Grassmann Manifold Optimization for Fast L_1-Norm Principal Component Analysis

Breton Minnehan, *Member, IEEE*, and Andreas Savakis, *Senior Member, IEEE*

***Abstract*—In this letter, we propose a fast Grassmann manifold optimization method for L_1-norm based principal component analysis (GM-L_1-PCA). Our approach is a two-step iterative cost-minimization and manifold retraction technique that efficiently finds all principal components simultaneously. We perform complexity analysis and show that GM-L_1-PCA achieves a significant reduction in processing time while obtaining comparable or better results to current state-of-the-art L_1-PCA methods. We further demonstrate the improvement of GM-L_1-PCA technique over L_2-PCA on a dataset of facial imagery corrupted with outlying data points. Our experiments show that GM-L_1-PCA is computationally more efficient and produces results with lower reprojection error than previous methods. Furthermore, the processing time of our approach is relatively independent of dataset size and well suited for various big-data problems commonly encountered today.**

***Index Terms*—Robust principal component analysis, L_1-PCA, Grassmann manifold optimization, dimensionality reduction, big data.**

I. Introduction

MANY of the techniques employed in deep learning today trace their roots back to the field of signal processing. Notable examples include gradient descent optimizers [1], 2-D convolutions [2] and batch normalization [3]. In this letter, we propose a method for linear dimensionality reduction that combines the Grassmann Manifold (GM) optimization method first proposed in [4] with a computational framework developed for deep learning. The Grassmann manifold $\mathcal{G}^{D\times K}$ is the set of all rank-K subspaces in $\mathbb{R}^D$. Since the GM includes all possible orthogonal projections of the data, the Principal Component Analysis (PCA) solution resides on a Grassmann manifold. This observation allows us to use the GM optimization technique for the challenging L_1-norm based Principal Component Analysis (L_1-PCA) problem [5], [6].

Outlier invariant dimensionality reduction is a useful tool for many applications from signal analysis to classification and machine learning problems. We focus on L_1-PCA, because L_2-PCA is highly influenced by outliers in the training set [7], whereas L_1-PCA has been found to be outlier insensitive. An illustrative example is presented in Fig. 1 and discussed in more detail in the results section. Current L_1-PCA methods, however, have a high computational cost. Obtaining a fast solution to the L_1-PCA problem would be beneficial in many applications utilizing principal component analysis and useful in big data applications such as face recognition [8], [9], video background removal [10] and image outlier detection [11].

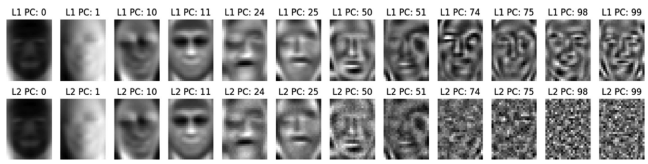

Fig. 1. Example principal components with the proposed GM-L_1-PCA (Top) method and L_2-PCA (Bottom) resulting from training on a dataset of facial images with 10% of the sample corrupted with uniform noise. Higher order principal components capture less noise in the L_1-norm formulation.

II. Background

Much of the research in PCA has dealt with methods based on the L_2-Norm version [12]–[14]. Due to the properties of the L_2-Norm, there are several equivalent optimization formulations for L_2-PCA [14]. However, only two of them are pertinent to this work. First, as originally proposed in [12], an orthogonal projection $\mathbf{R}$ is learned in order to project the D dimensional input data $\mathbf{X}$ to a K dimensional representation, where $D > K$, such that the L_2-Norm between the original data and its reprojection is minimized, as represented in

$$\mathbf{R}_{L_2} = \underset{\mathbf{R}\in\mathbb{R}^{D\times K},\mathbf{R}^T\mathbf{R}=\mathbf{I}_K}{\arg\min} (\|\mathbf{X}-\mathbf{X}\mathbf{R}\mathbf{R}^T\|_2^2) \tag{1}$$

The second formulation of interest for L_2-PCA, given in Eq. (2), aims to maximize the energy of the data in the projection space. The solution for L_2-PCA can be obtained by decomposing the covariance matrix of the input data, $\mathbf{X}\mathbf{X}^T$, and taking the eigenvectors corresponding to the K highest eigenvalues.

$$\mathbf{R}_{L_2} = \underset{\mathbf{R}\in\mathbb{R}^{D\times K},\mathbf{R}^T\mathbf{R}=\mathbf{I}_K}{\arg\max} (\|\mathbf{X}\mathbf{R}\|_2^2) \tag{2}$$

If the data contains outliers from a significantly different distribution, the noise has a destructive impact on the principal component. This effect is more apparent in the case of the L_2-PCA because of the squared magnitude in the L_2-Norm, as explained in [7].

