Multimed Tools Appl (2016) 75:1481–1493
DOI 10.1007/s11042-014-2004-4

Large-scale paralleled sparse principal component analysis

W. Liu · H. Zhang · D. Tao · Y. Wang · K. Lu

Received: 27 November 2013 / Revised: 25 February 2014 / Accepted: 1 April 2014 /
Published online: 24 April 2014

Abstract Principal component analysis (PCA) is a statistical technique commonly used in multivariate data analysis. However, PCA can be difficult to interpret and explain since the principal components (PCs) are linear combinations of the original variables. Sparse PCA (SPCA) aims to balance statistical fidelity and interpretability by approximating sparse PCs whose projections capture the maximal variance of original data. In this paper we present an efficient and paralleled method of SPCA using graphics processing units (GPUs), which can process large blocks of data in parallel. Specifically, we construct parallel implementations of the four optimization formulations of the generalized power method of SPCA (GP-SPCA), one of the most efficient and effective SPCA approaches, on a GPU. The parallel GPU implementation of GP-SPCA (using CUBLAS) is up to eleven times faster than the corresponding CPU implementation (using CBLAS), and up to 107 times faster than a MatLab implementation. Extensive comparative experiments in several real-world datasets confirm that SPCA offers a practical advantage.

Keywords Sparse principal · Component analysis · Power method · GPU · Large-scale · Parallel method

1 Introduction

Principal component analysis (PCA) [15] is a well-established tool used for data analysis and dimensionality reduction [10] [11] [12] [13] [27] [31]. The goal of PCA is to find a sequence of orthogonal factors that represent the directions of largest variance. PCA is used in many applications, including machine learning [32], image processing [28], neurocomputing, engineering, and computer networks, especially for large datasets. However, despite its power and

W. Liu · H. Zhang · Y. Wang
China University of Petroleum (East China), Qingdao, Shandong, China

W. Liu
e-mail: liuwf@upc.edu.cn

D. Tao (✉)
South China University of Technology, Guangzhou, Guangdong, China
e-mail: dtao.scut@gmail.com

K. Lu
University of the Chinese Academy of Sciences, Beijing, China

popularity, a major limitation of PCA is that the derived principal components (PCs) are difficult to interpret and explain because they tend to be linear combinations of all the original variables.

Over the past 10 years, sparse principal component analysis (SPCA) has been used to improve the interpretability of PCs. SPCA aims to find a reasonable balance between statistical fidelity and interpretability by approximating sparse PCs. Briefly, SPCA methods can be divided into two groups: (1) ad hoc methods [16] [4] and (2) sparsity penalization methods [17] [36] [1] [2] [23] [18]. Ad hoc methods post-process the components obtained from classical PCA; for example, Jolliffe [16] uses rotation techniques in the standard PCA subspace to find sparse loading vectors, while Cadima and Jolliffe [4] simply set the PCA loadings with small absolute values to zero. Sparsity penalization methods usually formulate the SPCA problem as an optimization program by adding a sparsity-penalized term into the PCA framework. For example, Jolliffe et al.[17] maximize the Rayleigh quotient of the data covariance matrix under the L1-norm penalty in the SCoTLASS algorithm. Zou et al.[36] formulate sparse PCA as a regression-type optimization problem by imposing the LASSO penalty on the regression coefficients. In the DSPCA algorithm, d'Aspremont et al. [1] solve a convex relaxation of the sparse PCA, while Moghaddam et al.[23] and d'Aspremont et al.[2] go on to use greedy methods in order to solve the combinatorial problems encountered in sparse PCA. Finally, Journée et al.[18] propose the generalized power method for sparse PCA (GP-SPCA), in which sparse PCA is formulated as two single-unit and two block optimization problems. GP-SPCA has optimal convergence properties when either the objective function, or the feasible set, are strongly convex [18].

There is ever growing collection, sharing, combination, and use of massive amounts of data. The analysis of such "big data" has become essential in many commercial and scientific applications, from image analysis [20] [21] to genome sequencing. Parallel computing algorithms are essential for large-scale, high-dimensional data. Fortunately, modern graphics processing units (GPUs) have a highly parallel structure that makes them ideally suited to processing big data algorithms as well as graphics [25].

