Journal of Machine Learning Research 13 (2012) 429-458 Submitted 6/11; Revised 11/11; Published 2/12

Online Learning in the Embedded Manifold of Low-rank Matrices

Uri Shalit[*] URI.SHALIT@MAIL.HUJI.AC.IL
Computer Science Department and ICNC/ELSC
The Hebrew University of Jerusalem
91904 Jerusalem, Israel

Daphna Weinshall DAPHNA@CS.HUJI.AC.IL
Computer Science Department
The Hebrew University of Jerusalem
91904 Jerusalem, Israel

Gal Chechik[†] GAL@GOOGLE.COM
The Gonda Brain Research Center
Bar Ilan University
52900 Ramat-Gan, Israel

Editor: Léon Bottou

Abstract

When learning models that are represented in matrix forms, enforcing a low-rank constraint can dramatically improve the memory and run time complexity, while providing a natural regularization of the model. However, naive approaches to minimizing functions over the set of low-rank matrices are either prohibitively time consuming (repeated singular value decomposition of the matrix) or numerically unstable (optimizing a factored representation of the low-rank matrix). We build on recent advances in optimization over manifolds, and describe an iterative online learning procedure, consisting of a gradient step, followed by a *second-order retraction* back to the manifold. While the ideal retraction is costly to compute, and so is the projection operator that approximates it, we describe another retraction that can be computed efficiently. It has run time and memory complexity of $O((n+m)k)$ for a rank-k matrix of dimension $m \times n$, when using an online procedure with rank-one gradients. We use this algorithm, LORETA, to learn a matrix-form similarity measure over pairs of documents represented as high dimensional vectors. LORETA improves the mean average precision over a passive-aggressive approach in a factorized model, and also improves over a full model trained on pre-selected features using the same memory requirements. We further adapt LORETA to learn positive semi-definite low-rank matrices, providing an online algorithm for *low-rank metric learning*. LORETA also shows consistent improvement over standard weakly supervised methods in a large (1600 classes and 1 million images, using *ImageNet*) multi-label image classification task.

Keywords: low rank, Riemannian manifolds, metric learning, retractions, multitask learning, online learning

1. Introduction

Many learning problems involve models represented in matrix form. These include metric learning, collaborative filtering, and multi-task learning where all tasks operate over the same set of features.

∗. Also at The Gonda Brain Research Center, Bar Ilan University, 52900 Ramat-Gan, Israel.
†. Also at Google Research, 1600 Amphitheatre Parkway, Mountain View CA, 94043.

In many of these tasks, a natural way to regularize the model is to limit the rank of the corresponding matrix. In metric learning, a low-rank constraint allows to learn a low dimensional representation of the data in a discriminative way. In multi-task problems, low-rank constraints provide a way to tie together different tasks. In all cases, low-rank matrices can be represented in a factorized form that dramatically reduces the memory and run-time complexity of learning and inference with that model. Low-rank matrix models could therefore scale to handle substantially many more features and classes than models with full rank dense matrices.

Unfortunately, the rank constraint is non-convex, and in the general case, minimizing a convex function subject to a rank constraint is NP-hard (Natarajan, 1995).[1] As a result of these issues, two main approaches have been commonly used to address the problem of learning under a low-rank constraint. Sometimes, a matrix $W \in \mathbb{R}^{n\times m}$ of rank k is represented as a product of two low dimension matrices $W = AB^T, A \in \mathbb{R}^{n\times k}, B \in \mathbb{R}^{m\times k}$ and simple gradient descent techniques are applied to each of the product terms separately (Bai et al., 2009). Second, projected gradient algorithms can be applied by repeatedly taking a gradient step and projecting back to the manifold of low-rank matrices. Unfortunately, computing the projection to that manifold becomes prohibitively costly for large matrices and cannot be computed after every gradient step.

Work in the field has focused mostly on two realms. First, learning low-rank positive semi-definite (PSD) models (as opposed to general low-rank models), as in the works of Kulis et al. (2009) and Meyer et al. (2011). Second, approximating a noisy matrix of observations by a low-rank matrix, as in the work of Negahban and Wainwright (2010). This task is commonly addressed in the field of recommender systems. Importantly, the current paper does not address the problem of low-rank *approximation to a given data matrix*, but rather addresses the problem of learning a *low-rank parametric model* in the context of ranking and classification.

In this paper we propose new algorithms for online learning on the manifold of low-rank matrices. It is based on an operation called *retraction*, which is an operator that maps from a vector space that is tangent to the manifold, into the manifold (Do Carmo, 1992; Absil et al., 2008). Retractions include the projection operator as a special case, but also include other operators that can be computed substantially more efficiently. We use second order retractions to develop LORETA —an online algorithm for learning low-rank matrices. LORETA has a memory and run time complexity of $O((n+m)k)$ per update when the gradients have rank one. We show below that the case of rank-one gradients is relevant to numerous online learning problems.

We test LORETA in two different domains and learning tasks. First, we learn a bilinear similarity measure among pairs of text documents, where the number of features (text terms) representing each document could become very large. LORETA performed better than other techniques that operate on a factorized model, and also improves retrieval precision by 33% as compared with training a full rank model over pre-selected most informative features, using comparable memory footprint. Second, we applied LORETA to image multi-label ranking, a problem in which the number of classes could grow to millions. LORETA significantly improved over full rank models, using a fraction of the memory required. These two experiments suggest that low-rank optimization could become very useful for learning in high-dimensional problems.

1. Some special cases are solvable (notably, PCA), relying mainly on singular value decomposition (Fazel et al., 2005) and semi-definite programming techniques. For SDP of rank $k \geq 2$ it is not known whether this problem is NP-hard. For $k = 1$ it is equivalent to the MAX-CUT problem (Briët et al., 2010). Both SDP and SVD scale poorly to large scale tasks.

This paper is organized as follows. We start with an introduction to optimization on manifolds, describing the notion of retractions. We then derive our low-rank online learning algorithm in three variants: one which learns a general low-rank matrix, one which learns a low-rank PSD matrix, and one which concentrates most of the learning in a reduced dimensional space. Finally we test our algorithms in two applications: learning similarity of text documents, and multi-label ranking on a set of one million images.

This paper extends a shorter version published in Advances in Neural Information Systems (Shalit et al., 2010), by adding a new PSD version of the algorithm, much larger-scale and wider experiments, giving a full mathematical discussion and proofs, and adding thorough complexity analysis.

2. Optimization on Riemannian Manifolds

The field of numerical optimization on smooth manifolds has advanced significantly in the past few years. For a recent exposition on this subject see Absil et al. (2008). We start with a short introduction to embedded manifolds, which are the focus of this paper.

An *embedded manifold* is a smooth subset of an ambient space $\mathbb{R}^n$. For instance, the set $\{\mathbf{x} : ||\mathbf{x}||_2 = 1, \mathbf{x} \in \mathbb{R}^n\}$, the unit sphere, is an $n-1$ dimensional manifold embedded in n-dimensional space $\mathbb{R}^n$. As another example, the *orthogonal group* O_n, which comprises of the set of orthogonal $n \times n$ matrices, is an $\frac{n(n-1)}{2}$ dimensional manifold embedded in $\mathbb{R}^{n\times n}$. Here we focus on the manifold of *low-rank* matrices, namely, the set of $n \times m$ matrices of rank k where $k < m,n$. It is an $(n+m)k-k^2$ dimensional manifold embedded in $\mathbb{R}^{n\times m}$, which we denote $\mathcal{M}_k^{n,m}$, or plainly $\mathcal{M}$. Embedded manifolds inherit many properties from the ambient space, a fact which simplifies their analysis. For example, the natural Riemannian metric for embedded manifolds is simply the Euclidean metric restricted to the manifold.

Motivated by online learning, we focus here on developing a stochastic gradient descent procedure to minimize a loss function $\mathcal{L}$ over the manifold of low-rank matrices $\mathcal{M}_k^{n,m}$,

$$\min_W \quad \mathcal{L}(W) \qquad \text{s.t.} \quad W \in \mathcal{M}_k^{n,m} \quad .$$

To illustrate the challenge in this problem, consider a simple stochastic gradient descent algorithm (Figure 1). At every step t of the algorithm, a gradient step update $W^t - \tilde{\nabla}\mathcal{L}(W^t)$ takes the model outside of the manifold $\mathcal{M}$ and has to be mapped back onto the manifold. The most common mapping operation is the *projection* operation, which, given a point $W^t - \tilde{\nabla}\mathcal{L}(W^t)$ outside the manifold, would find the closest point in $\mathcal{M}$. Unfortunately, the projection operation is very expensive to compute for the manifold of low-rank matrices, since it basically involves a singular value decomposition. Here we describe a wider class of operations called *retractions*, that serve a similar purpose: they find a point on the manifold that is in the direction of the gradient. To explain how retractions are computed, we first describe the notion of a *tangent space* and the *Riemannian gradient* of a function on a manifold.

2.1 Riemannian Gradient and the Tangent Space

Each point W in an embedded manifold $\mathcal{M}$ has a tangent space associated with it, denoted $T_{\mathbf{W}}\mathcal{M}$, as shown in Figure 2 (for a formal definition of the tangent space, see Appendix A). The tangent space is a vector space of the same dimension as the manifold that can be identified in a natural way

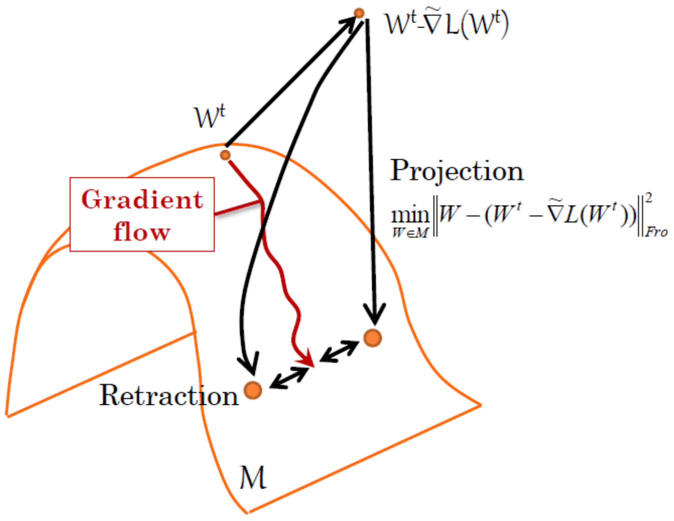
W^t-$\tilde{\nabla}L(W^t)$
W^t
Projection
$\min_{W\in M}\left\|W-(W^t-\tilde{\nabla}L(W^t))\right\|^2_{Fro}$
Gradient flow
Retraction
M

Figure 1: Projection onto the manifold is just a particular case of a retraction. Retractions are defined as operators that approximate the geodesic gradient flow on the manifold.

with a linear subspace of the ambient space. It is usually simple to compute the linear projection P_W of any point in the ambient space onto the tangent space $T_{\mathbf{W}}\mathcal{M}$.

Given a manifold $\mathcal{M}$ and a differentiable function $\mathcal{L}:\mathcal{M}\to\mathbb{R}$, the *Riemannian gradient* $\nabla\mathcal{L}(W)$ of $\mathcal{L}$ on $\mathcal{M}$ at a point $\mathbf{W}$ is a vector in the tangent space $T_{\mathbf{W}}\mathcal{M}$. A very useful property of embedded manifolds is the following: given a differentiable function f defined on the ambient space (and thus on the manifold), the Riemannian gradient of f at point W is simply the linear projection P_W of the Euclidean gradient of f onto the tangent space $T_W\mathcal{M}$.

Thus, if we denote the Euclidean gradient of $\mathcal{L}$ in $\mathbb{R}^{n\times m}$ by $\tilde{\nabla}\mathcal{L}$, we have $\nabla\mathcal{L}(W)=P_W(\tilde{\nabla}\mathcal{L})$. An important consequence follows in case the manifold represents the set of points obeying a certain constraint. In this case the Riemannian gradient of f is equivalent to the Euclidean gradient of f minus the component which is normal to the constraint. Indeed this normal component is exactly the component which is irrelevant when performing constrained optimization.

The Riemannian gradient allows us to compute $W^{t+\frac{1}{2}}=W^t-\eta^t\nabla\mathcal{L}(W)$, for a given iterate point W^t and step size η^t. We now examine how $W^{t+\frac{1}{2}}$ can be mapped back onto the manifold.

2.2 Retractions

Intuitively, *retractions* capture the notion of "going along a straight line" on the manifold. The mathematically ideal retraction is called the *exponential mapping* (Do Carmo, 1992, Chapter 3): it maps the tangent vector $\xi\in T_W\mathcal{M}$ to a point along a geodesic curve which goes through W in the direction of ξ Figure 1. Unfortunately, for many manifolds (including the low-rank manifold considered here) calculating the geodesic curve is computationally expensive (Vandereycken et al., 2009). A major insight from the field of Riemannian manifold optimization is that one can use retractions which merely approximate the exponential mapping. Using such retractions maintains the convergence properties obtained with the exponential mapping, but is much cheaper computationally for a suitable choice of mapping.