One approach to minimizing the impact of outlying data points is to replace the L_2-Norm with the L_1-Norm when calculating the principal components (PCs). The drawback to using the L_1-Norm is that the reprojection theorem does not hold under the L_1-Norm, thus the Singular Value Decomposition (SVD)

Manuscript received July 11, 2018; revised November 29, 2018; accepted November 29, 2018. Date of publication December 13, 2018; date of current version December 24, 2018. This work was supported in part by AFOSR Dynamic Data Driven Applications Systems (DDDAS) program. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Marco Felipe Duarte. *(Corresponding author: Breton Minnehan.)*

The authors are with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: blm2144@rit.edu; andreas.savakis@rit.edu).

Digital Object Identifier 10.1109/LSP.2018.2886742

method used for the L_2 norm can no longer be used. Additionally, the corollary L_1 optimization problems, in Equations (3) and (4), are not equivalent.

$$\mathbf{R}_{L_1} = \underset{\mathbf{R}\in\mathbb{R}^{D\times K},\mathbf{R}^T\mathbf{R}=\mathbf{I}_K}{\arg\min} (\|\mathbf{X}-\mathbf{X}\mathbf{R}\mathbf{R}^T\|_1) \quad (3)$$

$$\mathbf{R}_{L_1} = \underset{\mathbf{R}\in\mathbb{R}^{D\times K},\mathbf{R}^T\mathbf{R}=\mathbf{I}_K}{\arg\max} (\|\mathbf{X}\mathbf{R}\|_1) \quad (4)$$

It is not known which optimization formulation is best suited for L_1-PCA. One of the advantages of our approach is that it can be used with either of the above loss functions. Some works have focused on minimizing the reprojection error in (3), for example [6], [15]–[17]. However, due to the non-smooth error surface, an optimal solution has not yet been developed. The L_1 energy maximization formulation, Eq. 4, has been demonstrated to be more tractable. Two optimal solutions have been developed in [18], [19] with computational complexity $\mathcal{O}(2^{NK})$ and $\mathcal{O}(N^{rank(\mathbf{X})K-K+1})$, respectively, where N is the number of data points. More recently a sub-optimal but more efficient method for L_1-PCA was proposed in [20]. In this work, Markopoulos *et al.* develop a method which greedily searches for each L_1 principal component using bit flipping to find the components that maximize the projection energy.

III. Methodology

The proposed Grassmann Manifold optimization for L_1-PCA (GM-L_1-PCA) is inspired by a general dimensionality reduction framework using GM optimization [4]. Cunningham and Ghahramani demonstrate in [4] that GM optimization can be used to solve for all of the L_2 PCs simultaneously using a two-step iterative objective-optimization and manifold-retraction process. They experimentally show that the GM retraction method produces identical results to the optimal SVD-decomposition method.

In this letter, we propose a deep learning formulation that uses gradient descent (SGD) with momentum [21] during backpropagation, instead of the line-search optimization method along the manifold tangent used in [4]. We utilize the Tensorflow deep learning framework [22] to perform back-propagation with automatic differentiation without the need to first obtain a closed form solution for the symbolic derivatives of the objective functions. Thus, our framework allows for more diverse objective functions, such as those in Equations (3) or (4). Furthermore, SGD with momentum has the ability to avoid local minima on highly non-convex error surfaces, which results in better convergence compared to SGD [21].

