forms for some operations we need. $\Pi(X)$ returns X's column space; Π_* is Π's differential.

A.2 PROOF OF THEOREM 1

We first restate the assumptions from section 3.1:

Assumptions: **(a)** The random variables $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \Sigma_x)$ and $\mathbf{Y} \sim \mathcal{N}(\mathbf{0}, \Sigma_y)$ with $\Sigma_x \preceq cI_d$ and $\Sigma_y \preceq cI_d$ for some $c > 0$. **(b)** The samples X and Y drawn from $\mathcal{X}$ and $\mathcal{Y}$ respectively have zero mean. **(c)** For a given $k \leq d$, Σ_x and Σ_y have non-zero top-k eigen values.

Let F be the trace value solution for Eq. (2), and $\widetilde{F}$ be the trace value solution for Eqs. equation 3a, equation 3b, we next restate Theorem 1 and give its proof:

Theorem. *Under the assumptions and notations above, the approximation error $E = \|F - \widetilde{F}\|$ is bounded and goes to zero while the whitening constraints in equation 3b are satisfied.*

Proof. Let Q_u, S_u, Q_v, S_v be the solutions for Eqs. equation 3a and equation 3b, $\widetilde{U}, \widetilde{V}$ be matrices consisting of top-k eigen vectors of $(1/N)X^T X$ and $(1/N)Y^T Y$ respectively, U, V be solutions for Eq. equation 2. Let $\widetilde{X}_u = X\widetilde{U}Q_u S_u$ and $\widetilde{Y}_v = Y\widetilde{V}Q_v S_v$. Also let $X_u = XU$ and $Y_v = YV$. Observe that mean of $X_u, Y_v, \widetilde{X}_u$ and $\widetilde{Y}_v$ are zero. Moreover the sample covariance of X_u and Y_v are given by $U^T C_X U$ and $V^T C_Y V$ respectively. Thus by the constraint in Eq. equation 2, $X_u^T X_u = I_k$ and $Y_v^T Y_v = I_k$. Let these covariance matrices be denoted by $C(X_u)$ and $C(Y_v)$

respectively. Analogously the sample covariance of $\widetilde{X}_u$ and $\widetilde{Y}_v$ are given by $S_u^T Q_u^T \widetilde{U}^T C_X \widetilde{U} Q_u S_u$ and $S_v^T Q_v^T \widetilde{V}^T C_Y \widetilde{U} Q_v S_v$ respectively. Let these covariance matrices be denoted by $C(\widetilde{X}_u)$ and $C(\widetilde{Y}_v)$ respectively.

Using Proposition 1, we know X_u, X_v, $\widetilde{X}_u$ and $\widetilde{Y}_v$ follow sub-Gaussian distributions.

Let $F = \text{trace}\left(U^T C_{XY} V\right)$ which can be rewritten as $F = \text{trace}\left(X_u^T Y_v\right)$. Moreover, let $\widetilde{F} = \text{trace}\left(S_u^T Q_u^T \widetilde{U}^T C_{XY} \widetilde{V} Q_v S_v\right)$ which similarly can be rewritten by $\widetilde{F} = \text{trace}\left(\widetilde{X}_u^T \widetilde{Y}_v\right)$.

Consider the approximation error between the objective functions as $E = |F - \widetilde{F}|$. We can rewrite $E = |\text{trace}\left(X_u^T Y_v\right) - \text{trace}\left(\widetilde{X}_u^T \widetilde{Y}_v\right)|$. Due to von Neumann's trace inequality and CauchySchwarz inequality, we have

$$\begin{aligned} E &= |\text{trace}\left(\widetilde{X}_u^T \widetilde{Y}_v - X_u^T Y_v\right)| \\ &\leq |\text{trace}\left(\left(\widetilde{X}_u - X_u\right)^T \left(\widetilde{Y}_v - Y_v\right)\right)| \\ &\leq \sum_i \sigma_i(\widetilde{X}_u - X_u)\sigma_i(\widetilde{Y}_v - Y_v) \\ &\leq \| \left(\widetilde{X}_u - X_u\right) \|_F \| \left(\widetilde{Y}_v - Y_v\right) \|_F \end{aligned} \tag{A.1}$$