Definition 1 *Given a point W in an embedded manifold $\mathcal{M}$, a retraction is any function $R_W : T_{\mathbf{W}}\mathcal{M} \to \mathcal{M}$ which satisfies the following two conditions (Absil et al., 2008, Chapter 4):*

1. *Centering: $R_W(0) = W$.*

2. *Local rigidity: The curve $\gamma : (-\varepsilon, \varepsilon) \to \mathcal{M}$ defined by $\gamma_\xi(\tau) = R_W(\tau\xi)$ satisfies $\dot{\gamma}_\xi(0) = \xi$, where $\dot{\gamma}$ is the derivative of γ by τ.*

It can be shown that any such retraction approximates the exponential mapping to a first order (Absil et al., 2008). *Second-order retractions*, which approximate the exponential mapping to second order around W, have to satisfy in addition the following stricter condition:

$$P_W\left(\frac{\mathrm{d}R_W(\tau\xi)}{\mathrm{d}\tau^2}|_{\tau=0}\right) = 0,$$

for all $\xi \in T_W\mathcal{M}$, where P_W is the *linear* projection from the ambient space onto the tangent space $T_{\mathbf{W}}\mathcal{M}$. When viewed intrinsically, the curve $R_W(\tau\xi)$ defined by a second-order retraction has zero acceleration at point W, namely, its second order derivatives are all normal to the manifold. The best known example of a second-order retraction onto embedded manifolds is the projection operation (Absil and Malick, 2010), which maps a point X to the point $Y \in \mathcal{M}$ which is closest to it in the Frobenius norm. That is, the projection of X onto $\mathcal{M}$ is simply:

$$Proj_{\mathcal{M}}(X) = \underset{Y \in \mathcal{M}}{\operatorname{argmin}} \|X - Y\|_{Fro}$$

Importantly, such projections are viewed here as one type of a second order approximation to the exponential mapping, which can be replaced by any other second-order retractions, when computing the projection is too costly (see Figure 1).

Given the tangent space and a retraction, we now define a Riemannian gradient descent procedure for the loss $\mathcal{L}$ at point $W^t \in \mathcal{M}$. Conceptually, the procedure has three steps (Figure 2):

1. **Step 1: Ambient gradient:** Obtain the Euclidean gradient $\tilde{\nabla}\mathcal{L}(W^t)$ in the ambient space.

2. **Step 2: Riemannian gradient:** Linearly project the ambient gradient onto the tangent space $T_{\mathbf{W}}\mathcal{M}$. Compute $\xi^t = P_{W^t}(\tilde{\nabla}\mathcal{L}(W^t))$.

3. **Step 3: Retraction:** Retract the Riemannian gradient ξ^t back to the manifold: $W^{t+1} = R_{W^t}(\xi^t)$.

With a proper choice of step size, this procedure can be proved to have local convergence for any retraction (Absil et al., 2008). In practice, the algorithm merges these three steps for efficiency, as discussed in the next section.

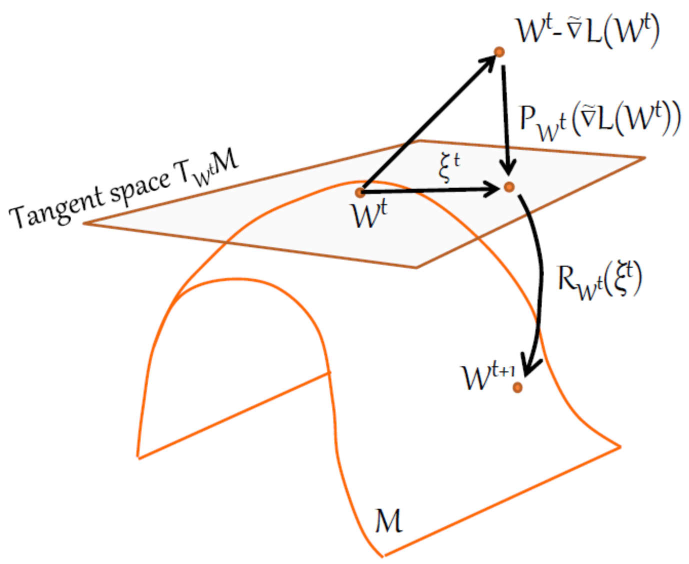

Figure 2: A three step procedure for computing a retracted gradient at point W^t. Step 1: standard (Euclidean) gradient step. Step 2: linearly project ambient gradient onto tangent space $T_{\mathbf{W}}\mathcal{M}$ in order to get the Riemannian gradient ξ^t. Step 3: retract the Riemannian gradient ξ^t back to the manifold.

3. Online Learning on the Low-rank Manifold

Based on the retractions described above, we now present an online algorithm for learning low-rank matrices, by performing stochastic gradient descent on the manifold of low-rank matrices. We name the algorithm LORETA (for a *LOw rank RETraction Algorithm*). At every iteration the algorithm suffers some loss, and performs a Riemannian gradient step followed by a retraction to the manifold $\mathcal{M}_k^{n,m}$. Section 3.1 discusses general online updates. Section 3.2 discusses the very common case where the online updates induce a gradient of rank $r = 1$.

Algorithm 1 : Online algorithm for learning in the manifold of low-rank matrices

Input: Initial low-rank model matrix $W^0 \in \mathcal{M}_k^{n,m}$. Examples $(x_0, x_1, \ldots)$. Loss function $\mathcal{L}$. Gradient descent step sizes $(\eta^0, \eta^1, \ldots)$.
Output: Final low-rank model matrix $W^{final} \in \mathcal{M}_k^{n,m}$.

```
repeat:
    Get example $x_t$
    Calculate the stochastic loss gradient: $\tilde{\nabla}\mathcal{L}(W^t; x_t)$
    Linearly project onto the tangent space: $\xi^t = P_{W^t}(\tilde{\nabla}\mathcal{L}(W^t; x_t))$
    Retract back to the manifold: $W^{t+1} = R_{W^t}(-\eta^t \xi^t)$
until stopping condition is satisfied
```

In what follows, lowercase Greek letters like ξ denote an abstract tangent vector, and uppercase Roman letters like A denote concrete matrix representations as kept in memory (taking $n \times m$ float numbers to store). We intermix the two notations, as in $\xi = AZ$, when the meaning is clear from the context. The set of $n \times k$ matrices of rank k is denoted $\mathbb{R}_*^{n \times k}$.

3.1 The General-Rank LORETA Algorithm

In online learning we are repeatedly given a rank-r gradient matrix $Z = \tilde{\nabla}\mathcal{L}W$, and want to compute a step on $\mathcal{M}_k^{n,m}$ in the direction of Z. As a first step we find its linear projection onto the tangent space $\hat{Z} = P_W(Z)$.

We start with a lemma that gives a representation of the tangent space $T_{\mathbf{W}}\mathcal{M}$ (Figure 2), extending the constructions given by Vandereycken and Vandewalle (2010) to the general manifold of low-rank matrices.

Lemma 2 *Let $W \in \mathcal{M}_k^{n,m}$ have a (non-unique) factorization $W = AB^T$, where $A \in \mathbb{R}_*^{n \times k}$, $B \in \mathbb{R}_*^{m \times k}$. Let $A_\perp \in \mathbb{R}^{n \times (n-k)}$ and $B_\perp \in \mathbb{R}^{m \times (m-k)}$ be the orthogonal complements of A and B respectively, such that $A_\perp^T A = 0$, $B_\perp^T B = 0$, $A_\perp^T A_\perp = I_{n-k}$, $B_\perp^T B_\perp = I_{m-k}$. The tangent space to $\mathcal{M}_k^{n,m}$ at W is:*

$$T_{\mathbf{W}}\mathcal{M} = \left\{ \begin{bmatrix} A & A_\perp \end{bmatrix} \begin{bmatrix} M & N_1^T \\ N_2 & 0 \end{bmatrix} \begin{bmatrix} B^T \\ B_\perp^T \end{bmatrix} : M \in \mathbb{R}^{k \times k}, N_1 \in \mathbb{R}^{(m-k) \times k}, N_2 \in \mathbb{R}^{(n-k) \times k} \right\}.$$

Proof The proof is given in Appendix A. ■

We note that the assumption that A and B are both of full column rank is tantamount to assuming that the model W is exactly of rank k, and no less. Let $\xi \in T_{\mathbf{W}}\mathcal{M}$ be a tangent vector to $W = AB^T$. From the characterization above it follows that ξ can be decomposed in a unique manner into three orthogonal components: $\xi = \xi^{AB} + \xi^{AB_\perp} + \xi^{A_\perp B}$, where:

$$\xi^{AB} = AMB^T, \quad \xi^{AB_\perp} = AN_1^T B_\perp^T, \quad \xi^{A_\perp B} = A_\perp N_2 B^T. \tag{1}$$

It is easy to verify that each pair is orthogonal, following from the relations $A_\perp^T A = 0$, $B_\perp^T B = 0$.

We wish to find the three matrices M, N_1 and N_2 associated with $\hat{Z} = P_W(Z)$, such that $\hat{Z} = AMB^T + AN_1^T B_\perp^T + A_\perp N_2 B^T$. We can find each of the matrices M, N_1 and N_2 separately, because each belongs to a space orthogonal to the other two. Thus we solve the following three problems:

$$\begin{aligned} &\arg\min_{M \in \mathbb{R}^{k \times k}} && \|Z - AMB^T\|_{Fro}^2, \\ &\arg\min_{N_1 \in \mathbb{R}^{(m-k) \times k}} && \|Z - AN_1^T B_\perp^T\|_{Fro}^2, \\ &\arg\min_{N_2 \in \mathbb{R}^{(n-k) \times k}} && \|Z - A_\perp N_2 B^T\|_{Fro}^2. \end{aligned}$$

To find the minimum of each of these three equations, we compute the derivatives and set them to zero. The solutions involve the pseudoinverses of A and B. Since A and B are of full column rank, their pseudoinverses are $A^\dagger = (A^T A)^{-1} A^T$, $B^\dagger = (B^T B)^{-1} B^T$.

$$\begin{aligned} M &= (A^TA)^{-1}A^TZB(B^TB)^{-1} = A^{\dagger}ZB^{\dagger^T}, \\ N_1 &= B_{\perp}^TZ^TA(A^TA)^{-1} = B_{\perp}^TZ^TA^{\dagger}, \\ N_2 &= A_{\perp}^TZB(B^TB)^{-1} = A_{\perp}^TZB^{\dagger^T}. \end{aligned} \tag{2}$$

The matrix $AA^{\dagger}$ is the matrix projecting onto the column space of A, and similarly for B. We will denote these matrices by P_A, P_B, etc. For the matrices projecting onto $A_{\perp}$ and $B_{\perp}$'s columns we actually have $P_{A_{\perp}} = A_{\perp}A_{\perp}^T$ because the columns of $A_{\perp}$ are orthogonal, and likewise for $P_{B_{\perp}}$. Substituting the expressions in $Equation$ (2) into expressions of the components of the Riemannian gradient vector in $Equation$ (1), we obtain:

$$\xi^{AB} = P_AZP_B, \quad \xi^{AB_{\perp}} = P_AZP_{B_{\perp}}, \quad \xi^{A_{\perp}B} = P_{A_{\perp}}ZP_B.$$

We can now define the retraction. The following theorem presents the retraction we will apply.

Theorem 3 *Let $W \in \mathcal{M}_k^{n,m}$, $W = AB^T$, and $W^{\dagger} = B^{\dagger T}A^{\dagger}$. Let $\xi \in T_W\mathcal{M}_k^{n,m}$, $\xi = \xi^{AB} + \xi^{AB_{\perp}} + \xi^{A_{\perp}B}$, as in $Equation$ (1), and let:*

$$\begin{aligned} V_1 &= W + \frac{1}{2}\xi^{AB} + \xi^{A_{\perp}B} - \frac{1}{8}\xi^{AB}W^{\dagger}\xi^{AB} - \frac{1}{2}\xi^{A_{\perp}B}W^{\dagger}\xi^{AB} \quad , \\ V_2 &= W + \frac{1}{2}\xi^{AB} + \xi^{AB_{\perp}} - \frac{1}{8}\xi^{AB}W^{\dagger}\xi^{AB} - \frac{1}{2}\xi^{AB}W^{\dagger}\xi^{AB_{\perp}} \quad . \end{aligned}$$

The mapping

$$R_W(\xi) = V_1W^{\dagger}V_2$$

is a second order retraction from a neighborhood $\Theta_W \subset T_W\mathcal{M}_k^{n,m}$ to $\mathcal{M}_k^{n,m}$.

