METRIC LEARNING WITH RANK AND SPARSITY CONSTRAINTS

Bubacarr Bah† *Stephen Becker*⋆ *Volkan Cevher*† *Baran Gözcü*†

† Laboratory for Information and Inference Systems, EPFL, Switzerland
⋆ IBM Research, Yorktown Heights, USA

ABSTRACT

Choosing a distance preserving measure or *metric* is fundamental to many signal processing algorithms, such as k-means, nearest neighbor searches, hashing, and compressive sensing. In virtually all these applications, the efficiency of the signal processing algorithm depends on how fast we can evaluate the learned metric. Moreover, storing the chosen metric can create space bottlenecks in high dimensional signal processing problems. As a result, we consider data dependent metric learning with rank as well as sparsity constraints. We propose a new non-convex algorithm and empirically demonstrate its performance on various datasets; a side benefit is that it is also much faster than existing approaches. The added sparsity constraints significantly improve the speed of multiplying with the learned metrics without sacrificing their quality.

Index Terms— Metric learning, Nesterov acceleration, sparsity, low-rank, proximal gradient methods

1. INTRODUCTION

Learning "good distance" metrics for signals is key in real-world applications, such as data classification and retrieval. For instance, in the k-nearest-neighbor (KNN) classifier, we identify the k-nearest labeled images given a test image in the space of visual features. Hence, it is important to learn metrics that capture the similarity as well as the dissimilarity of datasets. Interestingly, several works have shown that properly chosen distance metrics can significantly benefit KNN accuracy as compared to the usual Euclidean metric [1–3].

In the standard metric learning problem (MLP) we learn a (semi-) norm $\|\mathbf{x}\|_{\mathbf{B}} = \sqrt{\mathbf{x}^T\mathbf{B}\mathbf{x}}$ that respects data $\mathbf{x} \in \mathbb{R}^N$ dissimilarity constraints on a set $\mathcal{D}$ while obtaining the best similarity on a set $\mathcal{S}$:

$$\begin{aligned} &\min_{\mathbf{B}\succeq 0} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{B}}^2 \\ &\text{subject to} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{B}}^2 \geq 1. \end{aligned} \tag{1}$$

This work was supported in part by the European Commission under Grant MIRG-268398, ERC Future Proof, SNF 200021-132548, SNF 200021-146750 and SNF CRSII2-147633. The author names are in alphabetical order.

Even though (1) is a convex program, its numerical solution proves to be challenging since it does not fit into the standard convex optimization formulations, such as semidefinite or quadratic programming. Moreover, as the number of variables is quadratic with the number of features, even the most basic gradient-only approaches do not scale well [1, 4].

In general, the solution of (1) results in metrics that are full rank. Hence, the learned metric creates storage as well as computational bottlenecks. As a result, several works [2, 3, 5, 6] consider learning rank-constrained metrics. Unfortunately, enforcing rank constraints on the already computationally challenging MLP (1) makes it non-convex. Surprisingly, under certain conditions, it is possible to prove approximation quality and the generalization of solution algorithms in the high-dimensional limit [3, 5].

In this paper, we reformulate the standard MLP (1) into a non-convex optimization framework that can incorporate sparsity constraints in addition to the rank. That is, we learn distance metrics $\mathbf{B} = \mathbf{A}\mathbf{A}^T$ such that $\mathbf{A} \in \mathbb{R}^{N\times r}$ for a given $r \ll N$ and $\mathbf{A}$ is *sparse*. A sparse $\mathbf{A}$ has computational benefits like low storage and computational complexity. Consequently, this work could be useful in sparse low-rank matrix factorization which has numerous applications in machine learning including learning [7] and deep neural networks (deep learning) [8] and autoencoding. This work is also related to optimizing projection matrices introduced in [9].

Our approach can also incorporate additional convex constraints on $\mathbf{A}$. To illustrate our algorithm, we use a symmetric and non-smooth version of the MLP (1) in the manner of [4, 5]. We then use the Nesterov smoothing approach to obtain a smooth cost that has approximation guarantees to the original problem, followed by Burer-Monteiro splitting [10] with quasi-Newton enhancements. Even without the sparsity constraints, our algorithmic approach is novel and leads to improved results over the previous state-of-the-art.

The paper is organized as follows. Section 2 sets up the notation used in the paper and introduces the needed definitions; while Section 3 describes the problem and its reformulation into an appropriate optimization framework. Section 4 states the algorithm and its theoretical background; and its followed by Section 5 which showcases the experimental results. Section 6 concludes the paper.