where σ_i(A) denote the i th singular value of matrix A and $\| \bullet \|_F$ donote frobenius norm.
Now, using Proposition 2, we get

$$\begin{aligned} \| \left(\widetilde{X}_u - X_u\right) \|_F &\leq \min\left(\sqrt{2k}\|\Delta_x\|_2, \frac{2\|\Delta_x\|_2^2}{\lambda_k^x - \lambda_{k+1}^x}\right) \\ \| \left(\widetilde{Y}_v - Y_v\right) \|_F &\leq \min\left(\sqrt{2k}\|\Delta_y\|_2, \frac{2\|\Delta_y\|_2^2}{\lambda_k^y - \lambda_{k+1}^y}\right) \end{aligned} \tag{A.2}$$

where, $\Delta_x = C(X_u) - C(\widetilde{X}_u)$, $\Delta_y = C(Y_v) - C(\widetilde{Y}_v)$. Here λ^xs and λ^ys are the eigen values of $C(X_u)$ and $C(Y_v)$ respectively. Now, assume that $C(X_u) = I_k$ and $C(Y_v) = I_k$ as X_u and Y_v are solutions of Eq. equation 2. Furthermore assume $\lambda_k^x - \lambda_{k+1}^x \geq \Lambda$ and $\lambda_k^y - \lambda_{k+1}^y \geq \Lambda$ for some $\Lambda > 0$. Then, we can rewrite Eq. equation A.1 as

$$E \leq \min\left(\sqrt{2k}\|I_k - C(\widetilde{X}_u)\|_2, \frac{2\|I_k - C(\widetilde{X}_u)\|_2^2}{\Lambda}\right) \min\left(\sqrt{2k}\|I_k - C(\widetilde{Y}_v)\|_2, \frac{2\|I_k - C(\widetilde{Y}_v)\|_2^2}{\Lambda}\right)$$

And as $C(\widetilde{X}_u) \to I_k$ or $C(\widetilde{Y}_v) \to I_k$, $E \to 0$. □

A.3 RSG+ ALGORITHM

Here we show our algorithm with more details about the gradients in every step in Alg.2.

A.4 IMPLEMENTATION DETAILS OF CCA ON FIXED DATASET

Implementation details. On all three benchmark datasets, we only passed the data once for both our RSG+ and MSG Arora et al. (2017) and we use the code from Arora et al. (2017) to produce MSG results. We conducted experiments on different dimensions of target space: $k = 1, 2, 4$. The choice of k is motivated by the fact that the spectrum of the datasets decays quickly. Since our RSG+ processes data in small blocks, we let data come in mini-batches (mini-batch size was set to 100).

A.5 RUNTIME OF RSG+ AND BASELINE METHODS

The runtime comparison of RSG+ and MSG is reported in Table 4. Our algorithm is 5–10 times faster.

Algorithm 2: Riemannian SGD based algorithm (RSG+) to compute canonical directions

Input: $X \in \mathbf{R}^{N \times d_x}$, $Y \in \mathbf{R}^{N \times d_y}$, $k > 0$
Output: $U \in \mathbf{R}^{d_x \times k}$, $V \in \mathbf{R}^{d_y \times k}$

Initialize $\widetilde{U}, \widetilde{V}, Q_u, Q_v, S_u, S_v$;

Partition data X, Y into batches of size B. Let j^{th} batch be denoted by X_j and Y_j ;

for $j \in \{1, \cdots, \lfloor \frac{N}{B} \rfloor\}$ **do**

Gradient for top-k principal vectors: calculating $\nabla_{\widetilde{U}} \widetilde{F}_{\text{pri}}, \nabla_{\widetilde{V}} \widetilde{F}_{\text{pri}}$