Proof The proof is given in Appendix B. ■

A more succinct representation of this retraction is the following:

Lemma 4 *The retraction $R_W(\xi)$ can be presented as:*

$$\begin{aligned} R_W(\xi) = &\left[A\left(I_k + \frac{1}{2}M - \frac{1}{8}M^2\right) + A_{\perp}N_2\left(I_k - \frac{1}{2}M\right)\right] \cdot \\ &\left[B\left(I_k + \frac{1}{2}M^T - \frac{1}{8}\left(M^T\right)^2\right) + B_{\perp}N_1\left(I_k - \frac{1}{2}M^T\right)\right]^T. \end{aligned}$$

Proof The proof is given in Appendix C. ■

As a result from Lemma 4, we can calculate the retraction as the product of two low-rank factors: the first is an $n \times k$ matrix, the second a $k \times m$ matrix. Given a gradient $\tilde{\nabla}\mathcal{L}(x)$ in the ambient space, we can calculate the matrices M, N_1 and N_2 which allow us to represent its projection onto the tangent space, and furthermore allow us to calculate the retraction. We now have all the ingredients

Algorithm 2 : Naive Riemannian stochastic gradient descent

Input: Matrices $A \in \mathbb{R}_*^{n\times k}$, $B \in \mathbb{R}_*^{m\times k}$ s.t. $W = AB^T$. Gradient matrix $G \in \mathbb{R}^{n\times m}$ s.t. $G = -\eta\tilde{\nabla}\mathcal{L}(W) \in \mathbb{R}^{n\times m}$, where $\tilde{\nabla}\mathcal{L}(W)$ is the gradient in the ambient space and $\eta > 0$ is the step size.

Output: Matrices $Z_1 \in \mathbb{R}_*^{n\times k}$, $Z_2 \in \mathbb{R}_*^{m\times k}$ such that $Z_1 Z_2^T = R_W(-\eta\nabla\mathcal{L}(W))$.

Compute:

	matrix dimension
$A^\dagger = (A^TA)^{-1}A^T$, $B^\dagger = (B^TB)^{-1}B^T$	$k\times n$, $k\times m$
$A_\perp$, $B_\perp$= orthogonal complements of A, B	$n\times(n-k)$, $m\times(m-k)$
$M = A^\dagger G B^{\dagger T}$	$k\times k$
$N_1 = B_\perp^T G^T A^{\dagger T}$	$(m-k)\times k$
$N_2 = A_\perp^T G B^{\dagger T}$	$(n-k)\times k$
$Z_1 = A\left(I_k + \frac{1}{2}M - \frac{1}{8}M^2\right) + A_\perp N_2\left(I_k - \frac{1}{2}M\right)$	$n\times k$
$Z_2 = B\left(I_k + \frac{1}{2}M^T - \frac{1}{8}(M^T)^2\right) + B_\perp N_1\left(I_k - \frac{1}{2}M^T\right)$	$m\times k$

necessary for a Riemannian stochastic gradient descent algorithm. The procedure is outlined in Algorithm 2.

Algorithm 2 explicitly computes and stores the orthogonal complement matrices $A_\perp$ and $B_\perp$, which in the low rank case $k \ll m, n$, have size $O(mn)$, the same as the full sized W. To improve the memory complexity, we use the fact that the matrices $A_\perp$ and $B_\perp$ always operate with their transpose. Since $A_\perp$ and $B_\perp$ have orthogonal columns, the matrix $A_\perp A_\perp^T$ is actually the projection matrix that we denoted earlier by $P_{A_\perp}$, and likewise for $B_\perp$. Because of orthogonal complementarity, these projection matrices are equal to $I_n - P_A$ and $I_m - P_B$ respectively. Thus we can write $A_\perp N_2 = \left(I - AA^\dagger\right) Z B^{\dagger^T}$, and a similar identity for $B_\perp N_1$.

Consider now the case where the gradient matrix is of rank-r and is available in a factorized form $Z = G_1 G_2^T$, with $G_1 \in \mathbb{R}^{n\times r}$, $G_2 \in \mathbb{R}^{m\times r}$. Using the factorized gradient we can reformulate the algorithm to keep in memory only matrices of size at most $\max(n,m)\times k$ or $\max(n,m)\times r$. Optimizing the order of matrix operations so that the number of operations is minimized gives Algorithm 3. The runtime complexity of Algorithm 3 is readily computed based on matrix multiplications complexity,[2] and is $O\left((n+m)(k+r)^2\right)$.

3.2 LORETA With Rank-one Gradients

In many learning problems, the gradient matrix $\tilde{\nabla}\mathcal{L}(W)$ required for a gradient step update has a rank of one. This is the case for example, when the matrix model W acts as a bilinear form on two vectors, p and q, and the loss is a piecewise linear function of $\mathbf{p}^T W \mathbf{q}$ (as in Grangier and Bengio, 2008; Chechik et al., 2010; Weinberger and Saul, 2009; Shalev-Shwartz et al., 2004 and Section 7.1 below). In that case, the gradient is the rank-one outer product matrix $\mathbf{p}\mathbf{q}^T$. As another example, consider the case of multitask learning, where the matrix model W operates on a vector input $\mathbf{p}$, and the loss is the squared loss $\|W\mathbf{p} - \mathbf{q}\|^2$ between the multiple predictions $W\mathbf{p}$ and the true labels $\mathbf{q}$. The gradient of this loss is $(W\mathbf{p} - \mathbf{q})\,\mathbf{p}^T$, which is again a rank-one matrix. We now show how to

2. We assume throughout this paper the use of ordinary (schoolbook) matrix multiplication.

Algorithm 3 : **LORETA-r -** General-rank Riemannian stochastic gradient descent

Input: Matrices $A \in \mathbb{R}_*^{n\times k}$, $B \in \mathbb{R}_*^{m\times k}$ s.t. $W = AB^T$. Matrices $G_1 \in \mathbb{R}^{n\times r}$, $G_2 \in \mathbb{R}^{m\times r}$ s.t. $G_1G_2^T = -\eta\tilde{\nabla}\mathcal{L}(W) \in \mathbb{R}^{n\times m}$, where $\tilde{\nabla}\mathcal{L}(W)$ is the gradient in the ambient space and $\eta > 0$ is the step size.

Output: Matrices $Z_1 \in \mathbb{R}_*^{n\times k}$, $Z_2 \in \mathbb{R}_*^{m\times k}$ such that $Z_1Z_2^T = R_W(-\eta\nabla\mathcal{L}(W))$.

Compute:	matrix dimension	runtime complexity
$A^\dagger = (A^TA)^{-1}A^T, \quad B^\dagger = (B^TB)^{-1}B^T$	$k\times n, \quad k\times m$	$O((n+m)k^2)$
$\mathbf{a_1} = A^\dagger \cdot G_1, \quad \mathbf{b_1} = B^\dagger \cdot G_2$	$k\times r, \quad k\times r$	$O((n+m)kr)$
$\mathbf{a_2} = A\cdot\mathbf{a_1}$	$n\times r$	$O(nkr)$
$Q = \mathbf{b_1}^T \cdot \mathbf{a_1}$	$r\times r$	$O(kr^2)$
$\mathbf{a_3} = -\frac{1}{2}\mathbf{a_2} + \frac{3}{8}\mathbf{a_2}\cdot Q + G_1 - \frac{1}{2}G_1\cdot Q$	$n\times r$	$O(nr^2)$
$Z_1 = A + \mathbf{a_3}\cdot\mathbf{b_1}^T$	$n\times k$	$O(nkr)$
$\mathbf{b_2} = \left(G_2^TB\right)\cdot B^\dagger$	$r\times m$	$O(mkr)$
$\mathbf{b_3} = -\frac{1}{2}\mathbf{b_2} + \frac{3}{8}Q\cdot\mathbf{b_2} + G_2^T - \frac{1}{2}Q\cdot G_2^T$	$r\times m$	$O(mr^2)$
$Z_2^T = B^T + \mathbf{a_1}\cdot\mathbf{b_3}$	$k\times m$	$O(mkr)$

reduce the complexity of each iteration to be linear in the model rank k when the rank of the gradient matrix is $r = 1$.

Algorithm 4 : **LORETA-1 -** Rank-one Riemannian stochastic gradient descent

Input: Matrices $A \in \mathbb{R}_*^{n\times k}$, $B \in \mathbb{R}_*^{m\times k}$ s.t. $W = AB^T$. Matrices $A^\dagger$ and $B^\dagger$, the pseudo-inverses of A and B respectively. Vectors $\mathbf{p} \in \mathbb{R}^{n\times 1}$, $\mathbf{q} \in \mathbb{R}^{m\times 1}$ s.t. $\mathbf{pq}^T = -\eta\tilde{\nabla}\mathcal{L}(W) \in \mathbb{R}^{n\times m}$, where $\tilde{\nabla}\mathcal{L}(W)$ is the gradient in the ambient space and $\eta > 0$ is the step size.

Output: Matrices $Z_1 \in \mathbb{R}_*^{n\times k}$, $Z_2 \in \mathbb{R}_*^{m\times k}$ s.t. $Z_1Z_2^T = R_W(-\eta\nabla\mathcal{L}(W))$. Matrices $Z_1^\dagger$ and $Z_2^\dagger$, the pseudo-inverses of Z_1 and Z_2 respectively.

Compute:	matrix dimension	runtime complexity
$\mathbf{a_1} = A^\dagger\cdot\mathbf{p}, \mathbf{b_1} = B^\dagger\cdot\mathbf{q}$	$k\times 1$	$O((n+m)k)$
$\mathbf{a_2} = A\cdot\mathbf{a_1}$	$n\times 1$	$O(nk)$
$s = \mathbf{b_1}^T\cdot\mathbf{a_1}$	1×1	$O(k)$
$\mathbf{a_3} = \mathbf{a_2}\left(-\frac{1}{2}+\frac{3}{8}s\right) + \mathbf{p}(1-\frac{1}{2}s)$	$n\times 1$	$O(n)$
$Z_1 = A + \mathbf{a_3}\cdot\mathbf{b_1}^T$	$n\times k$	$O(nk)$
$\mathbf{b_2} = \left(\mathbf{q}^TB\right)\cdot B^\dagger$	$1\times m$	$O(mk)$
$\mathbf{b_3} = \mathbf{b_2}\left(-\frac{1}{2}+\frac{3}{8}s\right) + \mathbf{q}^T(1-\frac{1}{2}s)$	$1\times m$	$O(m)$
$Z_2^T = B^T + \mathbf{a_1}\cdot\mathbf{b_3}$	$k\times m$	$O(mk)$
$Z_1^\dagger = rank\_one\_pseudoinverse\_update(A, A^\dagger, \mathbf{a_3}, \mathbf{b_1})$	$k\times n$	$O(nk)$
$Z_2^\dagger = rank\_one\_pseudoinverse\_update(B, B^\dagger, \mathbf{b_3}, \mathbf{a_1})$	$k\times m$	$O(mk)$

Given rank-one gradients, the most computationally demanding step in Algorithm 3 is the computation of the pseudo-inverse of the matrices A and B, taking $O(nk^2)$ and $O(mk^2)$ operations. All other operations are $O(\max(n,m)\cdot k)$ at most. To speed up calculations we use the fact that for

$r = 1$ the outputs Z_1 and Z_2 become rank-one updates of the input matrices A and B. This enables us to keep the pseudo-inverses $A^\dagger$ and $B^\dagger$ from the previous round, and perform a rank-one update to them, following a procedure developed by Meyer (1973). The full procedure is included in Appendix D. This procedure is similar to the better known Sherman-Morrison formula for the inverse of a rank-one perturbed matrix, and its computational complexity for an $n \times k$ matrix is $O(nk)$ operations. Using that procedure, we derive our final algorithm, LORETA-*1*, the rank-one Riemannian stochastic gradient descent. Its overall time and space complexity are both $O((n+m)k)$ per gradient step. It can be seen that the LORETA-*1* algorithm uses only basic matrix operations, with the most expensive ones being low-rank matrix-vector multiplication and low-rank matrix-matrix addition. The memory requirement of LORETA-1 is about $4nk$ (assuming $m = n$), since it receives four input matrices of size nk ($A, B, A^\dagger, B^\dagger$) and assuming it can compute the four outputs ($Z_1, Z_2, Z_1^\dagger, Z_2^\dagger$), in-place while destroying previously computed terms.

4. Online Learning of Low-rank Positive Semidefinite Matrices

In this section we adapt the derivation above to the special case of positive semidefinite (PSD) matrices. PSD matrices are of special interest because they encode a true Euclidean metric. An n-by-n PSD matrix W of rank-k can be factored as $W = YY^T$, with $Y \in \mathbb{R}^{n\times k}$. Thus, the bilinear form $x^T Wz$ is equal to $(Yx)^T(Yz)$, which is a Euclidean inner product in the space spanned by Y's columns. These properties have led to an extensive use of PSD matrix models in metric and similarity learning, see, for example, Xing et al. (2002), Goldberger et al. (2005), Globerson and Roweis (2006), Bar-Hillel et al. (2006) and Jain et al. (2008). The set of n-by-n PSD matrices of rank-k forms a manifold of dimension $nk - \frac{k(k-1)}{2}$, embedded in the Euclidean space $\mathbb{R}^{n\times n}$ (Vandereycken et al., 2009). We denote this manifold by $\mathcal{S}_+(k,n)$.