2. PRELIMINARIES

Notation: We denote integer scalars by lowercase letters (e.g., i, k, r, p), real scalars by lowercase Greek letters (e.g., α, δ, λ), sets by uppercase calligraphic letters (e.g., $\mathcal{S}$), vectors by lowercase boldface letter (e.g., $\mathbf{x}$) and matrices by uppercase boldface letter (e.g., $\mathbf{X}$). We denote $\mathbb{S}_+^{N\times N}$ as the set of positive semidefinite (PSD) matrices. The usual ℓ_1 norm and ℓ_0 pseudo-norm (number of non-zero entries) are extended to matrices by reshaping the matrix to a vector.

In Section 3, we reformulate (1) into a problem that is tightly connected with learning Johnson-Lindenstrauss (JL) embeddings or restricted isometry property (RIP) dimensionality reduction. A matrix $\mathbf{\Phi} \in \mathbb{R}^{r\times N}$ is a JL embedding of p points, $\mathcal{X} := \{\mathbf{x}_i\}_{i=1}^p$, if $r = \mathcal{O}(\log p)$ and there exists a positive constant $\delta > 0$ such that the following relations holds:

$$(1-\delta)\|\mathbf{x}_i-\mathbf{x}_j\|_2^2 \le \|\mathbf{\Phi}(\mathbf{x}_i-\mathbf{x}_j)\|_2^2 \le (1+\delta)\|\mathbf{x}_i-\mathbf{x}_j\|_2^2, \quad (2)$$

for every $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, $\mathbf{x}_i \ne \mathbf{x}_j$ [11, 12]. Using the definition of the metric in Section 1, we can rewrite (2) as follows:

$$(1-\delta)\|\mathbf{x}_i-\mathbf{x}_j\|_2^2 \le \|\mathbf{x}_i-\mathbf{x}_j\|_{\mathbf{B}}^2 \le (1+\delta)\|\mathbf{x}_i-\mathbf{x}_j\|_2^2, \quad (3)$$

where $\mathbf{B} = \mathbf{\Phi}^T\mathbf{\Phi}$. Without loss of generality, to simplify the algebra, we are going to enforce the similarity or dissimilarity constraints on normalized differences of data points:

Definition 1. *Given $\mathcal{X} \subset \mathbb{R}^N$ we define the set of secants vectors of points $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ with $\mathbf{x}_i \ne \mathbf{x}_j$ as:*

$$\mathcal{S}(\mathcal{X}) := \left\{\mathbf{v}_{ij} = \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|_2}\right\}. \quad (4)$$

3. PROBLEM DESCRIPTION

In this section, we set up the basic optimization problem that reveals the computational difficulty of (1). In general, the MLP (1) considers relationships between points based on their pairwise distances. Hence, we would require that the metric $\mathbf{B}$ preserves the pairwise distances of the points in $\mathcal{D}$ up to a distortion parameter δ as in (3) to yield a more stringent constraint for (1) while ignoring this requirement for points in $\mathcal{S}$. However, we set up a symmetric problem which then uses RIP in the manner of [4, 5] that can be adjusted depending on the individual application.

In this symmetric formulation, using secant vectors, (3) simplifies to $|\mathbf{v}_{ij}^T\mathbf{B}\mathbf{v}_{ij} - 1| \le \delta$ for $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ with $\mathbf{x}_i \ne \mathbf{x}_j$. Re-indexing the $\mathbf{v}_{ij}$ to $\mathbf{v}_l$ for $l = 1, \dots, M$, where $M = \binom{p}{2}$, we form the set of M secant vectors $\mathcal{S}(\mathcal{X}) = \{\mathbf{v}_1, \dots, \mathbf{v}_M\}$ into an $N \times M$ matrix $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_M]$. Then we learn a $\mathbf{B}$ that minimizes $|\mathbf{v}_l^T\mathbf{B}\mathbf{v}_l - 1|$ (for a slight abuse of notation we will refer to this quantity as δ) over all $\mathbf{v}_l \in \mathcal{S}(\mathcal{X})$. It is known [13, 14] that the rank of $\mathbf{B}$ can be bounded as follows:

$$\operatorname{rank}(\mathbf{B}) \le \left\lceil \frac{\sqrt{8|\mathcal{S}(\mathcal{X})|+1}-1}{2} \right\rceil. \quad (5)$$

Let us define a linear transform $\boldsymbol{\mathcal{A}} : \mathbb{S}_+^{N\times N} \to \mathbb{R}^M$ as $\boldsymbol{\mathcal{A}}(\mathbf{B}) := \operatorname{diag}\left(\mathbf{V}^T\mathbf{B}\mathbf{V}\right)$, where $\operatorname{diag}(\mathbf{H})$ denotes a vector of the entries of the principal diagonal of the matrix $\mathbf{H}$.