The pseudo-code for GM-L_1-PCA is given in Algorithm 1. The gradients for weight values in R are calculated using automatic differentiation, which systematically applies the chain rule. This results in the updated matrix $\bar{\mathbf{R}}_i = \mathbf{R}_i + \mathbf{Z}_i$, where $\mathbf{Z}_i = \lambda\mathbf{Z}_{i-1} - \nabla_{\mathbf{R}_i} l$. A visual depiction of the two step iterative GM optimization process is shown in Fig. 2. In the first step, the projection matrix is optimized by back-propagating the loss of the objective function using gradient descent. This is done for each iteration i by taking a step, $\mathbf{Z}_i$, in the direction of the loss gradient at the current location $\mathbf{R}_i$, where λ is the momentum parameter, $f_{\mathbf{X}}$ is the loss function and l is the loss for the the data $\mathbf{X}$ at current location $\mathbf{R}_i$. However, it is likely that any step $\mathbf{Z}_i$ will result in $\bar{\mathbf{R}}_i$ leaving the Grassmann Manifold, which means that its components are no longer orthogonal. Thus, in order to preserve orthogonality, after each optimization iteration the updated matrix must be retracted to the manifold by a retraction operation $r_R(\bar{\mathbf{R}}_i)$. The goal of the retraction step is to find the closest subspace on the manifold to the updated matrix $\bar{\mathbf{R}}_i$. Fortunately, because of the unitary invariance of the Frobenius norm, SVD can be used to find the closest subspace on the manifold as, $r_R(\bar{\mathbf{R}}_i) = \mathbf{U}_i\mathbf{V}_i^T$, given $(\mathbf{R}_i + \mathbf{Z}_i) = \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{V}_i^T$, where $\mathbf{U}_i$ and $\mathbf{V}_i^T$ are orthogonal $D\times K$ and $K\times K$ matrices, respectively, and $\mathbf{\Sigma}_i$ is the matrix of singular values. This results in the update step $\pi_{\mathbf{R}_i}$ along the Grassmann manifold to the matrix $\mathbf{R}_{i+1}$. Because the GM is locally linear, the gradient step $\mathbf{Z}_i$ is larger in magnitude than the retraction step $r_R(\bar{\mathbf{R}}_i)$ and contributes most heavily to the update step $\pi_{\mathbf{R}_i}$ along the manifold. By updating the matrix $\mathbf{R}_i$ with a step in the direction that minimizes the loss, l, GM-L_1-PCA converges to a local minimum since the loss is lower bounded by zero, assuming that the step size is small and the surface is locally smooth as in other gradient descent methods, [15], [23].

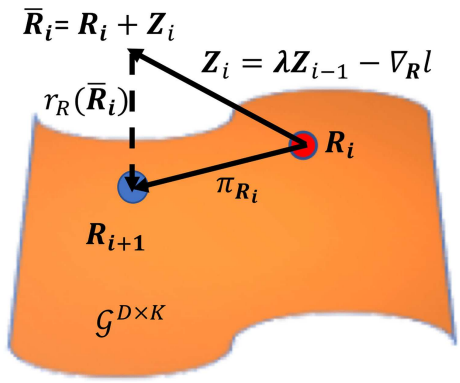

Fig. 2. Visualization of the GM optimization process to update the projection matrix $\mathbf{R}$ for the ith iteration in Algorithm 1. The first step is to update $\mathbf{R}$ along the direction of the gradient of the loss. The second step retracts the updated projection back to the closest subspace on the Grassmann manifold. This process is repeated for a number of iteration specified by the user.

Algorithm 1: Linear Dimensionality Reduction Using GM-L_1-PCA with Iterative Gradient Descent and Retraction on the Grassmann Manifold. The Loss Function is Based on Eq. (3), but Our Method Also Allows Using the Loss in Eq. (4).