We now give a characterization of the tangent space of this manifold, due to Vandereycken and Vandewalle (2010).

Lemma 5 *Let $W \in \mathcal{S}_+(k,n)$ have a (non-unique) factorization $W = YY^T$, where $Y \in \mathbb{R}_*^{n\times k}$. Let $Y_\perp \in \mathbb{R}^{n\times(n-k)}$ be the orthogonal complement of Y such that $Y_\perp^T Y = 0$, $Y_\perp^T Y_\perp = I_{n-k}$. The tangent space to $\mathcal{S}_+(k,n)$ at W is:*

$$T_W\mathcal{S}_+(k,n) = \left\{ \begin{bmatrix} Y & Y_\perp \end{bmatrix} \begin{bmatrix} S & N^T \\ N & 0 \end{bmatrix} \begin{bmatrix} Y^T \\ Y_\perp^T \end{bmatrix} : S \in \mathbb{R}^{k\times k}, N \in \mathbb{R}^{(n-k)\times k}, S = S^T \right\}.$$

Proof See Vandereycken and Vandewalle (2010), Proposition 5.2. ■

Let $\xi \in T_W\mathcal{S}_+(k,n)$ be a tangent vector to $W = YY^T$. As shown by Vandereycken and Vandewalle (2010), ξ can be decomposed into two orthogonal components, $\xi = \xi^S + \xi^P$. Given a rank-r gradient matrix Z, and using the projection matrices P_Y and $P_{Y_\perp}$ they show that:

$$\xi^S = P_Y \frac{Z+Z^T}{2} P_Y,$$
$$\xi^P = P_{Y_\perp} \frac{Z+Z^T}{2} P_Y + P_Y \frac{Z+Z^T}{2} P_{Y_\perp}.$$

Using this characterization of the tangent vector when given an ambient gradient Z, one can define a retraction analogous to that defined in Section 3. This retraction is referred to as $R_W^{(2)}$ in Vandereycken and Vandewalle (2010).

Theorem 6 *Let $W \in \mathcal{S}_+(k,n)$, $W = YY^T$, and $W^\dagger$ be its pseudo-inverse. Let $\xi \in T_W\mathcal{S}_+(k,n)$, $\xi = \xi^S + \xi^P$, as described above, and let*

$$V = W + \frac{1}{2}\xi^S + \xi^P - \frac{1}{8}\xi^S W^\dagger \xi^S - \frac{1}{2}\xi^P W^\dagger \xi^S.$$

The mapping $R_W^{PSD}(\xi) = VW^\dagger V$ is a second order retraction from a neighborhood $\Theta_W \subset T_W\mathcal{S}_+(k,n)$ to $\mathcal{S}_+(k,n)$.

Proof See Vandereycken and Vandewalle (2010), Proposition 5.10. ■

Algorithm 5 : **LORETA-1-PSD** - Rank-one Riemannian PSD stochastic gradient descent

Input: A matrix $Y \in \mathbb{R}_*^{n\times k}$, s.t. $W = YY^T$. The matrix $Y^\dagger$, the pseudoinverse of Y. Vectors $\mathbf{p} \in \mathbb{R}^{n\times 1}$, $\mathbf{q} \in \mathbb{R}^{n\times 1}$ s.t. $pq^T = -\eta\tilde{\nabla}\mathcal{L}(W) \in \mathbb{R}^{n\times m}$, where $\tilde{\nabla}\mathcal{L}(W)$ is the gradient in the ambient space and $\eta > 0$ is the step size.

Output: Matrix $Z \in \mathbb{R}_*^{n\times k}$, s.t. $ZZ^T = R_W^{PSD}(-\eta\nabla\mathcal{L}(W))$. Matrix $Z^\dagger$, the pseudo-inverse of Z.

Compute:	matrix dimension	runtime complexity
$\mathbf{h_1} = Y^\dagger \mathbf{p}$	$k\times 1$	$O(nk)$
$\mathbf{h_2} = Y^\dagger \mathbf{q}$	$k\times 1$	$O(nk)$
$n_1 = \mathbf{h_1}^T\mathbf{h_1}$	1×1	$O(k)$
$n_2 = \mathbf{h_2}^T\mathbf{h_2}$	1×1	$O(k)$
$\hat{\mathbf{h_1}} = Y\mathbf{h_1}$	$n\times 1$	$O(nk)$
$\hat{\mathbf{h_2}} = Y\mathbf{h_2}$	$n\times 1$	$O(nk)$
$s = \mathbf{h_1}^T\mathbf{h_2}$	1×1	$O(k)$
$\mathbf{l_1} = (-\frac{1}{4} + \frac{3}{32}s)\hat{\mathbf{h_1}} + (\frac{1}{2} - \frac{1}{8}s)\mathbf{p} + \frac{3}{32}n_1\hat{\mathbf{h_2}} - \frac{1}{8}n_1\mathbf{q}$	$n\times 1$	$O(n)$
$\mathbf{l_2} = (-\frac{1}{4} + \frac{3}{32}s)\hat{\mathbf{h_2}} + (\frac{1}{2} - \frac{1}{8}s)\mathbf{q} + \frac{3}{32}n_2\hat{\mathbf{h_1}} - \frac{1}{8}n_2\mathbf{p}$	$n\times 1$	$O(n)$
$P_1 = \mathbf{l_1}\mathbf{h_2}^T$	$n\times k$	$O(nk)$
$P_2 = \mathbf{l_2}\mathbf{h_1}^T$	$n\times k$	$O(nk)$
$Z = Y + P_1 + P_2$	$n\times k$	$O(nk)$
$Z_{temp}^\dagger = rank\_one\_pseudoinverse\_update(Y, Y^\dagger, \mathbf{l_1}, \mathbf{h_2})$	$k\times n$	$O(nk)$
$Z^\dagger = rank\_one\_pseudoinverse\_update(Y + P_1, Z_{temp}^\dagger, \mathbf{l_2}, \mathbf{h_1})$	$k\times n$	$O(nk)$

Following the derivation of algorithms 2-4, and after some rearrangement, we obtain a PSD version of the LORETA-*1* algorithm. This PSD version is given in Algorithm (5). The algorithm is very similar to LORETA-*1* , but instead of learning a general rank-k matrix it learns a positive semidefinite rank-k matrix. The computational complexity and memory complexity of a gradient step for this algorithm is $O(nk)$, namely, it is linear in the reduced number of model parameters.

5. Manifold Identification

Until now, we formalized the problem of learning a low-rank matrix based on a factorization $W = AB$. At test time, computing the bilinear score using the model can be even faster when the data is sparse. For instance, given two vectors x_1 and x_2 with c_1 and c_2 non-zero values, computing the bilinear form $x_1^T AB^T x_2$ requires $O(c_1 k + k + kc_2) = O((c_1 + c_2)k)$ operations, and can be significantly faster than the dense case. However, at training time, the LORETA-1 algorithm still has a complexity of $O((m+n)k)$ for each iteration even when the data is sparse.

The current section describes an attempt to adapt LORETA-1 such that it treats sparse data more efficiently. The empirical evaluation of this adaptation showed mixed results, but we include the derivation for completeness. The main idea is to separate the low-rank projection into two steps. First, a projection to a low dimensional space Ax that can be computed efficiently when x is sparse. Then, learning a second matrix, whose role is to tune the representation in the k-dimensional space.

To explain the idea, we focus on the case of learning a low-rank model which parametrizes a similarity function. The model is $W = AB^T$, $A \in \mathbb{R}^{n\times k}$, $B \in \mathbb{R}^{n\times k}$. The similarity between two vectors $\mathbf{p}, \mathbf{q} \in \mathbb{R}^n$ is then given by

$$Sim(\mathbf{p},\mathbf{q}) = \mathbf{p}^T W \mathbf{q} = (A^T\mathbf{p})^T \cdot (B^T\mathbf{q}). \tag{3}$$

This similarity measure can be viewed as the cosine similarity in $\mathbb{R}^k$ between the projected vectors $B^T\mathbf{q}$ and $A^T\mathbf{p}$. We now introduce another similarity model which operates directly in the projected space. Formally, we have $M \in \mathbb{R}^{k\times k}$, and the similarity model is

$$Sim(\mathbf{p},\mathbf{q}) = (A^T\mathbf{p})^T M (B^T\mathbf{q}) = \mathbf{p}^T AMB^T\mathbf{q}. \tag{4}$$

Clearly, since the model in Equation (4) involves only linear matrix multiplications, its descriptive power is equivalent to that of the model Equation (3). However, it has the potential to be learned faster. To speed the training we can iterate between learning the outer projections A,B using LORETA , and learning the inner low-dimensional similarity model M using standard methods operating in the low-dimensional space. Specifically, the idea is to execute s update steps of M for every update step of A,B (Algorithm 6). After s update steps to M, it is decomposed using SVD to obtain $M = USV^T$, and these factors are used to update the outer projections using $A \leftarrow AU sqrt(S)$, $B \leftarrow BV sqrt(S)$.

Consider the computational complexity: Given two sparse vectors x_1, x_2 with c_1 and c_2 non-zero values respectively, projecting them using A and B to the low dimensional space takes $O(k(c_1+c_2))$, and an update step of M takes $O(k^2)$. Decomposing M using SVD takes $O(k^3)$, so the overall complexity for s updates is $O\left(k \cdot \left(s(k+c_1+c_2)+k^2\right)\right)$. When $s \geq k$ the cost of decomposition is amortized across many M updates and does not increase the overall complexity. The update of A, B takes $O(k(n+m))$ as before. This approach is related to the idea of manifold identification (Oberlin and Wright, 2007), where the learning of A, B "identifies" a manifold of rank k and the inner steps operate to tune the representation within that subspace.

This iterative procedure could be a significant speed up compared to the original $O((m+n)k)$. Unfortunately, when we tested this algorithm in a similarity learning task (as in Section 7.1), its performance was not as good as that of LORETA-1. The main reason was numerical instability: The matrix M typically collapsed to match few directions in A, and decomposing it has amplified the same A directions. This approach awaits deeper investigation which is outside the scope of the current paper.

6. Related Work

A recent summary of many advances in the field of optimization on manifolds is given by Absil et al. (2008). Advances in this field have lately been applied to matrix completion (Keshavan et al.,

Algorithm 6 : Manifold identification meta-algorithm

Input: Initial model matrices $A \in \mathbb{R}_*^{n\times k}$, $B \in \mathbb{R}_*^{m\times k}$ s.t. $W = AB^T$. Matrices $A^\dagger$ and $B^\dagger$, the pseudo-inverses of A and B respectively. Loss function $\mathcal{L}$.
Output: Matrices $A \in \mathbb{R}_*^{n\times k}$, $B \in \mathbb{R}_*^{m\times k}$ s.t. $W = AB^T$.
Parameters: η_1: LORETA step size . η_2: low-dimensional similarity learning step size. s: number of low-dimensional learning steps per round

```
repeat:
    $[g_1, g_2] = \nabla\mathcal{L}(AB^T)$
    $[A, B, A^\dagger, B^\dagger] = \text{LORETA}\left(A, B, A^\dagger, B^\dagger, g_1, g_2, \eta_1\right)$
    initialize $M = I_k$
    for i=1:s
        $[g_1, g_2] = \nabla\mathcal{L}(AMB^T)$
        $M = full-rank-metric-learning\left(M, A^T g_1, B^T g_2, \eta_2\right)$
    endfor
    $[U, S, V] = svd(M)$
    $A = A \cdot U \cdot sqrt(S)$
    $B = B \cdot V \cdot sqrt(S)$
until stopping condition is satisfied
```

2010), tensor-rank estimation (Eldén and Savas, 2009; Ishteva et al., 2011) and sparse PCA (Journée et al., 2010b).

Broadly speaking, there are two kinds of manifolds used in optimization. The first are *embedded manifolds*, manifolds that form a subset of Euclidean space, and are the ones we employ in this work. The second kind are *quotient manifolds*, which are formed by defining an equivalence relation on a Euclidean space, and endowing the resulting equivalence classes with an appropriate Riemannian metric. For example, the equivalence relation on $\mathbb{R}^n$ defined by $x \sim y \iff \exists\lambda > 0, x = \lambda y$, yields a quotient space called the *real projective space* when given a proper Riemannian metric.

More specific to the field of low-rank matrix manifolds, work has been done on the general problem of optimization with low-rank positive semi-definite (PSD) matrices. The latest and most relevant is the work of Meyer et al. (2011). In this work, Meyer and colleagues develop a framework for Riemannian stochastic gradient descent on the manifold of PSD matrices, and apply it to the problem of kernel learning and the learning of Mahalanobis distances. Their main technical tool is that of quotient manifolds mentioned above, as opposed to the embedded manifold we use in this work. Another paper which uses a quotient manifold representation is that of Journée et al. (2010a), which introduces a method for optimizing over low-rank PSD matrices.