Learning the $\mathbf{B}$ that minimizes δ for our symmetric MLP (1) therefore becomes the following non-convex problem:

$$\begin{aligned} &\min_{\mathbf{B}} \ \|\boldsymbol{\mathcal{A}}(\mathbf{B}) - \mathbf{1}_M\|_\infty \\ &\text{subject to} \quad \mathbf{B} \succeq 0, \quad \operatorname{rank}(\mathbf{B}) = r. \end{aligned} \quad (6)$$

Instead of learning $\mathbf{B}$ directly, we instead opt to learn its factors, i.e., $\mathbf{B} = \mathbf{A}\mathbf{A}^T$, which is also known as Burer-Monteiro splitting [10]:

$$\min_{\mathbf{A}\in\mathbb{R}^{N\times r}} \ \|\boldsymbol{\mathcal{A}}\left(\mathbf{A}\mathbf{A}^T\right) - \mathbf{1}_M\|_\infty \quad (7)$$

The advantages of the non-convex formulation (7) is two fold: (i) it reduces storage space since the optimization variable lives in a much lower dimensional space (i.e., $N \times r \ll N \times N$), and (ii) it enables us to add additional constraints and regularizers on the factors $\mathbf{A}$ directly.

For instance, while the rank constraints can be achieved by constraining that the dimension of $\mathbf{A}$ be $N \times r$, we can also consider adding an ℓ_0-norm constraint as well as an ℓ_1-regularizer term (as in [15]) to have the following more general formulation.

$$\begin{aligned} &\min_{\mathbf{A}\in\mathbb{R}^{N\times r}} \ \|\boldsymbol{\mathcal{A}}\left(\mathbf{A}\mathbf{A}^T\right) - \mathbf{1}_M\|_\infty + \lambda\|\mathbf{A}\|_1 \\ &\text{subject to:} \quad \|\mathbf{A}\|_0 \le \sigma. \end{aligned} \quad (8)$$

The ℓ_0-norm constraint enables us to specify *a priori* the sparsity of the output of our algorithm. It is also possible to add a constraint on the Frobenius norm of $\mathbf{A}$ directly.

Note that the approach in [4] solves a convex relaxation of (6), where the rank constraint is replaced by a nuclear norm constraint. That solution works in the $N \times N$ space and requires eigendecompositions. In contrast, we do not do a eigendecomposition in our approach to recover $\mathbf{A}$. Moreover, we can strictly enforce rank constraints while minimizing δ.

Unfortunately, our problem (8) is not smooth and hence we only have access to a subgradient of the objective. Instead, we consider the following smoothed version:

$$\begin{aligned} &\min_{\mathbf{A}\in\mathbb{R}^{N\times r}} \ f(\boldsymbol{\mathcal{A}}\left(\mathbf{A}\mathbf{A}^T\right) - \mathbf{1}_M) + \lambda\|\mathbf{A}\|_1 \\ &\text{subject to:} \quad \|\mathbf{A}\|_0 \le \sigma. \end{aligned} \quad (9)$$

The simplest choice of a smoothing function is $f(\mathbf{z}) = \|\mathbf{z}\|_2^2$ which can be interpreted as penalizing the *average* δ for all secant vectors. In this paper, we focus on the smoothing function choice $f(\mathbf{z}) = f_\mu(\mathbf{z}) = \mu\log(\sum_{i=1}^M e^{z_i/\mu} + e^{-z_i/\mu})$ since $\forall\, \mathbf{z} \in \mathbb{R}^M$, $\lim_{\mu\to 0} f_\mu(\mathbf{z}) = \|\mathbf{z}\|_\infty$. Furthermore, f_μ is C^∞ on $\mathbb{R}^M$ and its gradient (in $\mathbf{z}$) is Lipschitz continuous with constant μ^{-1} (e.g., [16]), though unfortunately the gradient is not Lipschitz in $\mathbf{A}$ (when $\mathbf{z} = \boldsymbol{\mathcal{A}}(\mathbf{A}\mathbf{A}^T) - \mathbf{1}_M$).