In this study we consider how to build compact, unsupervised representations of large-scale, high-dimensional data using sparse PCA schemes, with an emphasis on executing the algorithm in the GPU environment. The work can be regarded as a set of parallel optimization procedures for SPCA; specifically, we construct parallel implementations of the four optimization formulations used in GP-SPCA. To the best of our knowledge, GP-SPCA has not previously been implemented using GPUs. We compare the GPU implementation (on an NVIDIA Tesla C2050) with the corresponding CPU implementation (on a six-core 3.33 GHz high-performance cluster) and show that the parallel GPU implementation of GP-SPCA is up to 11 times faster than the corresponding CPU implementation, and up to 107 times faster than the corresponding MatLab implementation. We also conduct extensive comparative experiments of SPCA and PCA on several benchmark datasets, which provide further evidence that SPCA outperforms PCA in the majority of cases.

The remainder of this paper is organized as follows. GP-SPCA is briefly introduced in Section 2. The implementation of GP-SPCA on GPUs using CUBLAS is described in Section 3, and the experiments are presented in Section 4. We conclude in Section 5.

2 Generalized power method of SPCA

Let $A \in R^{p \times n}$ be a matrix encoding p samples of n variables. SPCA aims to find principal components that are both sparse and explain as much of the variance in the data as possible, and in doing so finds a reasonable trade-off between statistical fidelity and interpretability. GP-SPCA considers two single-unit and two block formulations of SPCA, in order to extract m sparse principal components, with $m=1$ for two single-unit formulations of SPCA and $p \geq m \geq 1$

for the two block formulations of SPCA. GP-SPCA maximizes a convex function on the unit Euclidean sphere in R^p (for $m=1$) or on the Stiefel manifold in $R^{p\times m}$ (for $m>1$). Depending on the type of penalty (either l_1 or l_0) used to enforce sparsity, there are four formulations of SPCA, namely single-unit SPCA via the l_1-penalty (GP-SPCA-SL1), single-unit SPCA via the l_0-penalty (GP-SPCA-SL0), block SPCA via the l_1-penalty (GP-SPCA-BL1), and block SPCA via the l_0-penalty (GP-SPCA-BL0).

Denote the unit Euclidean ball (resp. sphere) in R^k by $B^k=\{y\in R^k | y^Ty\le 1\}$ (resp. $S^k=\{y\in R^k | y^Ty=1\}$). Denote the space of $n\times m$ matrices with unit-norm columns by $[S^n]^m=\{Y\in R^{n\times m} | Diag(Y^TY)=I_m\}$, where $Diag(\ )$ is the diagonal matrix, by extracting the diagonal of the argument. Denote the Stiefel manifold by $S_m^p=\{Y\in R^{n\times m} | Y^TY=I_m\}$, and write $sign(t)$ for the sign of the argument $t\in R$ and $t_+=max\{0,t\}$. The characteristics of the four variants of GP-SPCA are summarized in Table 1 [18]. And we implement all the four formulations of GP-SPCA on the GPU to boost the efficiency.

GP-SPCA has optimal convergence properties when either the objective functions, or the feasible set, are strongly convex, which is the case with the single-unit formulations and can be enforced in the block cases [18].

3 GPU implementation of SPCA (GP-SPCA)

GPUs are typically used for computer graphics processing in general-purpose computing. There is a discrepancy between the floating-point capability of the CPU and GPU because the GPU is specialized for intensive, highly-parallel computation, and is therefore specifically designed to devote more transistors to data processing rather than data caching and flow control, as shown in Fig. 1 [25].

CUDA™ is a general-purpose parallel computing architecture designed by NVIDIA, which has a parallel programming model and instruction set architecture. CUDA guides the programmer to partition a problem into a sub-problem that can be solved as independent parallel blocks of threads in a thread hierarchy; Fig. 2 illustrates the hierarchy of threads, blocks, and grids used in CUDA. As well as the CUDA programming environment, NVIDIA also supplies toolkits for the programmer: CUBLAS [26] is one such library that implements Basic Linear Algebra Subprograms (BLAS).