In their 2010 paper (Vandereycken and Vandewalle, 2010), Vandereycken et al. introduced a retraction for PSD matrices in the context of modeling systems of partial differential equations. We build on this work in order to construct our methods of learning general and PSD low-rank matrices.

In general, the problem of minimizing a convex function over the set of low-rank matrices was addressed by several authors, including Fazel (2002). Recht et al. (2010) and more recently Jain et al. (2011) also consider the same problem, with additional affine constraints, and its connection to recent advances in compressed sensing. The main tools used in these papers are the trace norm (sum of singular values) and semi-definite programming. See also Fazel et al. (2005) for a short introduction to these methods.

More closely related to the current paper are the papers by Kulis et al. (2009) and Meka et al. (2008). Kulis et al. (2009) deal with learning low-rank PSD matrices, and use the rank-preserving log-det divergence and clever factorization and optimization in order to derive an update rule with runtime complexity of $O(nk^2)$ for an $n \times n$ matrix of rank k. Meka et al. (2008) use online learning in order to find a minimal rank square matrix under approximate affine constraints. The algorithm does not directly allow a factorized representation, and depends on an "oracle" component, which typically requires to compute an SVD.

Multi-class ranking with a large number of features was studied by Bai et al. (2009), and in the context of factored representations, by Weston et al. (2011) (WSABIE). WSABIE combines projected gradient updates with a novel sampling scheme which is designed to minimize a ranking loss named WARP. WARP is shown to outperform simpler triplet sampling approaches. Since WARP yields rank-1 gradients, it can easily be adapted for Riemannian SGD, but we leave experiments with such sampling schemes to future work.

7. Experiments

We tested LORETA in two learning tasks: learning a similarity measure between pairs of text documents using the 20-newsgroups data collected by Lang (1995), and learning to rank image label annotations based on a multi-label annotated set, using the *ImageNet* data set (Deng et al., 2009). Matlab code for LORETA-1 is available online at *http://chechiklab.biu.ac.il/research/LORETA*.

7.1 Learning Similarity on the 20 Newsgroups Data Set

In our first set of experiments, we looked at the problem of learning a similarity measure between pairs of text documents. Similarity learning is a well studied problem, closely related to metric learning (see Yang 2007 for a review). It has numerous applications in information retrieval such as *query by example*, and finding related content on the web.

One approach to learn pairwise relations is to measure the similarity of two documents $\mathbf{p}, \mathbf{q} \in \mathbb{R}^n$ using a bilinear form parametrized by a model $W \in \mathbb{R}^{n \times n}$:

$$S_W(\mathbf{p}, \mathbf{q}) = \mathbf{p}^T W \mathbf{q}.$$

Such models can be learned online (Chechik et al., 2010) and were shown to achieve high precision. Sometimes the matrix W is required to be symmetric and positive definite, which means it actually encodes a metric, also known as a Mahalanobis distance. Unfortunately, since the number of parameters grows as n^2, storing the matrix W in memory is only feasible for limited feature dimensionality. To handle larger vocabularies, like those containing all textual terms found in a corpus, a common approach is to pre-select a subset of the features and train a model over the low dimensional data. However, such preprocessing may remove crucial signals in the data even if features are selected in a discriminative way.

To overcome this difficulty, we used LORETA-1 and LORETA-1-PSD to learn a rank-k parametrization of the model W. This model can be factorized as $W = AB^T$, where $A, B \in \mathbb{R}^{n \times k}$ for the general case, or as $W = AA^T$ for the PSD case. In each of our experiments, we selected a subset of n features, and trained a rank k model. We varied the number of features n and the rank of the matrix k

so as to use a fixed amount of memory. For example, we used a rank-10 model with $50K$ features, and a rank-50 model with $10K$ features.

7.1.1 SIMILARITY LEARNING WITH LORETA-1

We use an online procedure similar to that in Grangier and Bengio (2008) and Chechik et al. (2010). At each round, three instances are sampled: a query document $\mathbf{q} \in \mathbb{R}^n$, and two documents $\mathbf{p}_+$, $\mathbf{p}_- \in \mathbb{R}^n$ such that $\mathbf{p}_+$ is known to be more similar to $\mathbf{q}$ than $\mathbf{p}_-$. We wish that the model assigns a higher similarity score to the pair $(\mathbf{q}, \mathbf{p}_+)$ than the pair $(\mathbf{q}, \mathbf{p}_-)$, and hence use the online ranking hinge loss defined as $l_W(\mathbf{q}, \mathbf{p}_+, \mathbf{p}_-) = [1 - S_W(\mathbf{q}, \mathbf{p}_+) + S_W(\mathbf{q}, \mathbf{p}_-)]_+$, where $[z]_+ = max(z, 0)$.

We initialized the model to be a truncated identity matrix, with only the first k ones along the diagonal. This corresponds in our case to choosing the k most informative terms as the initial data projection.

7.1.2 DATA PREPROCESSING AND FEATURE SELECTION

We used the 20 newsgroups data set (people.csail.mit.edu/jrennie/20Newsgroups), containing 20 classes with approximately 1000 documents each. We removed stop words but did not apply stemming. The document terms form a vocabulary of 50,000 terms, and we selected a subset of these features that conveyed high information about the identity of the class (over the training set) using the *infogain* criterion (Yang and Pedersen, 1997). This is a discriminative criterion,which measures the number of bits gained for category prediction by knowing the presence or absence of a term in a document. The selected features were normalized using *tf-idf*, and then represented each document as a bag of words. Two documents were considered similar if they shared the same class label, out of the possible 20 labels.

7.1.3 EXPERIMENTAL PROCEDURE AND EVALUATION PROTOCOL

The 20 newsgroups site proposes a split of the data into train and test sets. We repeated splitting 5 times based on the sizes of the proposed splits (a train / test ratio of 65% / 35%). We evaluated the learned similarity measures using a ranking criterion. We view every document $\mathbf{q}$ in the test set as a query, and rank the remaining test documents $\mathbf{p}$ by their similarity scores $\mathbf{q}^T W \mathbf{p}$. We then compute the precision (fraction of positives) at the top r ranked documents. We then average the precision over all positions r such that there exists a positive example in the top r. This final measure is called *mean average precision*, and is commonly used in the information retrieval community (Manning et al., 2008, Chapter 8).

7.1.4 COMPARISONS

We compared LORETA with the following approaches.

1. **Naive gradient descent** (GD): similar to Bai et al. (2009). The model is represented as a product of two matrices $W = AB^T$. Stochastic gradient descent steps are computed over the factors A and B, for the same loss used by LORETA $l_W(\mathbf{q}, \mathbf{p}_+, \mathbf{p}_-)$. The GD steps are:

$$A_{new} = A - \eta\, \mathbf{q}(\mathbf{p}_- - \mathbf{p}_+)^T B,$$
$$B_{new} = B - \eta\, (\mathbf{p}_- - \mathbf{p}_+)\mathbf{q}^T A\,.$$

We found this approach to be very unstable, and thus its results are not presented.

2. **Naive PSD gradient descent**: similar to the method above, except that now the model is constrained to be PSD. The model is represented as a product $W = AA^T$. Stochastic gradient descent steps are computed over the factor A for the same loss used by LORETA : $l_W(\mathbf{q}, \mathbf{p}_+, \mathbf{p}_-)$. As shown by Meyer et al. (2011), this is in fact equivalent to Riemannian stochastic GD in the manifold of PSD matrices when this manifold is endowed with a certain metric the authors call the *flat metric*.

   The GD step is:

$$A_{new} = A - \eta\, \left(\mathbf{q}(\mathbf{p}_- - \mathbf{p}_+)^T + (\mathbf{p}_- - \mathbf{p}_+)\mathbf{q}^T\right) A.$$

   The step size η was chosen by cross validation. This approach was more stable in the PSD case than in the general case, probably because the invariant space here is only the group of orthogonal matrices (which are well-conditioned), as opposed to the group of invertible matrices which might be ill-conditioned.

3. **Iterative Passive-Aggressive (PA)**: since we found the above general GD procedure **(1)** to be very unstable, we experimented with a related online algorithm from the family of passive-aggressive algorithms (Crammer et al., 2006). We iteratively optimize over A given a fixed B and vice versa. The optimization is a tradeoff between minimizing the loss l_W, and limiting how much the models change at each iteration. The steps sizes for updating A and B are computed to be:

$$\begin{aligned} \eta_A &= \min\left(\frac{l_W(\mathbf{q}, \mathbf{p}_+, \mathbf{p}_-)}{\|\mathbf{q}\|^2 \cdot \|B^T(\mathbf{p}_+ - \mathbf{p}_-)\|^2}, C\right), \\ \eta_B &= \min\left(\frac{l_W(\mathbf{q}, \mathbf{p}_+, \mathbf{p}_-)}{\|(\mathbf{p}_+ - \mathbf{p}_-)\|^2 \cdot \|A^T\mathbf{q}\|^2}, C\right). \end{aligned}$$

   C is a predefined parameter controlling the maximum magnitude of the step size, chosen by cross-validation. This procedure is numerically more stable because of the normalization by the norms of the matrices multiplied by the gradient factors.

4. **Full rank similarity learning models.** We compared with two full rank online metric learning methods, LEGO (Jain et al., 2008) and OASIS (Chechik et al., 2010). Both algorithms learn a full (non-factorized) model, and were run with $n = 1000$, in order to be consistent with the memory constraint of LORETA-1. We have also compared with both full-rank models using rank 2000, that is, 4 times the memory constraint. We have not compared with batch approaches such as Kulis et al. (2009), since they are not expected to scale to very large data sets such as those our work is ultimately aiming towards.

In addition, we have experimented with the method for learning PSD matrices using a polar geometry characterization of the quotient manifold, due to Meyer et al. (2011). This method's runtime complexity is $O((n+m)k^2)$, and we have found that its performance was not in line with the methods described above.

7.1.5 RESULTS

Figure 3c shows the mean average precision obtained with all the above methods. LORETA outperforms the other approaches across all ranks. LORETA-PSD achieves slightly higher precision than LORETA. The reason may be that similarity was defined based on two samples belonging to a common class, and this relation is symmetric and transitive, two relations which are respected by PSD matrices but not by general similarity matrices. Moreover, LORETA-PSD learned faster along the training iterations when compared with LORETA - see Figure 3a for a comparison of the learning curves. Interestingly, for both LORETA algorithms learning a low-rank model of rank 30, using the best 16660 features, was significantly more precise than learning a much fuller model of rank 100 and 5000 features, or a model using the full 50000 word vocabulary but with rank 10 . The intuition is that LORETA can be viewed as adaptively learning a linear projection of the data into low dimensional space, which is tailored to the pairwise similarity task.

7.2 Image Multilabel Ranking

Our second set of experiments tackled the problem of learning to rank labels for images taken from a large number of classes $(L = 1660)$ with multiple labels per image.

In our approach, we learn a linear classifier over n features for each label $c \in \mathcal{C} = \{1,\dots,L\}$, and stack all models together to a single matrix $W \in \mathbb{R}^{L\times n}$. At test time, given an image $\mathbf{p} \in \mathbb{R}^n$, the product $W\mathbf{p}$ provides scores for every label for that image $\mathbf{p}$. Given ground truth labeling, a good model would rank the true labels higher than the false ones. Each row of the matrix model can be thought of as a sub-model for the corresponding label. Imposing a low-rank constraint on the model implies that these sub-models are linear combinations of a smaller number of latent models. Alternatively, we can view learning a factored rank-k model $W = AB^T$ as learning a projection and classifier in the projected space concurrently. The matrix B^T projects the data onto a k dimensional space, and the matrix A consists of L classifiers operating in the low-dimensional space. The data we used for the experiment had $\sim$1500 labels, but the full ImageNet data set currently has $\sim$15000 labels, and is growing.

7.2.1 ONLINE LEARNING OF LABEL RANKINGS WITH LORETA-1

At each iteration, an image $\mathbf{p}$ is sampled, and using the current model W the scores for all its labels are computed, $W\mathbf{p}$. These scores are compared with the ground truth labeling $\mathbf{y} = \{y_1,\dots,y_r\} \subset \mathcal{C}$. We wish for all the scores of the true labels to be higher than the scores for the other labels by a margin of 1. Thus, the learner suffers a multilabel multiclass hinge loss as follows. Let $\bar{y} = \operatorname{argmax}_{s\notin\mathbf{y}}(W\mathbf{p})_s$, be the negative label which obtained the highest score, where $(W\mathbf{p})_s$ is the s^{th} component of the score vector $W\mathbf{p}$.