4. PROXIMAL GRADIENT METHOD

The proximal gradient method (aka forward-backward splitting method) generalizes the basic projected gradient method. This method is used on problems of the type:

$$\min_{\mathbf{A}} f(\mathbf{A}) + \phi(\mathbf{A}) \tag{10}$$

under the assumption that f has a gradient. By allowing ϕ to take on infinite values, this can model constraints. For example, the constraint $\mathbf{A} \in C$ is modeled using the indicator function $\iota_C(\mathbf{A}) = \begin{cases} 0 & \mathbf{A} \in C \\ +\infty & \mathbf{A} \notin C \end{cases}$. Our main tool is the proximity operator "prox", defined as:

Definition 2. *For a fixed $\tau > 0$, the proximity operator of a function ϕ is the map*

$$\text{prox}_{\tau\phi(\cdot)}(\mathbf{Y}) \in \underset{\mathbf{A}}{\text{argmin}}\ \tau\phi(\mathbf{A}) + \frac{1}{2}\|\mathbf{A} - \mathbf{Y}\|_2^2.$$

The prox is unique and non-expansive if ϕ is convex, proper and lower semi-continuous. Note that this reduces to the projection onto C when $\phi = \iota_C$. The proximal gradient method is listed in Algorithm 1; see [17] for variants, applications and convergence results. We also include a Nesterov accelerated variant due to [18] that requires almost no extra computation and has much faster empirical convergence.

Algorithm 1 General proximal gradient method with Nesterov acceleration

```
Require: stepsize $\tau$, prox function $\text{prox}_\phi$, gradient function
    $\nabla f(\cdot)$, initial point $\mathbf{A}_0$
1: $\mathbf{Y} \leftarrow \mathbf{A}_0$
2: If using Nesterov acceleration, $\alpha_k \equiv \frac{k}{k+3}$, otherwise,
   $\alpha_k \equiv 0$
3: for $k = 1, 2, \ldots$ do
4:     $\mathbf{A}_{k+1} = \text{prox}_{\tau\phi(\cdot)}(\mathbf{Y} - \tau\nabla f(\mathbf{Y}))$
5:     $\mathbf{Y} = \mathbf{A}_{k+1} + \alpha_k(\mathbf{A}_{k+1} - \mathbf{A}_k)$
6: end for
```

Convergence. When ∇f is Lipschitz continuous with parameter L, the step-size is $\tau \leq 1/L$, and if both f and ϕ are convex, then the sequence $(\mathbf{A}_k)$ converges to a global minimizer of (10). Unfortunately, both f and ϕ in our model (9) are non-convex and ∇f is not Lipschitz. However, if $(\mathbf{A}_k)$ is bounded, then ∇f is Lipschitz restricted to this set, and also by the boundedness of the sequence and by some technical regularity of the function, the work of [19] guarantees convergence to a local stationary point. In this sense, if the algorithm has converged, we can *a posteriori* use the boundedness of the sequence to show that the limit point is a local stationary point. This is a slightly different guarantee than that proved in [20].

Computing the gradient. We restrict now this analysis to the log-sum-exp case since the derivation for the quadratic case is even simpler. Define the log-sum-exp function $\text{lse}(\mathbf{z}) = \mu \log(\sum_i e^{z_i/\mu})$ which, for a fixed $\mathbf{z}$, converges to $\max_i z_i$ as $\mu \to 0^+$. To approximate $\|\mathbf{z}\|_\infty$ we use $\text{lse}(D(\mathbf{z}))$ where $D(\mathbf{z}) = (\mathbf{z}, -\mathbf{z})$ (with adjoint $(D^*(\mathbf{w}))_i = w_i - w_{i+M}$ for $\mathbf{w} \in \mathbb{R}^{2M}$). Altogether, we have

$$f(\mathbf{A}) = \text{lse}(D(\boldsymbol{\mathcal{A}}(\mathbf{A}\mathbf{A}^T) - \mathbf{b})).$$