1. Partition X_j (Y_j) into L ($L = \lfloor \frac{B}{k} \rfloor$) blocks of size $d_x \times k$ ($d_y \times k$);
2. Let the lth block be denoted by Z_l^x (Z_l^y);
3. Orthogonalize each block and let the orthogonalized block be denoted by $\hat{Z}_l^x$ ($\hat{Z}_l^y$);
4. Let the subspace spanned by each $\hat{Z}_l^x$ (and $\hat{Z}_l^y$) be $\hat{\mathcal{Z}}_l^x \in \mathsf{Gr}(k, d_x)$ (and $\hat{\mathcal{Z}}_l^y \in \mathsf{Gr}(k, d_y)$);

$$\nabla_{\widetilde{U}} \widetilde{F}_{\text{pri}} = -\sum_l \mathsf{Exp}_{\widetilde{U}}^{-1}\left(\hat{\mathcal{Z}}_l^x\right) \quad \nabla_{\widetilde{V}} \widetilde{F}_{\text{pri}} = -\sum_l \mathsf{Exp}_{\widetilde{V}}^{-1}\left(\hat{\mathcal{Z}}_l^y\right) \tag{4}$$

Gradient from equation 3: calculating $\nabla_{\widetilde{U}} \widetilde{F}_{\text{can}}, \nabla_{\widetilde{V}} \widetilde{F}_{\text{can}}, \nabla_{Q_u} \widetilde{F}_{\text{can}}, \nabla_{Q_v} \widetilde{F}_{\text{can}}, \nabla_{S_u} \widetilde{F}_{\text{can}}, \nabla_{S_v} \widetilde{F}_{\text{can}}$

$\nabla_{\widetilde{U}} \widetilde{F}_{\text{can}} = \frac{\partial \widetilde{F}}{\partial \widetilde{U}} - \widetilde{U} \frac{\partial \widetilde{F}}{\partial \widetilde{U}}^T \widetilde{U}$ $\quad$ $\nabla_{\widetilde{V}} \widetilde{F}_{\text{can}} = \frac{\partial \widetilde{F}}{\partial \widetilde{V}} - \widetilde{V} \frac{\partial \widetilde{F}}{\partial \widetilde{V}}^T \widetilde{V}$;

$\nabla_{Q_u} \widetilde{F}_{\text{can}} = \frac{\partial \widetilde{F}}{\partial Q_u} - \frac{\partial \widetilde{F}}{\partial Q_u}^T$ $\quad$ $\nabla_{Q_v} \widetilde{F}_{\text{can}} = \frac{\partial \widetilde{F}}{\partial Q_v} - \frac{\partial \widetilde{F}}{\partial Q_v}^T$;

$\nabla_{S_u} \widetilde{F}_{\text{can}} = \text{Upper}\left(\frac{\partial \widetilde{F}}{\partial S_u}\right)$ $\quad$ $\nabla_{S_v} \widetilde{F}_{\text{can}} = \text{Upper}\left(\frac{\partial \widetilde{F}}{\partial S_v}\right)$;

Here, Upper returns the upper triangular matrix of the input matrix and $\frac{\partial \widetilde{F}}{\partial \widetilde{U}}, \frac{\partial \widetilde{F}}{\partial \widetilde{V}}, \frac{\partial \widetilde{F}}{\partial Q_u}, \frac{\partial \widetilde{F}}{\partial Q_v}, \frac{\partial \widetilde{F}}{\partial S_u}, \frac{\partial \widetilde{F}}{\partial S_v}$ give the Euclidean gradients. For completeness, the closed form expression of the gradients is,

$$\frac{\partial \widetilde{F}}{\partial \widetilde{U}} = -C_{XY} \widetilde{V} Q_v S_v Q_u^T S_u^T \qquad \frac{\partial \widetilde{F}}{\partial Q_u} = -\widetilde{U}^T C_{XY} \widetilde{V} Q_v S_v S_u^T \qquad \frac{\partial \widetilde{F}}{\partial S_u} = -Q_u^T \widetilde{U}^T C_{XY} \widetilde{V} Q_v S_v$$

$$\frac{\partial \widetilde{F}}{\partial \widetilde{V}} = -C_{YX} \widetilde{U} Q_u S_u Q_v^T S_v^T \qquad \frac{\partial \widetilde{F}}{\partial Q_v} = -\widetilde{V}^T C_{YX} \widetilde{U} Q_u S_u S_v^T \qquad \frac{\partial \widetilde{F}}{\partial S_v} = -S_u^T Q_u^T \widetilde{U}^T C_{XY} \widetilde{V} Q_v \tag{5}$$