Here we implement all formulations of GP-SPCA on the GPU using CUBLAS. The data space is allocated both on the host memory (CPU) and on the device memory (GPU). Data are initialized on the host memory before being transferred to the device memory, after which parallel computation is performed on the device memory. The results are then transferred back to the host memory when computation is complete.

4 Experiments

In this section, we conduct comparative experiments to evaluate the efficiency of GPU computing and the effectiveness of GP-SPCA.

5 Efficiency of GPU computing

In order to compare the efficiency of GPU and CPU computing, we first conduct the CPU implementation of GP-SPCA using GSL CBLAS [9], which is a highly efficient implementation of BLAS. We also compare the implementation with the MatLab application presented in [18] .

A six-core 3.33 GHz high performance cluster was used for the CPU implementation, and an NVIDIA Tesla C2050 for the GPU implementation. Twenty test instances were generated

Table 1 The four variant formulations of GP-SPCA

	Original form of SPCA	Reformulation
GP-SPCA-SL1	$\phi_{l_1}(\gamma)\equiv\max_{z\in B^n}\sqrt{z^T\Sigma z}-\gamma\|z\|_1$	$\phi^2_{l_1}(\gamma)\equiv\max_{x\in S^p}\sum_{i=1}^{n}\left[\left\|a_i^Tx\right\|-\gamma\right]_+^2$
GP-SPCA-SL0	$\phi_{l_0}(\gamma)\equiv\max_{z\in B^n} z^T\Sigma z-\gamma\|z\|_0$	$\phi_{l_0}(\gamma)\equiv\max_{x\in S^p}\sum_{i=1}^{n}\left[\left(a_i^Tx\right)^2-\gamma\right]_+$
GP-SPCA-BL1	$\phi_{l_{1,m}}(\gamma)\equiv\max_{\substack{X\in S_m^p\\ Z\in[S^n]^m}} Tr\left(X^TAZN\right)-\sum_{j=1}^{m}\gamma_j\sum_{i=1}^{n}\left\|z_{ij}\right\|$	$\phi^2_{l_{1,m}}(\gamma)\equiv\max_{X\in S_m^p}\sum_{j=1}^{m}\sum_{i=1}^{n}\left[\mu_j\left\|a_i^Tx_j\right\|-\gamma_j\right]_+$
GP-SPCA-BL0	$\phi_{l_{0,m}}(\gamma)\equiv\max_{\substack{X\in S_m^p\\ Z\in[S^n]^m}} Tr\left(Diag\left(X^TAZN\right)^2\right)-\sum_{j=1}^{m}\gamma_j\left\|z_j\right\|_0$	$\phi_{l_{0,m}}(\gamma)\equiv\max_{X\in S_m^p}\sum_{j=1}^{m}\sum_{i=1}^{n}\left[\left(\mu_ja_i^Tx\right)^2-\gamma_j\right]_+$

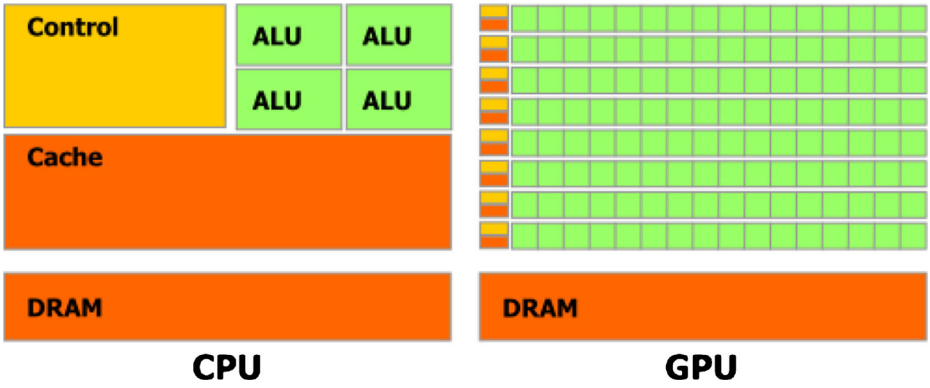

Fig. 1 The difference between GPU and CPU [25] . The GPU is especially well-suited for data-parallel computation, and the same program is executed on many elements in parallel

for each input matrix $A_{P\times N}$ ($N\in[5.0\times10^2,3.2\times10^4]$, $P=N/10$). Here, $m=5$ is the number of sparse PCs, and $\gamma\in\{0.01,0.05\}$ is the aforementioned parameter that balances the sparsity and variance of the PCs.