The loss is then $\mathcal{L}(W,\mathbf{p},\mathbf{y}) = \sum_{i=1}^{r}\left[(W\mathbf{p})_{\bar{y}} - (W\mathbf{p})_{y_i} + 1\right]_+$, which is the sum of the margins between the top-ranked false label and all the positive labels which violated the margin of one from it. We used the subgradient G of this loss for LORETA: for the set of indices $i_1, i_2, \dots i_d \subset \mathbf{y}$ which incurred a non zero hinge loss, the i_j row of G is $\mathbf{p}$, and for the row $\bar{y}$ G is $-d\cdot\mathbf{p}$. The matrix G is rank one, unless no loss was suffered in which case it is 0.

The non-convex and stochastic nature of the learning procedure has lead us to try several initial conditions:

- **Zero matrix**: in this initialization we begin with a low-rank matrix composed entirely of zeros. This matrix is not included in the low-rank manifold $\mathcal{M}_k^{n,m}$, since its rank is less than k. We therefore perform a simple pre-training session in which we add up subgradients until a matrix of rank k is obtained. In practice we added the first $2k$ subgradients (each such subgradient being of rank one), and then performed an SVD to obtain the best rank-k model.

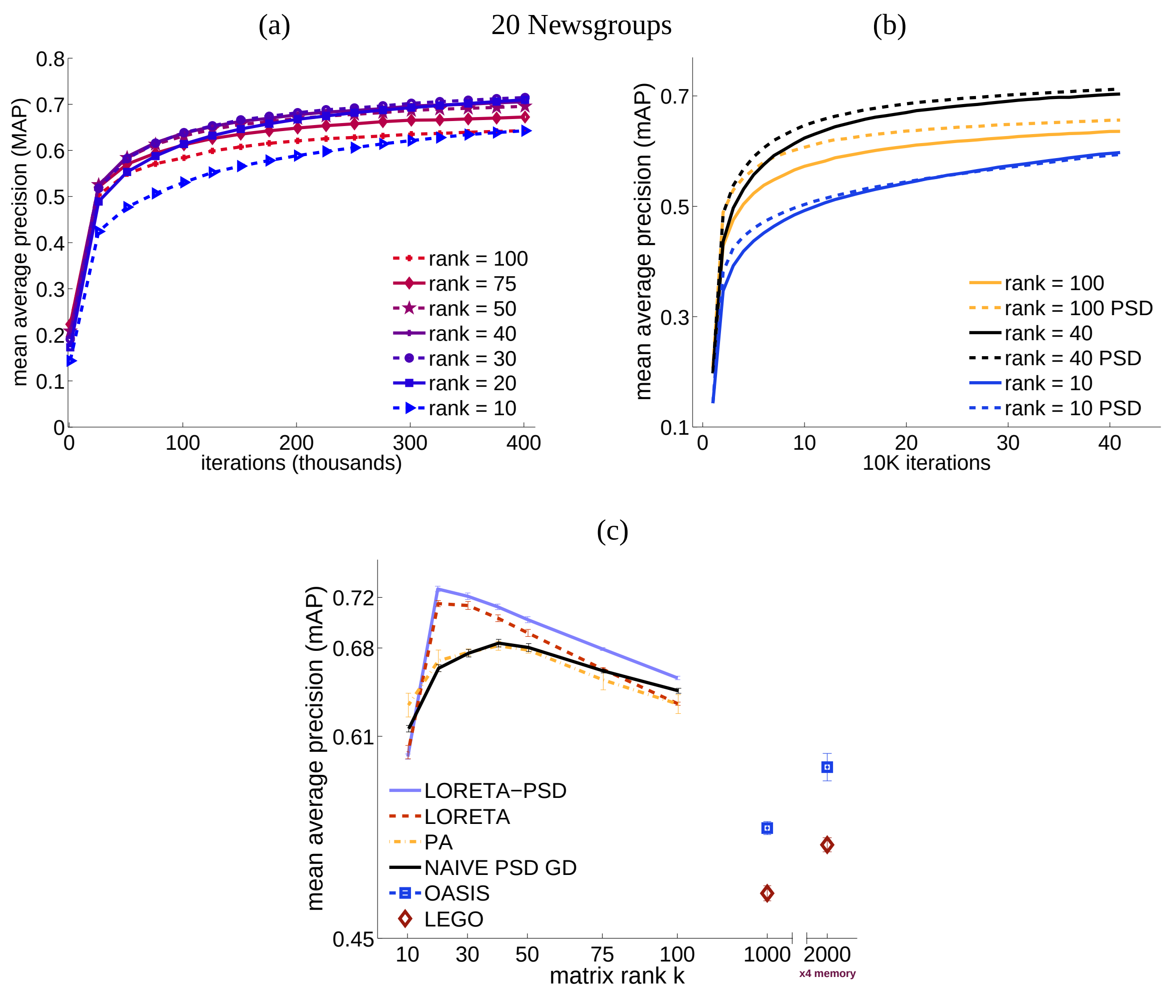

Figure 3: (a) Mean average precision (mAP) over 20 newsgroups test set as traced along LORETA learning for various ranks. Curve values are averages over 5 train-test splits. (b) Comparison of the learning curves of LORETA and LORETA-PSD. LORETA-PSD learns faster than LORETA across all ranks (shown are results for ranks 10, 40 and 100). (c) mAP of different models with varying rank. For each rank, a different number of features was selected using an information gain criterion, such that the total memory requirement is kept fixed (number of features $\times$ rank is constant). 50000 features were used for rank = 10. LEGO and OASIS were trained with the same memory (using 1000 features and rank=1000), as well as with 4 times the same memory (rank=2000). Error bars denote the standard error of the mean over 5 train-test splits.

We chose $2k$ because we wanted to ensure that the matrix we obtain has rank greater or equal to k.

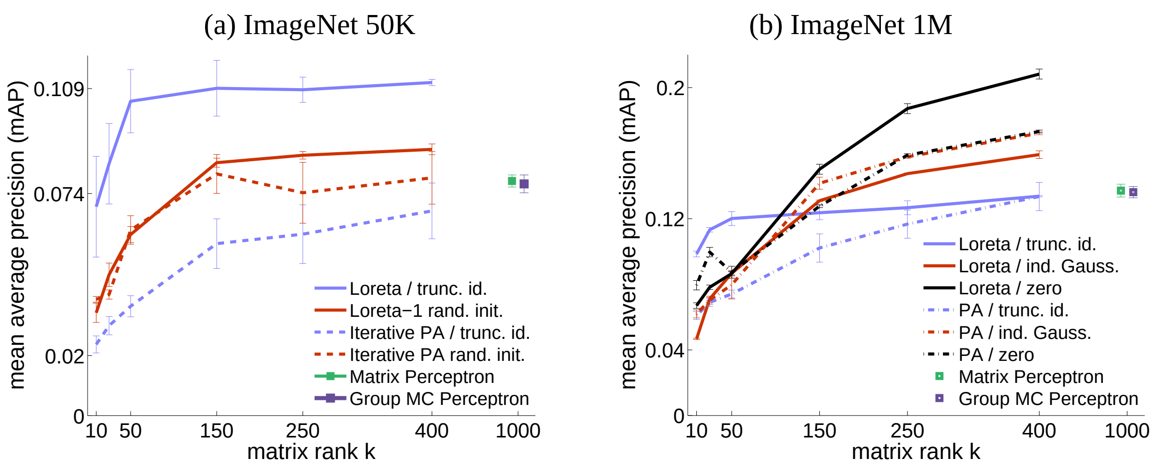

Figure 4: ImageNet data. Mean average precision (mAP) as a function of the rank k. Curves are means over five train-test splits. Error bars denote the standard error of the mean. Note the different scale of the left and right figure. All hyper parameters were selected using cross validation. Three different initializations were used: the zero matrix, a zero padded $k \times k$ identity matrix, and a product of two i.i.d. Gaussian matrices. See Section 7.2.1 for details.

- **Zero-padded identity**: we begin with a matrix composed of the $k \times k$ identity matrix I_k on the top left corner, padded with zeros so as to form an $L \times n$ matrix. This is guaranteed to be of rank k. The choice of the location of the identity matrix block is arbitrary.
- **Independent Gaussian**: we sample independently the entries of the two factor matrices $A \in \mathbb{R}^{n \times k}$, $B\mathbb{R}^{m \times k}$ from a standard normal distribution. This model is thus initialized as a product of two random Gaussian matrices.

7.2.2 Data Set and Preprocessing

We used data from the ImageNet 2010 Challenge (www.imagenet.org/challenges/LSVRC/2010/) containing images labeled with respect to the WordNet hierarchy. Each image was manually labeled with a single class label (for a total of 1000 classes). We added labels for each image, using classes along the path to the root of the hierarchy (adding 676 classes in total). We discarded ancestor labels covering more than 10% of the images, leaving 1660 labels (5.2 labels per image on average). We used ImageNet's bag of words representation, based on vector quantizing SIFT features with a vocabulary of 1000 words, followed by *tf-idf* normalization.

7.2.3 Experimental Procedure and Evaluation Protocol

We trained on two data sets. A medium scale one of 50000 images, and a large data set consisting of 908210 images. We tested on 20000 images for the medium scale, and 252284 images for the large scale. The quality of the learned label ranking was evaluated using the *mean average precision* (mAP) criterion mentioned in 7.1.3 above (Manning et al., 2008, Chapter 8).

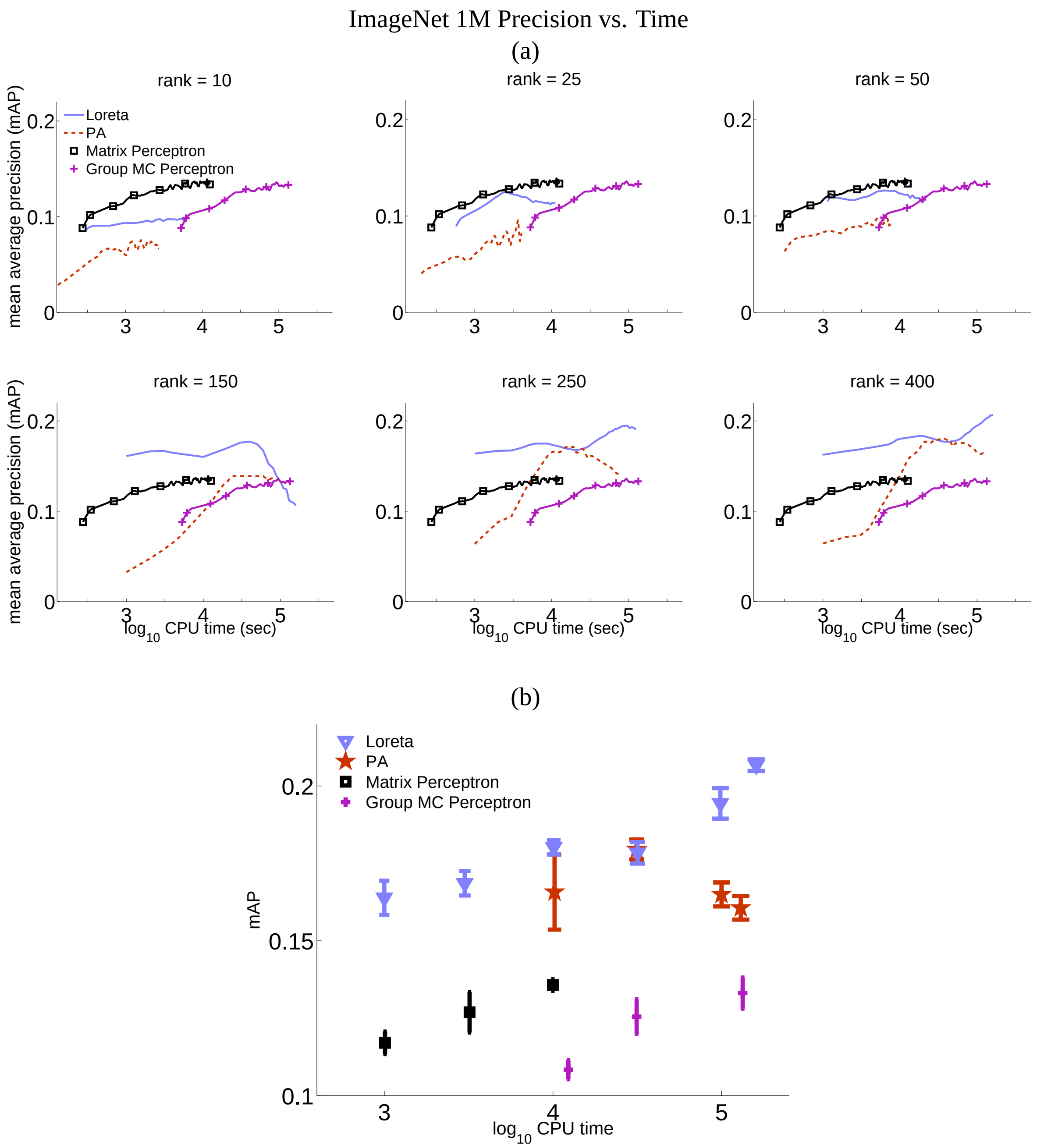

Figure 5: (a) Mean average precision (mAP) as function of single CPU processing time in seconds for different algorithms and model ranks, presented on a log-scale. Matrix Perceptron (black squares) and Group Multi-Class Perceptron (purple crosses) are both full rank (rank=1000), and their curves are reproduced on all six panels for comparison. For each rank and algorithm (LORETA and PA), we used the best performing initialization scheme. (b) mAP of the best performing model for different algorithms and time points. Error bars represent standard deviation across 5 train-test splits.