Data: $\mathbf{X} \in \mathbb{R}^{N\times D}$
Result: Locally Optimal $\mathbf{R}$
Initialize $\mathbf{R} \in \mathcal{G}^{D\times K}$, $\mathbf{Z}_0 = 0$, $iter$=max iterations;
for $i < iter$ **do**
  $l = \|\mathbf{X} - \mathbf{X}\mathbf{R}\mathbf{R}^T\|_1$; // Loss from Eq. (3)
  $\mathbf{Z}_i = \lambda\mathbf{Z}_{i-1} - \nabla_{\mathbf{R}_i} l$; // Calculate step $\mathbf{Z}_i$
  $\bar{\mathbf{R}} = \mathbf{R} + \mathbf{Z}_i$; // Update $\mathbf{R}$ with step $\mathbf{Z}$
  $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \bar{\mathbf{R}}$; // Compute SVD for $\bar{\mathbf{R}}$
  $\mathbf{R} = \mathbf{U}\mathbf{V}^T$; // Retract $\bar{\mathbf{R}}$ to $\mathcal{G}^{D\times K}$
end

A. Computational Complexity

We next perform complexity analysis on the GM-L_1-PCA. As shown in Algorithm 1, the proposed framework primarily consists of three parts: projection, back-propagation and manifold retraction. In the data projection step, first the projection matrix is multiplied with its transpose then the resulting matrix is multiplied with the input data, thus the complexity is $\mathcal{O}(DKD+NDD)$. The next step is the backwards propagation of the loss, which includes two substeps: calculating the

loss and calculating the partials of the derivatives of the loss with respect to the weights. The loss is calculated by taking the L1 norm of the projected and original data, thus its complexity is $\mathcal{O}(2ND)$. Once the loss is calculated, the complexity of calculating the partial derivatives is $\mathcal{O}(NDK)$. The last step of the algorithm is the retraction of the weight matrix on the Grassmann manifold. This step is done by performing SVD of the network weights with complexity $\mathcal{O}(D^2K+K^3)$. This leads to the overall computational complexity of our algorithm as $\mathcal{O}(D^2(2K+N)+ND(2+K)+K^3)$, per iteration.

Our complexity analysis above suggests that the three most influential factors in the processing time for GM-L_1-PCA are: the number of iterations of the algorithm, the input dimensions of the data D and the number of components extracted K. This means that the data-set size N plays a relatively small role in the processing time of the algorithm, which makes the GM-L_1-PCA algorithm well-suited for big data applications. This is especially evident when the computational complexity of GM-L_1-PCA is compared to BF-L_1-PCA, with a complexity of $\mathcal{O}(ND\min(N,D)+N^2K^2(K^2+D))$.

In our implementation we initialize the PCs with the results from standard L_2-PCA. If we assume that the L_1 PCs are found in the local neighborhood of the L_2 PCs, this initialization reduces the number of optimization iterations. An alternative would be to perform multiple runs with random initializations and keep the best result. In our experiments, we found that in addition to faster convergence, initializing with the L_2-PCA components resulted in better consistency in the obtained solution compared to random initialization.

IV. Results

We ran two sets of experiments, both designed to demonstrate the improved robustness with respect to noise of L_1-PCA relative to L_2-PCA. In our first set of experiments we use data generated from a set of random Gaussian distributions that were corrupted with outlying samples from a significantly different distribution. In the second set of experiments we artificially corrupt a facial identification dataset by replacing images with white noise. The goal of each experiment is to learn a linear projection to a lower dimensional space that optimally captures the signal in each dataset and not the noise. Thus, in our experiments, we extract the PCs on a dataset corrupted by noise then we test the reconstruction accuracy on a holdout set of data that is uncorrupted. We use the reprojection error of the clean holdout data as the performance metric for this work with the understanding that the method that better reconstructs the uncorrupted test data has learned more representative principal components for the true data without being affected by the noise. All data in these experiments were centered, i.e., made zero-mean for both the training and testing sets.

For all these experiments Eq. (3) was used to optimize $\mathbf{R}$. All experiments were performed on a computer running Red Hat Enterprise Linux Server 7.4 with four Intel Xeon E5-2650 CPUs.

A. Toy Dataset Experiments

For these experiments, we generated data that was sampled from a random 25-dimensional Gaussian distribution. We randomly replace 10% of the training set with values from a significantly different random distribution. By examining the behavior of the algorithms in the presence of data corruption we better understand the impact of noise on the extracted PCs. We examined the performance of the PCA methods, in terms of both speed and reprojection accuracy, relative to the dataset size and number of PCs calculated. For these experiments we generated 100 separate datasets and present the mean and standard deviation across all 100 tests. We compare the reprojection error achieved by our method with the results from a SVD-based implementation of L_2-PCA and the current state-of-the-art BF-L_1-PCA [20].