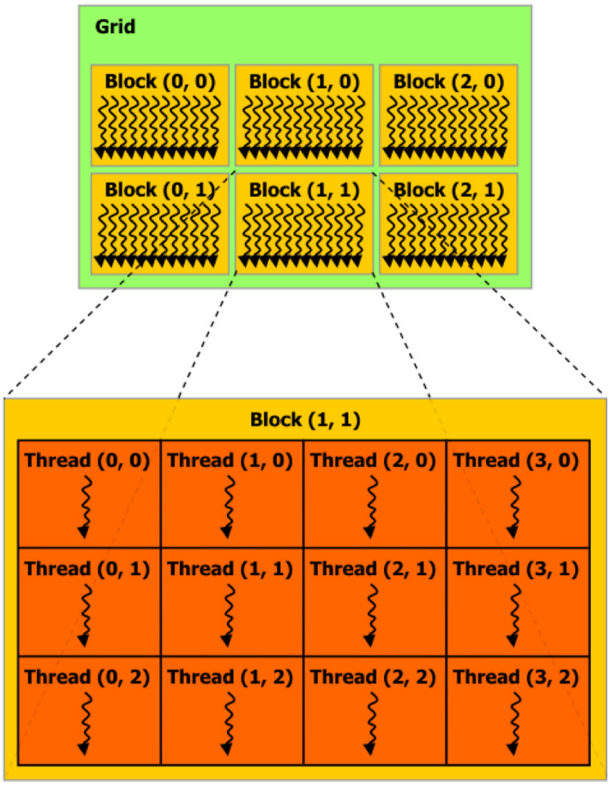

Fig. 2 Grids of thread blocks [25]. A program is divided into several grids each of which is partitioned into blocks of threads that execute independently from each other