7.2.4 COMPARISONS

We compared the performance of LORETA on this task with three other approaches:

1. **PA - Iterative Passive-Aggressive**: same as described in Section 7.1.4 above for the 20 Newsgroups experiment.
2. **Matrix Perceptron**: a full rank stochastic subgradient descent. The model is initialized as a zero matrix of size 1660×1000, and in each round the loss subgradient is subtracted from it. After a sufficient number of rounds, the model is typically full rank and dense.
3. **Group Multi-Class Perceptron**: a mixed (2,1) norm online mirror descent algorithm (Kakade et al., 2010). This algorithm encourages a group-sparsity pattern within the learned matrix model, thus presenting an alternative form of regularization when compared with low-rank models.

LORETA and PA were run using a range of model ranks. For all three methods, the step size (or the C parameter for PA) was chosen by 5-fold cross validation on a validation set.

7.2.5 RESULTS

Figure 4 plots the mAP precision of LORETA and PA for different model ranks, while showing on the right the mAP of the full rank 1000 Matrix Perceptron and $(2,1)$ norm algorithms. LORETA significantly improves over all other methods across all ranks. However, we note that LORETA, being a non-convex algorithm, does depend significantly on the method of initialization, with the zero-padded identity matrix being the best initialization for lower rank models, and the zero matrix the best initialization for higher rank models (rank ≥ 150).

In Figure 5 we show the accuracy as a function of CPU tim on a single CPU for the different algorithms and model ranks. We ran Matlab R2011a on an Intel Xeon 2.27 GHz machine, and used Matlab's `-singlethread` flag to control multithreading. The higher-rank LORETA models outperform all others both in the short time scale ($\sim$ 1000 sec.) and the long time scale ($\sim 100,000$ sec.). For some of the higher-rank models there is evident overtraining at some point, but this overtraining could be avoided by adopting an early-stopping procedure.

8. Discussion

We presented LORETA, an algorithm which learns a low-rank matrix based on stochastic Riemannian gradient descent and efficient retraction to the manifold of low-rank matrices. LORETA achieves superior precision in a task of learning similarity in high dimensional feature spaces, and in multi-label annotation, where it scales well with the number of classes. A PSD variant of LORETA can be used efficiently for low-rank metric learning.

There are many ways to tie together different classifiers in a multi-class setting. We have seen here that a low-rank assumption coupled with a Riemannian SGD procedure outperformed the (2,1) mixed norm. Other approaches leverage the hierarchical structure inherent in many of these tasks. For example, Deng et al. (2011) use the label hierarchy of ImageNet to compute a similarity measure between images.

For similarity learning, the approach we take in this paper uses a weak supervision based on ranking similar pairs: one only knows that the pair $(\mathbf{q},\mathbf{p}_{+})$ is more similar than the pair $(\mathbf{q},\mathbf{p}_{-})$. In some cases, a stronger supervision signal is available, like the classes of each objects are known. In these cases, Deng et al. (2011) have shown how to use class identities to construct good features by training an SVM classifier on each class and using its scaled output as a feature. They show that such features can lead to very good performance, with the added advantage that the features can be learned in parallel. The weak supervision approach that we take here aims to handle the case, which is particularly common in large scale data sets collected through web users' activity, where weaker supervision is much easier to collect.

In this paper, we used simple sampling schemes for both the similarity learning and multiple-labelling experiments. More elaborate sampling techniques such as those proposed by Weston et al. (2011), which focus on "hard negatives", may yield significant performance improvements. As these approaches typically involve rank-one gradients when implemented as online learning algorithms, they are well suited for being used in conjunction with LORETA, and this will be the subject of future work.

LORETA yields a factorized representation of the low-rank matrix. For similarity learning, these factors project to a low-dimensional space where similarity is evaluated efficiently. For classification, it can be viewed as learning two matrix components: one that projects the high dimensional data into a low dimension, and a second that learns to classify in the low dimension. In both approaches, the low-dimensional space is useful for extracting the relevant structure from the high-dimensional data, and for exploring the relations between large numbers of classes.

Acknowledgments

G.C. was supported by the Israeli science foundation Grant 1001/08, and by a Marie Curie reintegration grant PIRG06-GA-2009-256566. U.S. and D.W. were supported by the European Union under the DIRAC integrated project IST-027787. Uri Shalit is a recipient of the Google Europe Fellowship in Machine Learning, and this research is supported in part by this Google Fellowship. We would also like to thank the Gatsby Charitable Foundation for its generous support of the ICNC.

Appendix A. Proof of Lemma 2

We formally define the tangent space of a manifold at a point on the manifold, and then describe an auxiliary parametrization of the tangent space to the manifold $\mathcal{M}_k^{n,m}$ at a point $W \in \mathcal{M}_k^{n,m}$.

Definition 7 *The tangent space $T_{\mathbf{W}}\mathcal{M}$ to a manifold $\mathcal{M} \subset \mathbb{R}^n$ at a point $W \in \mathcal{M}$ is the linear space spanned by all the tangent vectors at 0 to smooth curves $\gamma : \mathbb{R} \to \mathcal{M}$ such that $\gamma(0) = W$. That is, the set of tangents in $\mathbb{R}^n$ to smooth curves within the manifold which pass through the point W.*

In order to characterize the tangent space of $\mathcal{M}_k^{n,m}$, we look into the properties of smooth curves γ, where for each t, $\gamma(t) \in \mathcal{M}_k^{n,m}$.

For any such curve, because of the rank k assumption, we may assume that for all $t \in \mathbb{R}$, there exist (non-unique) matrices $A(t) \in \mathbb{R}_*^{n\times k}$, $B(t) \in \mathbb{R}_*^{m\times k}$, such that $\gamma(t) = A(t)B(t)^T$. We now wish to find the tangent vectors to these curves. By the product rule we have:

$$\dot{\gamma}(0) = A(0)\dot{B}(0)^T + \dot{A}(0)B(0)^T.$$

Since $W = \gamma(0) = A(0)B(0)^T = AB^T$ we have for $W = AB^T$:

$$T_{\mathbf{W}}\mathcal{M}_k^{n,m} = \left\{AX^T + YB^T | X \in \mathbb{R}^{m\times k}, Y \in \mathbb{R}^{n\times k}\right\}. \tag{5}$$

This is because any choice of matrices X, Y such that $X = \dot{B}, Y = \dot{A}$ will give us some tangent vector, and for any tangent vector there exist such matrices. The space above is clearly a linear space. Being a tangent space to a manifold, it has the same dimension as the manifold: $(n+m)k - k^2$.

Recall the definition of the tangent space given in Lemma 1:

$$T_{\mathbf{W}}\mathcal{M}_k^{n,m} = \left\{ \begin{bmatrix} A & A_\perp \end{bmatrix} \begin{bmatrix} M & N_1^T \\ N_2 & 0 \end{bmatrix} \begin{bmatrix} B^T \\ B_\perp^T \end{bmatrix} : M \in \mathbb{R}^{k\times k}, N_1 \in \mathbb{R}^{(m-k)\times k}, N_2 \in \mathbb{R}^{(n-k)\times k} \right\}. \tag{6}$$

To prove Lemma 2, it is easy to verify by counting that the dimension of the space as defined in Equation (6) above is $(n+m)k - k^2$. Using the notation above, we can see that by taking $X = MB^T + N_1B_\perp^T$ and $Y = A_\perp N_2$, the space defined in Equation (6) is included in $T_{\mathbf{W}}\mathcal{M}_k^{n,m}$ as defined in Equation (5). Since it is a linear subspace of equal dimension, both spaces must be equal ■

Appendix B. Proof of Theorem 3

We state the theorem again here.

Theorem 8 *Let $W \in \mathcal{M}_k^{n,m}$, $W = AB^T$, and $W^\dagger = B^{\dagger T}A^\dagger$. Let $\xi \in T_W\mathcal{M}_k^{n,m}$, $\xi = \xi^{AB} + \xi^{AB_\perp} + \xi^{A_\perp B}$, as in 1, and let:*

$$V_1 = W + \frac{1}{2}\xi^{AB} + \xi^{A_\perp B} - \frac{1}{8}\xi^{AB}W^\dagger\xi^{AB} - \frac{1}{2}\xi^{A_\perp B}W^\dagger\xi^{AB} \quad ,$$
$$V_2 = W + \frac{1}{2}\xi^{AB} + \xi^{AB_\perp} - \frac{1}{8}\xi^{AB}W^\dagger\xi^{AB} - \frac{1}{2}\xi^{AB}W^\dagger\xi^{AB_\perp} \quad .$$

The mapping

$$R_W(\xi) = V_1W^\dagger V_2 \tag{7}$$

is a second order retraction from a neighborhood $\Theta_W \subset T_W\mathcal{M}_k^{n,m}$ to $\mathcal{M}_k^{n,m}$.

Proof To prove that Equation (7) defines a retraction, we first show that $V_1W^\dagger V_2$ is a rank-k matrix. Note that there exist matrices $Z_1 \in \mathbb{R}^{n\times k}$ and $Z_2 \in \mathbb{R}^{m\times k}$ such that $V_1 = Z_1B^T$ and , $V_2 = AZ_2^T$. A sufficient condition for the matrices Z_1 and Z_2 to be of full rank is that the matrix M is of limited norm. Thus, for all tangent vectors lying in some neighborhood $\Theta_W \subset T_W\mathcal{M}_k^{n,m}$ of $0 \in T_W\mathcal{M}_k^{n,m}$, the above relation is indeed a retraction to the manifold. In practice this is never a problem, as the set of matrices not of full rank is of zero measure, and in practice we have found these matrices to always be of full rank. Thus, $R_W(\xi) = V_1W^\dagger V_2 = Z_1B^TB(B^TB)^{-1}(A^TA)^{-1}A^TAZ_2^T = Z_1Z_2^T$, which, given that Z_1 and Z_2 are of full column rank, is exactly a rank-k, $n\times m$ matrix.

Next we show that $R_W(\xi)$ is a retraction, and of second order. It is obvious that $R_W(0) = W$, since the projection of the zero vector is zero, and thus ξ^{AB}, $\xi^{AB_\perp}$ and $\xi^{A_\perp B}$ are all zero.

Expanding $V_1W^\dagger V_2$ up to second order terms in ξ, many terms cancel and we end up with:

$$\begin{aligned} R_W(\xi) &= W + \xi^{AB} + \xi^{AB_\perp} + \xi^{A_\perp B} + \xi^{A_\perp B}W^\dagger\xi^{AB_\perp} + O(\|\xi\|^3) \\ &= W + \xi + \xi^{A_\perp B}W^\dagger\xi^{AB_\perp} + O(\|\xi\|^3). \end{aligned}$$

Local first order rigidity is immediately apparent. If we expand the only second order term, $\xi^{A_\perp B}W^\dagger\xi^{AB_\perp}$, we see that it equals $A_\perp N_2 N_1^T B_\perp^T$. We claim this term is orthogonal to the tangent space $T_W\mathcal{M}_k^{n,m}$. If we take, using the characterization in Lemma 2, an arbitrary tangent vector $A\tilde{M}B^T + A\tilde{N}_1^T B_\perp^T + A_\perp \tilde{N}_2 B^T$ in $T_{\mathbf{W}}\mathcal{M}_k^{n,m}$, we can calculate the inner product:

$$\begin{aligned}
&\left\langle \left(A_\perp N_2 N_1^T B_\perp^T\right), \left(A\tilde{M}B^T + A\tilde{N}_1^T B_\perp^T + A_\perp \tilde{N}_2 B^T\right)\right\rangle = \\
&tr\left(B_\perp N_1 N_2^T A_\perp^T A\tilde{M}B^T + B_\perp N_1 N_2^T A_\perp^T A\tilde{N}_1^T B_\perp^T + B_\perp N_1 N_2^T A_\perp^T A_\perp \tilde{N}_2 B^T\right) = \\
&tr\left(B_\perp N_1 N_2^T A_\perp^T A_\perp \tilde{N}_2 B^T\right) = \\
&tr\left(B^T B_\perp N_1 N_2^T A_\perp^T A_\perp \tilde{N}_2\right) = 0
\end{aligned}$$

with the equalities stemming from the fact that $A_\perp^T A = 0$, $B_\perp^T B = 0$, and from standard trace identities. Thus, the second order term cancels out if we project the second derivative of the curve defined by the retraction, as required by the second-order condition

$$P_W\left(\frac{\mathrm{d}R_W(\tau\xi)}{\mathrm{d}\tau^2}|_{\tau=0}\right) = 0 \quad \forall\xi \in T_W\mathcal{M}.$$