Note that the adjoint of $\boldsymbol{\mathcal{A}}(\mathbf{z})$ is $\boldsymbol{\mathcal{A}}^*(\mathbf{z}) = \mathbf{V}\text{diag}(\mathbf{z})\mathbf{V}^T$, and $(\nabla\,\text{lse})_i = (\sum_{i'} e^{x_{i'}/\mu})^{-1} e^{x_i/\mu}$. Viewing f as the composition of four functions (lse, D, $\boldsymbol{\mathcal{A}}(\cdot) - \mathbf{b}$ and $\mathbf{A} \mapsto \mathbf{A}\mathbf{A}^T$) and using the chain rule gives

$$\nabla f(\mathbf{A}) = 2\mathbf{V}\text{diag}(D^*(\mathbf{z}))\mathbf{V}^T\mathbf{A} \tag{11}$$

and $\mathbf{z} = \nabla\,\text{lse}(D(\boldsymbol{\mathcal{A}}(\mathbf{A}\mathbf{A}^T) - \mathbf{b}))$.

Computational Complexity. Compared to previous approaches, a major benefit of our approach is much better computational complexity if implemented carefully. The major computational bottleneck of the entire algorithm is in computing $\boldsymbol{\mathcal{A}}(\mathbf{B})$ and in $\boldsymbol{\mathcal{A}}^*(\mathbf{z})$. To be efficient, we (i) exploit the fact that $\mathbf{B} = \mathbf{A}\mathbf{A}^T$, and (ii) never explicitly form $\boldsymbol{\mathcal{A}}^*(\mathbf{z})$ as a matrix but rather treat this as an implicit operator that acts on other matrices.

Specifically, $\boldsymbol{\mathcal{A}}(\mathbf{B}) = \text{diag}(\mathbf{V}^T\mathbf{A}\mathbf{A}^T\mathbf{V}) = \text{diag}(\bar{\mathbf{V}}^T\bar{\mathbf{V}})$ for $\bar{\mathbf{V}} = \mathbf{A}^T\mathbf{V}$. This simplifies to taking the norms of the rows of $\bar{\mathbf{V}}$, and altogether requires $\mathcal{O}(rNM)$ flops, compared to naively computing $\mathbf{B} = \mathbf{A}\mathbf{A}^T$ and then taking $\boldsymbol{\mathcal{A}}(\mathbf{B})$ which requires $\mathcal{O}(MN^2)$ flops (recall $r \ll N \ll M$). For $\boldsymbol{\mathcal{A}}^*$, we appeal to (11) directly and compute $\mathbf{V}^T\mathbf{A}$ and then compute the rest of the multiplies, and again we require $\mathcal{O}(rNM)$ flops, compared to $\mathcal{O}(N^2M)$ naively. Methods based on convex relaxations do not see these numerical speed-ups since their variables $\mathbf{B}$ are generally rank N not r. Furthermore, at every iteration these convex methods require SVDs of $\mathbf{B}$ which can cost up to $\mathcal{O}(N^3)$.

Computing the proximity operator. By making use of the indicator function we can write the non-smooth portion of (9) as $\phi(\mathbf{A}) = \lambda\|\mathbf{A}\|_1 + \iota_{\{\mathbf{A} | \|\mathbf{A}\|_0 \leq \sigma\}}$. In the special case when $\lambda = 0$, the proximity operator of ϕ is just the (possibly non-unique) projection onto the top σ largest (in magnitude) entries. In the other special case when $\sigma \geq Nr$ so that the ℓ_0 term constraint has no effect, the proximity operator of ϕ is just soft-thresholding. In the general case, the proximity operator at the point $\mathbf{z}$ is calculated by first soft-thresholding $\mathbf{z}$ to get $\hat{\mathbf{z}}$ and then calculating $\lambda\tau|z_i| + \frac{1}{2}(\hat{z}_i - z_i)^2$ and choosing the top σ components that minimize this (i.e., by sorting). It is even possible to avoid a sort, but implementation details are unimportant since this step is much cheaper than computing the gradient.

Without sparsity constraints. If we drop the sparsity constraint $\|\mathbf{A}\|_0 \leq \sigma$ and the ℓ_1 regularizer, the objective is smooth and unconstrained. In this case, our code uses L-BFGS [21] as implemented in `minFunc` package [22] and using the gradient described above. Our experiments show L-BFGS is slightly faster than the Nesterov accelerated first-order algorithm, and it also has the advantage that it does not require an accurate initial step-size estimate.