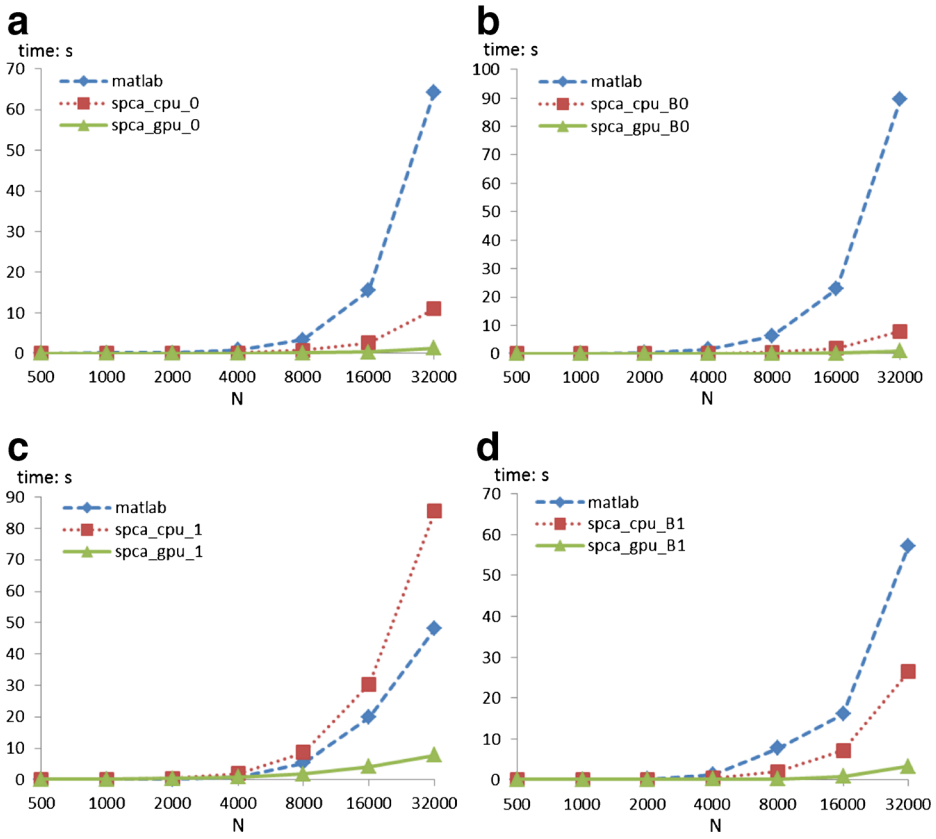

Fig. 3 A comparison of GP-SPCA performed on a GPU (Tesla C2050) and a CPU. The x-axis indicates the size of the input matrix and the y-axis denotes computation time. **a**. GP-SPCA-SL0, m=5, γ=0.01. **b**. GP-SPCA-BL0, m=5, γ=0.01. **c**. GP-SPCA-SL1, m=5, γ=0.05 **d**. GP-SPCA-BL1, m=5, γ=0.05

Figure 3 shows the average running time of different input matrices using different parameters. The difference in processing time (between CPU and GPU) increases with increasing size of the input matrix, with up to eleven times improvement in speed over the corresponding CBLAS implementation, and up to 107-times over the MatLab implementation.

Fig. 4 Examples of handwriting in the USPS database

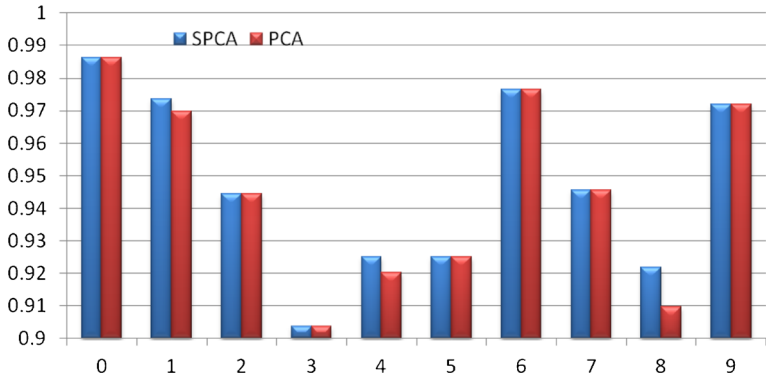

Fig. 5 Recognition of SPCA and PCA on USPS

6 Effectiveness of GP-SPCA

To evaluate the effectiveness of GP-SPCA in practice, we next conducted GP-SPCA and PCA experiments on several benchmark datasets, including the USPS database [14], the COIL20 database [24], and the Isolet spoken letter recognition database [3] . For each experiment, we used GP-SPCA and PCA to learn the project functions using training samples, before mapping all the samples (both training and test samples) into the lower dimensional subspace where recognition is performed using a nearest neighbor classifier.

7 USPS database

The USPS database [14] is a handwritten digit database containing 9,298 16 × 16 pixel handwritten digit images in total (Fig. 4). Each pixel is with 256 grey levels, thus each image is represented by a 256-dimensional vector. The database was split into 7,291 training images and 2007 test images as in [5] [6], with the parameter γ set to 0.1.

The results of SPCA and PCA in recognizing the ten handwritten digits are shown in Fig. 5, from which we can see that SPCA outperforms PCA in most cases.

8 COIL20 database

The COIL20 database [24] contains 1,440 images of 20 objects (for examples, see Fig. 6). The images of each object are taken five degrees apart as the object is rotated on a turntable, and as