Gradient to update canonical directions

$\nabla_{\widetilde{U}} \widetilde{F} = \nabla_{\widetilde{U}} \widetilde{F}_{\text{pri}} + \nabla_{\widetilde{U}} \widetilde{F}_{\text{can}}$ $\quad$ $\nabla_{\widetilde{V}} \widetilde{F} = \nabla_{\widetilde{V}} \widetilde{F}_{\text{pri}} + \nabla_{\widetilde{V}} \widetilde{F}_{\text{can}}$;

$\nabla_X \widetilde{F} = \nabla_X \widetilde{F}_{\text{can}}$ where, X is a generic entity: $X \in \{Q_u, Q_v, S_u, S_v\}$;

Batch update of canonical directions

$A = \mathsf{Exp}_A\left(-\gamma_j \nabla_A \widetilde{F}\right)$ where, A is a generic entity: $A \in \{\widetilde{U}, \widetilde{V}, Q_u, Q_v, S_u, S_v\}$;

end

$U = \widetilde{U} Q_u S_u$ and $V = \widetilde{V} Q_v S_v$;

Table 4: Wallclock runtime of one pass through the data of our RSG+ and MSG on MNIST, Mediamill and CIFAR (average of 5 runs).

	MNIST			Mediamill			CIFAR		
Time (s)	$k=1$	$k=2$	$k=4$	$k=1$	$k=2$	$k=4$	$k=1$	$k=2$	$k=4$
RSG+ (Ours)	4.16	4.24	4.71	1.89	1.60	1.44	14.80	17.22	22.10
MSG	35.32	42.09	49.17	11.59	14.21	17.34	80.21	100.80	106.55

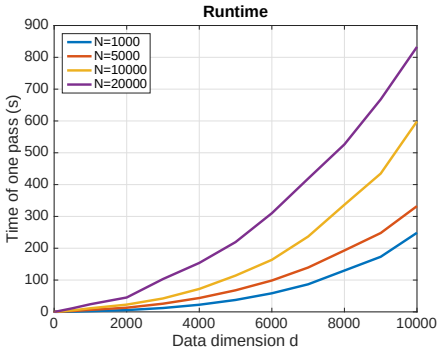


Figure 3: Runtime of RSG+ under different data dimensions and size of datasets.

We also plot the runtime of our algorithm under different data dimension (set $d_x = d_y = d$) and number of total samples sampled from joint gaussian distribution in Fig. 3.

A.6 Error metrics for fairness

Equality of Opportunity (EO) Hardt et al. (2016): A classifier h is said to satisfy EO if the prediction is independent of the protected attribute s (in our experiment s is a binary variable where $s = 1$ stands for *Male* and $s = 0$ stands for *Female*) for classification label $y \in \{0, 1\}$. We use the difference of false negative rate (conditioned on $y = 1$) across two groups identified by protected attribute s as the error metric, and we denote it as DEO.

Demographic Parity (DP) Yao & Huang (2017): A classifier h satisfies DP if the likelihodd of making a misclassification among the positive predictions of the classifier is independent of the protected attribute s. We denote the difference of demographic parity between two groups identified by the protected attribute as DDP.

A.7 Implementation details of fairness experiments

Implementation details. The network is trained for 20 epochs with learning rate 0.01 and batch size 256. We follow Donini et al. (2018) to use NVP (novel validation procedure) to evaluate our result: first we search for hyperparameters that achieves the highest classification score and then report the performance of the model which gets minimum fairness error metrics with accuracy within the highest 90% accuracies. When we apply our RSG+ on certain layers, we first use randomized projection to project the feature into 1k dimension, and then extract top-10 canonical components for training. Similar to our previous experiments on DeepCCA, the batch method does not scale to 1k dimension.

A.8 Resnet-18 architecture and position of Conv-0,1,2 in Table 2

The Resnet-18 contains a first convolutional layer followed by normalization, nonlinear activation, and max pooling. Then it has four residual blocks, followed by average polling and a fully connected layer. We denote the position after the first convolutional layer as conv0, the position after the first residual block as conv1 and the position after the second residual block as conv2. We choose early layers since late layers close to the final fully connected layer will have feature that is more directly relevant to the classification variable (*attractiveness* in this case).