5. EXPERIMENTAL RESULTS

We determine the quality of our approximate solution to the MLP (1) by how small δ is. In the first set of experiments we investigate how δ varies with the rank of the matrix we learn using both a set of synthetic data and a set of images of motorbikes [23]. The synthetic data set is the manifold data set used in [4], composed of translating white squares in a black background. We generate manifold images sizes of 40×40 pixels and resize grayscale images of the motorbikes to also be 40×40 pixels, resulting in points of dimension $N = 1600$.

We call the implementation of Algorithm 1 for metric learning the **Fast Adaptive Metric Learning** (FAML) algorithm. Using FAML we learn sparse (with sparsity σ fixed at 10%, and $\lambda = 0$) and dense factors of a distance metric $\mathbf{X}$ of the secant vectors of these points; we initialize the dense version with the PCA matrix, and the sparse version using the dense solution. Figure 1 shows how δ varies with the rank (number of rows) of the dense ("FAML - dense") and sparse ("FAML - sparse") factors of the matrix we learned, and compares to a random projection matrix ("Gaussian") and a PCA metric of the same data sets ("PCA"). Both our sparse and dense matrices have smaller δ. PCA is performed in a scalable way using a randomized SVD [24].

We also compare the matrices we learn to those matrices learned using NuMax and NuMax_CG [4] in terms of smallness of rank and δ. Precisely, we fixed a rank and run FAML to obtain a δ, then use this δ as an input for NuMax and NuMax_CG and record the rank they output. The right panel plot of Figure 1 show that both versions of FAML outperform both versions of NuMax in the low-rank regime, whereas NuMax does better when the rank r is larger. Note that it is always possible to initialize FAML-dense using the NuMax or NuMax_CG solution if we are willing to spend extra time.

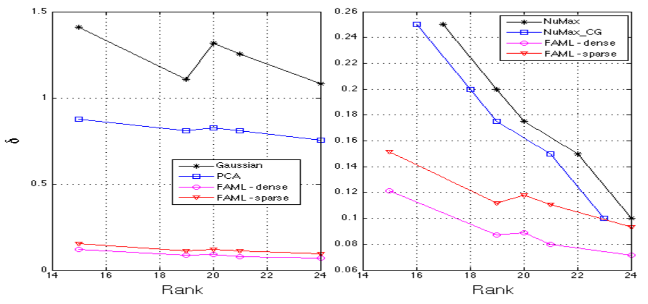

Fig. 1. Plots of the relationship between the δ (mean over 10 trials) and the rank of the matrices we learn for 2775 secants, with N fixed. Using the motorbike images of $N = 40 \times 40$ *Left panel:* a comparison with Gaussian and PCA, *Right panel:* a comparison with NuMax and NuMax_CG.

In the above-mentioned experiment we compare the time take by FAML to NuMax. The left panel of Figure 2 shows that either version of FAML gives smaller δ in shorter time than NuMax and NuMax_CG. Similarly, given an input rank, FAML converges faster than NuMax and NuMax_CG, as shown in the right panel of Figure 2.

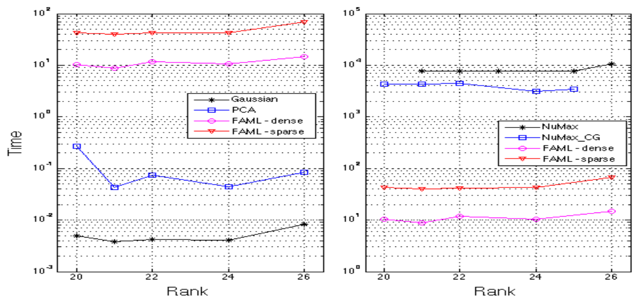

Fig. 2. Speed comparison in terms of time taken to get a target rank (or equivalently for a target δ) *Left panel:* a comparison with Gaussian and PCA, *Right panel:* a comparison with NuMax and NuMax_CG.

We also explore the high-N case by taking the full resolution motorbike images from [23], so the dimension becomes $N = 163 \times 261 = 42543$. We select 50 points and generate all 1225 possible secants. Figure 3 shows we can achieve moderate δ for low ranks, and do much better than PCA. The NuMax and NuMax_CG algorithms were run on this data but did not work because they require forming $N \times N$ dense matrices which do not fit in the 8 GB of RAM of the computer. Combined with the results of Figure 1, this suggests FAML outperforms NuMax when N is large or r is small.