Fig. 6 COIL20 examples

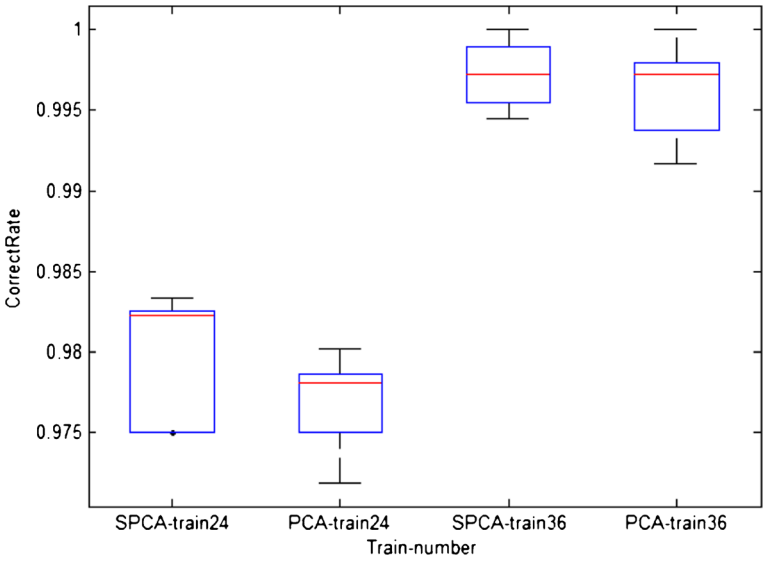

Fig. 7 The average recognition rates of SPCA and PCA on COIL20 data

a result each object is represented by 72 32×32 pixel images, with 256 grey levels per pixel. Thus, each image is represented by a 1,024-dimensional vector. We randomly selected two groups of 24 and 36 examples of each object as training sets, and used the remaining images for the test sets. The parameter γ was set to 0.3 for 24-example group, and 0.1 for the 36-example group. All the experiments were repeated five times.

Figure 7. shows that SPCA outperforms PCA in both cases. Figure 8, which shows the recognition rate of selected objects, demonstrates that SPCA outperforms PCA in most cases.

9 Isolet spoken letter recognition database

The Isolet spoken letter recognition database [3] contains 150 subjects, each of whom speaks each letter of the alphabet twice. The features include spectral coefficients; contour features, sonorant features, pre-sonorant features, and post-sonorant features as in [22] . The speakers were grouped into five sets of 30 speakers; three were used for training and two for testing in the first experiment and four groups for training the other for testing in the second experiment (to evaluate robustness). The parameter γ was set to 10^{-6} for the first experiment and 0.02 for

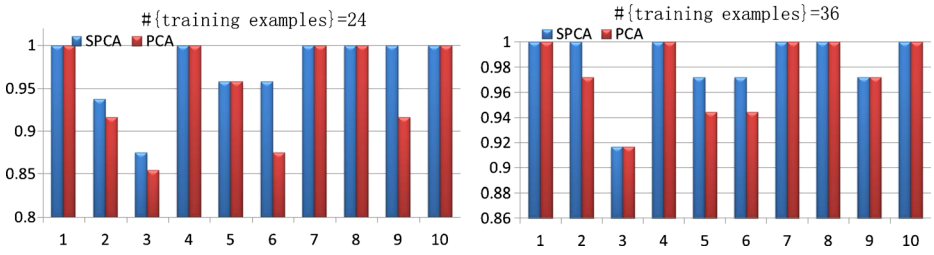

Fig. 8 The recognition results of selected objects

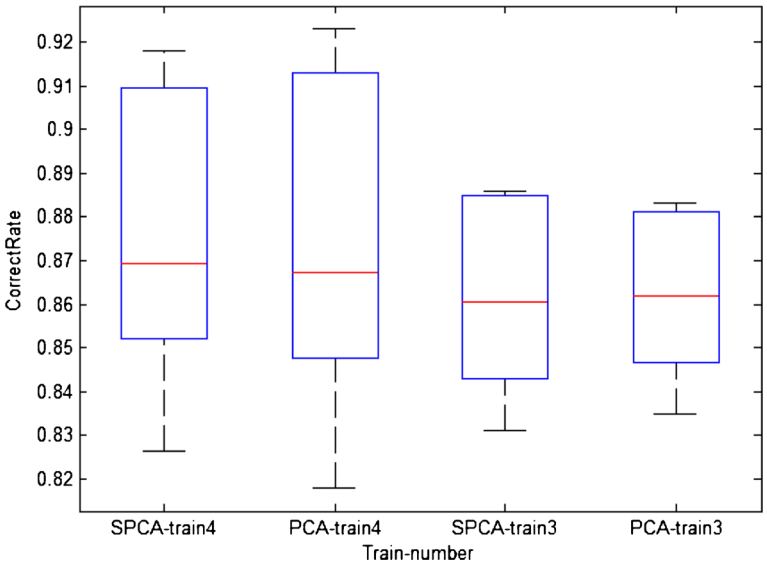

Fig. 9 The average recognition rates of SPCA and PCA on Isolet data

the second, and each experiment was repeated five times. Figs. 9 and 10 show the average recognition rates and recognition of each character, respectively. SPCA is superior to PCA in the majority of cases.