We see that the second order term is contained in the normal space. This concludes the proof that the retraction is a second order retraction. ■

Appendix C. Proof of Lemma 4

Let us see how can we calculate the needed terms explicitly. When evaluating the expression $V_1 W^\dagger V_2$, we can use the algebraic relations: $WW^\dagger = P_A$ and $W^\dagger W = P_B$. From this we can conclude that: $WW^\dagger\xi^{AB} = \xi^{AB}$, $\xi^{AB}W^\dagger W = \xi^{AB}$, $\xi^{A_\perp B}W^\dagger W = \xi^{A_\perp B}$ and $WW^\dagger\xi^{AB_\perp} = \xi^{AB_\perp}$. These relations, along with many terms that cancel out, lead to the following expression:

$$\begin{aligned}
R_W(\xi) = V_1 W^\dagger V_2 = \\
& W + \xi^{AB} + \xi^{AB_\perp} + \xi^{A_\perp B} - \frac{1}{8}\xi^{AB}W^\dagger\xi^{AB}W^\dagger\xi^{AB} - \frac{3}{8}\xi^{AB}W^\dagger\xi^{AB}W^\dagger\xi^{AB_\perp} \\
& - \frac{3}{8}\xi^{A_\perp B}W^\dagger\xi^{AB}W^\dagger\xi^{AB} + \xi^{A_\perp B}W^\dagger\xi^{AB_\perp} - \xi^{A_\perp B}W^\dagger\xi^{AB}W^\dagger\xi^{AB_\perp} \\
& + \frac{1}{16}\xi^{AB}W^\dagger\xi^{AB}W^\dagger\xi^{AB}W^\dagger\xi^{AB_\perp} + \frac{1}{16}\xi^{A_\perp B}W^\dagger\xi^{AB}W^\dagger\xi^{AB}W^\dagger\xi^{AB} \\
& + \frac{1}{64}\xi^{AB}W^\dagger\xi^{AB}W^\dagger\xi^{AB}W^\dagger\xi^{AB} + \frac{1}{4}\xi^{A_\perp B}W^\dagger\xi^{AB}W^\dagger\xi^{AB}\xi^{AB_\perp}.
\end{aligned}$$

We now substitute the matrices M, N_1 and N_2 into the above relation. Most terms cancel out. For example, we have the identity $\xi^{AB}W^\dagger\xi^{AB} = AM^2B^T$, $\xi^{AB}W^\dagger\xi^{AB}W^\dagger\xi^{AB} = AM^3B^T$ and so forth. We obtain the following relation:

$$\begin{aligned}
R_W(\xi) = & AB^T + AMB^T + AN_1^T B_\perp^T + A_\perp N_2 B^T - \frac{1}{8}AM^3B^T \\
& - \frac{3}{8}AM^2N_1^T B_\perp^T - \frac{3}{8}A_\perp N_2 M^2 B^T + A_\perp N_2 N_1^T B_\perp^T - A_\perp N_2 M N_1^T B_\perp^T \\
& + \frac{1}{16}AM^3N_1^T B_\perp^T + \frac{1}{16}A_\perp N_2 M^3 B^T + \frac{1}{64}AM^4B^T + \frac{1}{4}A_\perp N_2 M^2 N_1^T B_\perp^T.
\end{aligned}$$

Collecting terms by the leftmost and rightmost factors, we obtain:

$$\begin{aligned} R_W(\xi) = & A\left(I_k + M - \frac{1}{8}M^3 + \frac{1}{64}M^4\right)B^T \\ & + A\left(I_k - \frac{3}{8}M^2 + \frac{1}{16}M^3\right)N_1^T B_\perp^T \\ & + A_\perp N_2\left(I_k - \frac{3}{8}M^2 + \frac{1}{16}M^3\right)B^T \\ & + A_\perp N_2\left(I_k - M + \frac{1}{4}M^2\right)N_1^T B_\perp^T \quad . \end{aligned}$$

Finally, treating the first and fourth lines as a polynomial expression in M, and taking its polynomial square root, we can split the above sum into the product of an $n \times k$ matrix and a $k \times m$ matrix:

$$\begin{aligned} R_W(\xi) = & \left[A\left(I_k + \frac{1}{2}M - \frac{1}{8}M^2\right) + A_\perp N_2\left(I_k - \frac{1}{2}M\right)\right] \cdot \\ & \left[B\left(I_k + \frac{1}{2}M^T - \frac{1}{8}\left(M^T\right)^2\right) + B_\perp N_1\left(I_k - \frac{1}{2}M^T\right)\right]^T . \end{aligned}$$

Appendix D. Rank One Pseudoinverse Update Rule

For completeness we develop below the procedure for updating the pseudoinverse of a rank-1 perturbed matrix (Meyer, 1973), following the derivation of Petersen and Pedersen (2008). We wish to find a matrix G such that for a given matrix A along with its pseudo-inverse $A^\dagger$, and vectors of appropriate dimension c and d, we have:

$$\left(A + cd^T\right)^\dagger = A^\dagger + G.$$

We have used the fact that A has a full column rank to simplify slightly the algorithm of Petersen and Pedersen (2008).

References

P.-A. Absil and J. Malick. Projection-like retractions on matrix manifolds. Technical Report UCL-INMA-2010.038, Department of Mathematical Engineering, Université catholique de Louvain, July 2010.

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton Univ Press, 2008.

B. Bai, J. Weston, R. Collobert, and D. Grangier. Supervised semantic indexing. *Advances in Information Retrieval*, pages 761–765, 2009.

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(1):937–965, 2006.

Algorithm 7 : Rank one pseudo-inverse update

Input: Matrices $A, A^\dagger \in \mathbb{R}_*^{n\times k}$, such that $A^\dagger$ is the pseudo-inverse of A, vectors $c \in \mathbb{R}^{n\times 1}$, $d \in \mathbb{R}^{k\times 1}$

Output: Matrix $Z^\dagger \in \mathbb{R}_*^{k\times n}$, such that $Z^\dagger$ is the pseudo-inverse of $A + cd^T$.

```
Compute:                                              matrix dimension
    $v = A^\dagger c$                                         $k \times 1$
    $\beta = 1 + d^T v$                                       $1 \times 1$
    $n = A^{\dagger T} d$                                     $n \times 1$
    $\hat{n} = A^\dagger n$                                   $k \times 1$
    $w = c - Av$                                              $n \times 1$
    if $\beta \neq 0$ AND $\|w\| \neq 0$
        $G = \frac{1}{\beta}\hat{n} w^T$                      $k \times n$
        $s = \frac{\beta}{\|w\|^2\|n\|^2+\beta^2}$            $1 \times 1$
        $t = \frac{\|w\|^2}{\beta}\hat{n} + v$                $k \times 1$
        $\hat{G} = s \cdot t \left(\frac{\|n\|^2}{\beta} w + n\right)^T$    $k \times n$
        $G = G - \hat{G}$                                     $k \times n$
    elseif $\beta = 0$ AND $\|w\| \neq 0$
        $G = -A^\dagger \frac{n}{\|n\|^2}$                    $k \times 1$
        $G = G n^T$                                           $k \times 1$
        $\hat{G} = v \frac{w^T}{\|w\|^2}$                     $k \times n$
        $G = G - \hat{G}$                                     $k \times n$
    elseif $\beta \neq 0$ AND $\|w\| = 0$
        $G = -\frac{1}{\beta} v n^T$                          $k \times n$
    elseif $\beta = 0$ AND $\|w\| = 0$
        $\hat{v} = \frac{1}{\|v\|^2} v \left(v^T A^\dagger\right)$    $k \times n$
        $\hat{n} = \frac{1}{\|n\|^2} \left(A^\dagger n\right) n^T$    $k \times n$
        $G = \frac{v^T A^\dagger n}{\|v\|^2\|n\|^2} v n^T - \hat{v} - \hat{n}$    $k \times n$
    endif
    $Z^\dagger = A^\dagger + G$
```

J. Briët, F.M. de Oliveira Filho, and F. Vallentin. The Grothendieck problem with rank constraint. In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems, MTNS*, 2010.

G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135, 2010.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the 22nd IEEE Conference on Computer Vision and Pattern*

Recognition, pages 248–255, 2009.

J. Deng, A. Berg, and L. Fei-Fei. Hierarchical Semantic Indexing for Large Scale Image Retrieval. In *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition*, pages 785–792, 2011.

M.P. Do Carmo. *Riemannian Geometry*. Birkhauser, 1992.

L. Eldén and B. Savas. A Newton–Grassmann method for computing the best multi-linear rank-(r1, r2, r3) approximation of a tensor. *SIAM Journal on Matrix Analysis and applications*, 31(2): 248–271, 2009.

M. Fazel. *Matrix Rank Minimization with Applications*. PhD thesis, Electrical Engineering Department, Stanford University, 2002.

M. Fazel, H. Hindi, and S. Boyd. Rank minimization and applications in system theory. In *Proceedings of the 2004 American Control Conference*, pages 3273–3278. IEEE, 2005.

A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems*, volume 18, page 451, 2006.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, volume 17, 2005.

D. Grangier and S. Bengio. A discriminative kernel-based model to rank images from text queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1371–1384, 2008.

M. Ishteva, L. De Lathauwer, P.-A. Absil, and S. Van Huffel. Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme. *SIAM Journal on Matrix Analysis and Applications*, 32(1):115–132, 2011.

P. Jain, B. Kulis, I.S. Dhillon, and K. Grauman. Online metric learning and fast similarity search. In *Advances in Neural Information Processing Systems*, volume 20, pages 761–768, 2008.

P. Jain, R. Meka, and I. Dhillon. Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, volume 24, pages 937–945, 2011.

M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre. Low-Rank Optimization on the Cone of Positive Semidefinite Matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010a.

M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre. Generalized power method for sparse principal component analysis. *The Journal of Machine Learning Research*, 11:517–553, 2010b.

S.M. Kakade, S. Shalev-Shwartz, and A. Tewari. Regularization techniques for learning with matrices, 2010. Arxiv preprint arXiv:0910.0610v2.

R.H. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *The Journal of Machine Learning Research*, 99:2057–2078, 2010.

B. Kulis, M.A. Sustik, and I.S. Dhillon. Low-rank kernel learning with bregman matrix divergences. *The Journal of Machine Learning Research*, 10:341–376, 2009.

K. Lang. Learning to filter netnews. In *Proceeding of the 12th Internation Conference on Machine Learning*, pages 331–339, 1995.

C.D. Manning, P. Raghavan, H. Schutze, and Ebooks Corporation. *Introduction to Information Retrieval*, volume 1. Cambridge University Press Cambridge, UK, 2008.

R. Meka, P. Jain, C. Caramanis, and I.S. Dhillon. Rank minimization via online learning. In *Proceedings of the 25th International Conference on Machine learning*, pages 656–663, 2008.

C.D. Meyer. Generalized inversion of modified matrices. *SIAM Journal on Applied Mathematics*, 24(3):315–323, 1973.

G. Meyer, S. Bonnabel, and R. Sepulchre. Regression on fixed-rank positive semidefinite matrices: a Riemannian approach. *The Journal of Machine Learning Research*, 12:593–625, 2011.

B.K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24 (2):227–234, 1995.

Sahand Negahban and Martin J. Wainwright. Estimation of (near) low-rank matrices with noise and high-dimensional scaling. In *Proceedings of the 27th International Conference on Machine Learning*, pages 823–830, 2010.

C. Oberlin and S.J. Wright. Active set identification in nonlinear programming. *SIAM Journal on Optimization*, 17(2):577–605, 2007.

K.B. Petersen and M.S. Pedersen. The matrix cookbook, Oct. 2008.

B. Recht, M. Fazel, and P.A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

S. Shalev-Shwartz, Y. Singer, and A.Y. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the Twenty-first International Conference on Machine Learning*, page 94. ACM, 2004.

Uri Shalit, Daphna Weinshall, and Gal Chechik. Online learning in the manifold of low-rank matrices. In *Advances in Neural Information Processing Systems 23*, pages 2128–2136. MIT Press, 2010.

B. Vandereycken and S. Vandewalle. A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations. *SIAM Journal on Matrix Analysis and Applications*, 31(5): 2553–2579, 2010.

B. Vandereycken, P.-A. Absil, and S. Vandewalle. Embedded geometry of the set of symmetric positive semidefinite matrices of fixed rank. In *Statistical Signal Processing, 2009. SSP'09. IEEE/SP 15th Workshop on*, pages 389–392. IEEE, 2009.

K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009.

J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI–11)*, 2011.

E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, volume 15, pages 505–512. MIT Press, 2002.

L. Yang. An overview of distance metric learning. Technical report, School of Computer Science, Carnegie Mellon University, 2007.

Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420, 1997.