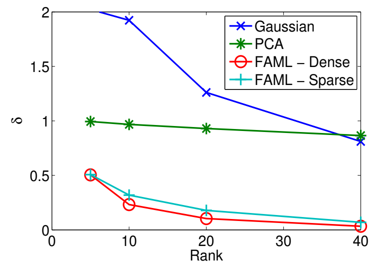

Fig. 3. Motorbike data with $N = 42543$.

6. CONCLUSION

We presented an optimization formulation of the metric learning problem that can handle sparsity and rank constraints. The enforcement of sparsity appears to be novel and may have impact in applications that require sparse matrices (e.g., Low-Density Parity Check codes) for speed or hardware implementation reasons. Our code is low-memory due to careful construction of the algorithm and implementation, and if it converges, it does so rapidly due to Nesterov acceleration and L-BFGS. In either the low-rank or high dimension regime, the method outperforms the NuMax algorithms. For further research we will apply NuMax_CG's column generation techniques in order to handle more secant vectors.

7. REFERENCES

[1] L. Yang and R. Jin, "Distance metric learning: A comprehensive survey," *Michigan State Universiy*, pp. 1–51, 2006.

[2] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *International Conf. Machine Learning (ICML)*, Corvallis, OR, 2007.

[3] B. Kulis, M. Sustik, and I. Dhillon, "Learning low-rank kernel matrices," in *Proceedings of the 23rd International conference on Machine learning (ICML)*. ACM, 2006, pp. 505–512.

[4] C. Hegde, A.C. Sankaranarayanan, W. Yin, and R.G. Baraniuk, "A convex approach for learning near-isometric linear embeddings," *preparation, August*, 2012.

[5] A. Sadeghian, B. Bah, and V. Cevher, "Energy-aware adaptive bi-Lipschitz embeddings," in *10th International Conference on Sampling Theory and Applications (SampTA)*. 2013, EURASIP.

[6] E. Grant, C. Hegde, and P. Indyk, "Nearly optimal linear embeddings into very low dimensions," in *IEEE GlobalSIP Symposium on Sensing and Statistical Inference*, December 2013.

[7] A. S. Lan, A. E. Waters, C. Studer, and R. G. Baraniuk, "Sparse factor analysis for learning and content analytics," *arXiv preprint arXiv:1303.5685*, 2013.

[8] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *IEEE Intl Conf. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 6655–6659.

[9] M. Elad, "Optimized projections for compressed sensing," *IEEE Trans. Signal Processing*, vol. 55, no. 12, pp. 5695–5702, 2007.

[10] S. Burer and R.D.C. Monteiro, "Local minima and convergence in low-rank semidefinite programming," *Math. Prog. (series A)*, vol. 103, no. 3, pp. 427–444, 2005.

[11] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemporary mathematics*, vol. 26, no. 189-206, pp. 1, 1984.

[12] E. J. Candes and T. Tao, "Decoding by linear programming," *Information Theory, IEEE Transactions on*, vol. 51, no. 12, pp. 4203–4215, 2005.

[13] A. I. Barvinok, "Problems of distance geometry and convex properties of quadratic maps," *Discrete & Computational Geometry*, vol. 13, no. 1, pp. 189–202, 1995.

[14] G. Pataki, "On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues," *Mathematics of Operations Research*, vol. 23, no. 2, pp. 339–358, 1998.

[15] A. Kyrillidis and V. Cevher, "Combinatorial selection and least absolute shrinkage via the CLASH algorithm," in *IEEE Intl. Symp. Information Theory*. IEEE, 2012, pp. 2216–2220.

[16] A. Beck and M. Teboulle, "Smoothing and first order methods: A unified framework," *SIAM J. Optimization*, vol. 22, pp. 557–580, 2012.

[17] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *SIAM Multiscale Model. Simul.*, vol. 4, no. 4, pp. 1168–1200, 2005.

[18] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. on Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[19] H. Attouch, J. Bolte, and B.F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss-Seidel methods," *Math. Prog.*, pp. 1–39, 2011.

[20] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *Math. Prog. Comp.*, pp. 1–29, 2010.

[21] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Math. Comp.*, vol. 25, no. 151, pp. 773–782, 1980.

[22] M. Schmidt, *minFunc software package*, 2012, `http://www.di.ens.fr/~mschmidt/Software/minFunc.html`.

[23] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories," in *IEEE CVPR 2004, Workshop on Generative-Model Based Vision*, 2004.

[24] N. Halko, P.G. Martinsson, and J.A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.