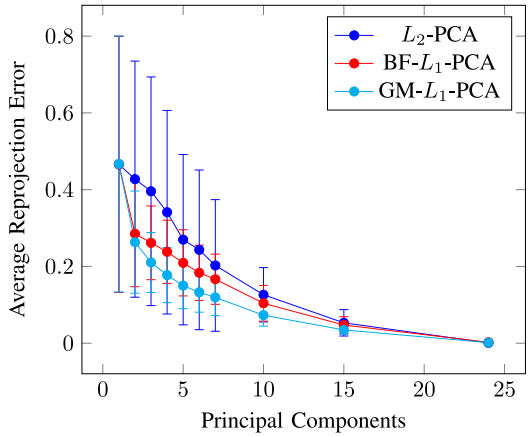

Fig. 3. Average reprojection error comparing three PCA methods, L_2-PCA, BF-L_1-PCA and GM-L_1-PCA, for various numbers of principal components. Mean and standard deviations calculated from 100 runs.

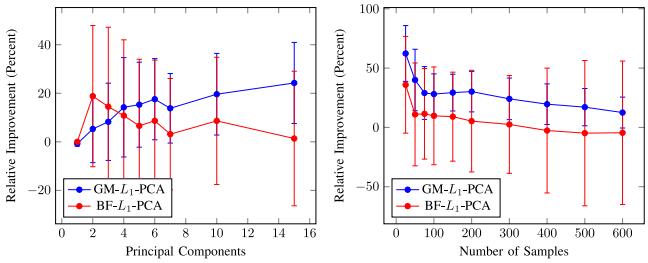

Fig. 4. Left: Percent improvement of proposed GM-L_1-PCA and Bit-Flipping L_1-PCA methods relative to L_2-PCA for various numbers of principal components. Right: Percent improvement of proposed GM-L_1-PCA and Bit-Flipping L_1-PCA methods relative to L_2-PCA for datasets with various numbers of samples. Mean and standard deviations for both experiments were calculated from 100 runs on randomly generated datasets.

In the first set of experiments we used a constant training set size of 400 samples and an uncorrupted test set with 100 samples. We then varied the number of principal components extracted from 1 to 24. The average re-projection error for L_2-PCA, BF-L_1-PCA and GM-L_1-PCA are shown in Fig. 3. It is clear in these plots that both L_1-PCA methods outperform L_2-PCA until 24 principal components are calculated at which point the re-projection error for all methods is extremely low.

In order to highlight the improvement of the L1-PCA methods over L_2-PCA we also plot the relative percent improvement in reprojection error of both L_1-PCA methods relative to L_2-PCA. The improvement of both methods relative to L_2-PCA for varying number of principal components is shown on the left plot of Fig. 4.

For our second set of experiments with the toy dataset the number of extracted principal components was kept constant at 10, and the dataset size was varied from 25 to 600. Again we

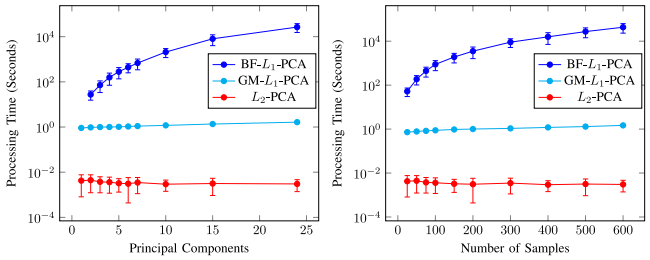

Fig. 5. Results illustrating the computational efficiency of GM-L_1-PCA. Left: Processing time for PCA algorithms run on 400 samples from the toy datasets with varying number of output dimensions between 1–24. Right: Processing time for PCA algorithms run on varying number of samples 25-600 from the toy datasets with a constant output dimension of 10. Time axis is on a logarithmic scale illustrating the large improvement in execution speed for the GM-L_1-PCA.

plot the relative reduction of reprojection error of both L_1-PCA methods compared to L_2-PCA. The results of these experiments are shown in the right plot of Fig. 4. In all instances GM-L_1-PCA has a lower reprojection error than the L_2-PCA and generally improves on the state-of-the-art L_1-PCA method.

In addition to reprojection error we also investigate the impact of the number of extracted components and the dataset size on execution time. The plots in Fig. 5 show the processing time for each method for both sets of experiments on the toy dataset. It is clear in these experiments that GM-L_1-PCA method is much faster than BF-L_1-PCA and can produce L_1-PCA results at a relative constant time irrespective of number of extracted components and the size of the dataset. While GM-L_1-PCA is not as fast as L_2-PCA, it achieves a fast processing time that is modestly increasing, from 0.7 to 1.6 seconds, with respect to both dataset size and number of principal components.

B. Labeled Faces in the Wild Experiments

For our next experiments, we worked with Labeled Faces in the Wild (LFW) [24], a dataset consisting of 13,233 faces. Images were converted to gray-scale and resized to 31×23 pixels for processing. We tested GM-L_1-PCA on a dataset where 10% of the images in the training set are replaced with random white noise. We plot the relative reduction in error of GM-L_1-PCA relative to L_2-PCA in Fig. 7. The reconstruction error shown in the plot is consistent with the common intuition about the behaviour of L1 and L2 based PCA methods in the presence of outliers. When few principal components are extracted both L1 and L2 based methods perform similarly, however, as more compenents are extracted the L2 based method is more impacted by outliers than the GM-L_1-PCA. The roughly 45% reduction in reprojection error achieved by GM-L_1-PCA confirms that L_1-PCA learns principal components that are less impacted by noise.

In addition to quantitative results, we provide visual examples in Fig. 6 for reprojections of uncorrupted test images using GM-L_1-PCA and standard L_2-PCA. These examples show that the L_1 principal components generate projections that have significantly less noise. This is particularly evident in areas of low variance such as the forehead or cheek region. The images in Fig. 1 show various principal components learned by L_1 and L_2 PCA in these experiments. The lower order PCs, 1–50, are similar for both L_1 and L_2-PCA, probably due to the

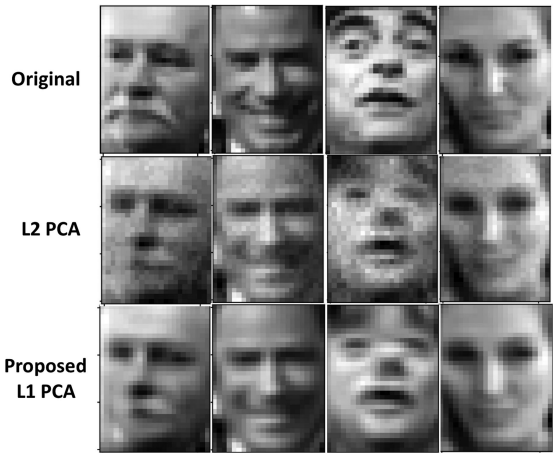

Fig. 6. Reprojection of face images using 100 components of the proposed GM-L_1-PCA and L_2-PCA, both extracted from a corrupted face training set. Top: Original Image. Center: Reprojection using L_2-PCA (the noise is visible in the reprojection). Bottom: Reprojection using GM-L_1-PCA.

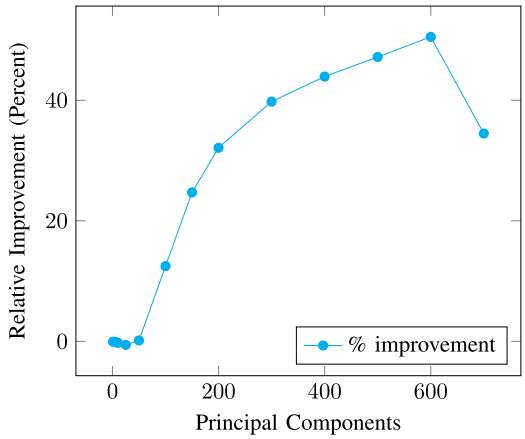

Fig. 7. Relative improvement in reprojection error of the GM-L_1-PCA method versus L_2-PCA.

initialization of L_1-PCA at the L_2-PCA components. The difference is more apparent for the higher order components where L_1-PCA captures more facial structures, such as edges around the eyes and nose, in comparison to the L_2 components.

V. Conclusion

This research presents a Grassmann Manifold optimization framework for L_1 Norm based Principal Component Analysis. We perform complexity analysis and provide experimental results which demonstrate that the proposed method is much faster than current state-of-the-art L_1-PCA methods. Computation time of the GM-L_1-PCA method is not significantly affected by data size which makes it uniquely suited for the big-data problems commonly faced in applications.

Acknowledgment

The authors would like to thank Dr. P. Markopoulos for insightful conversations on L1-PCA and for sharing his code.

REFERENCES

[1] L. Bottou, "Stochastic gradient learning in neural networks," in *Proc. Neuro-Nımes*, 1991, vol. 91, no. 8.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[3] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[4] J. P. Cunningham and Z. Ghahramani, "Linear dimensionality reduction: Survey, insights, and generalizations," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 2859–2900, 2015.

[5] S. Kundu, P. P. Markopoulos, and D. A. Pados, "Fast computation of the L 1-principal component of real-valued data," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 8028–8032.

[6] Q. Ke and T. Kanade, "Robust subspace computation using L1 norm," Comput. Sci. Dept., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. TR-CMU-CS-03-172, 2003.

[7] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *J. ACM*, vol. 58, no. 3, 2011, Art. no. 11.

[8] M. Johnson and A. Savakis, "Fast L1-eigenfaces for robust face recognition," in *Proc. IEEE Western New York Image Signal Process. Workshop*, Nov. 2014, pp. 1–5.

[9] F. Maritato, Y. Liu, S. Colonnese, and D. A. Pados, "Face recognition with L1-norm subspaces," *Proc. SPIE*, vol. 9857, pp. 1–8, 2016. [Online]. Available: https://doi.org/10.1117/12.2224953

[10] Y. Liu and D. A. Pados, "Compressed-sensed-domain L1-PCA video surveillance," *IEEE Trans. Multimedia*, vol. 18, no. 3, pp. 351–363, Mar. 2016.

[11] F. Ju, Y. Sun, J. Gao, Y. Hu, and B. Yin, "Image outlier detection and feature extraction via L1-norm-based 2-D probabilistic PCA," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 4834–4846, Dec. 2015.

[12] K. Pearson, "On lines and planes of closest fit to systems of points in space," *London, Edinburgh, Dublin Phil. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, 1901.

[13] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[14] I. T. Jolliffe, *Principal Component Analysis*. Berlin, Germany: Springer-Verlag, 2002.

[15] Q. Ke and T. Kanade, "Robust L1 norm factorization in the presence of outliers and missing data by alternative convex programming," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 739–746.

[16] J. P. Brooks and J. H. Dulá, "The L1-norm best-fit hyperplane problem," *Appl. Math. Lett.*, vol. 26, no. 1, pp. 51–55, 2013.

[17] J. P. Brooks, J. H. Dulá, and E. L. Boone, "A pure L1-norm principal component analysis," *Comput. Statist. Data Anal.*, vol. 61, pp. 83–98, 2013.

[18] P. P. Markopoulos, G. N. Karystinos, and D. A. Pados, "Some options for L1-subspace signal processing," in *Proc. 10th Int. Symp. Wireless Commun. Syst.*, 2013, pp. 622–626.

[19] P. P. Markopoulos, G. N. Karystinos, and D. Pados, "Optimal algorithms for L1-subspace signal processing," *IEEE Trans. Signal Process.*, vol. 62, no. 19, pp. 5046–5058, Oct. 2014.

[20] P. P. Markopoulos, S. Kundu, S. Chamadia, and D. A. Pados, "Efficient L1-norm principal-component analysis via bit flipping," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4252–4264, Aug. 2017.

[21] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, 1999.

[22] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Des. Implementation*, 2016, pp. 265–283.

[23] H. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*. New York, NY, USA: Springer, 2003.

[24] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Univ. Massachusetts, Amherst, Amherst, MA, USA, Tech. Rep. 07-49, 2007.