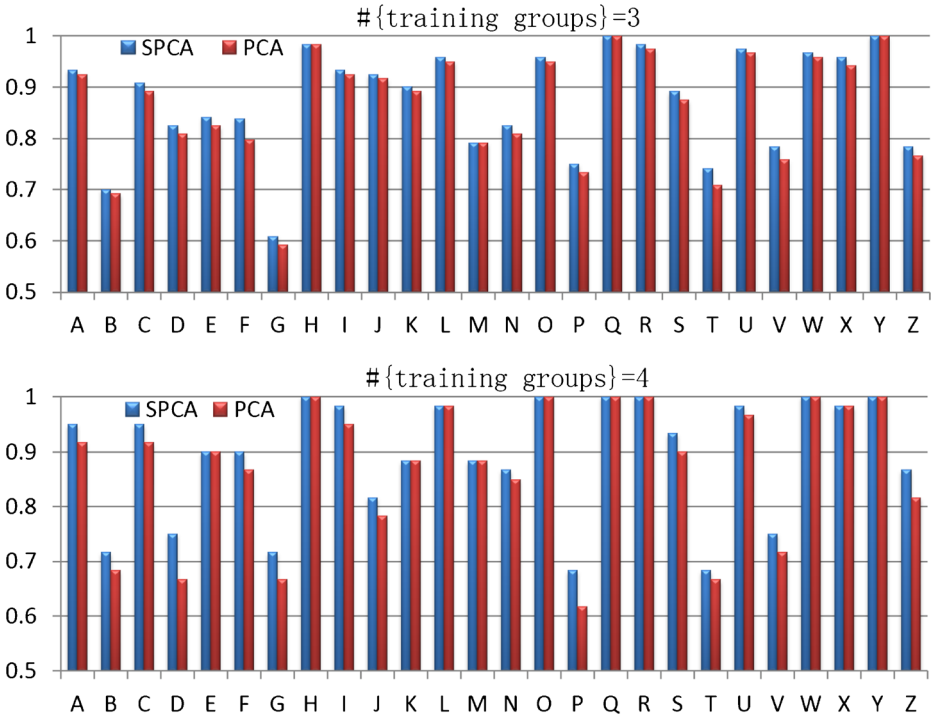

Fig. 10 Recognition rates for each character

10 Conclusion

Sparse PCA is a reasonable method for balancing statistical fidelity and interpretability. In this paper, we present a paralleled method of GP-SPCA, one of the most efficient SPCA approaches, using a GPU. Specifically, we construct parallel implementations of the four optimization formulations for the GPU, and compare this with a CPU implementation using CBLAS. Using real-world data, we experimentally validate the effectiveness of GP-SPCA and demonstrate that the parallel GPU implementation of GP-SPCA can significantly improve performance. This work has several potential applications in large-scale, high-dimension reduction problems such as video indexing [33] [7] [8]and web image annotation [34] [35] [19] [29] [30], which will be the subject of future study.

Acknowledgments This work was supported in part by the following projects: the National Natural Science Foundation of China (61271407, 61301242), Shandong Provincial Natural Science Foundation, China (ZR2011FQ016), the Fundamental Research Funds for the Central Universities, China University of Petroleum (East China) (13CX02096A, CX2013057, 27R1105019A).

References

1. d'Aspremont A, El Ghaoui L, Jordan MI, Lanckriet GRG (2007) A direct formulation for sparse PCA using semidefinite programming. SIAM Rev 49:434–448
2. D'Aspremont A, Bach FR, El Ghaoui L (2008) Optimal solutions for sparse principal component analysis. J Mach Learn Res 9:1269–1294
3. K. Bache and M. Lichman (2013) UCI machine learning repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science
4. Cadima J, Jolliffe IT (1995) Loadings and correlations in the interpretation of principal components. J Appl Stat 22:203–214
5. Cai D, He X, Han J, Huang T (2011) Graph regularized Non-negative matrix factorization for data representation. IEEE Trans PAM 33(8):1548–1560
6. Cai D, He X, Han J (2011) Speed Up kernel discriminant analysis. VLDB J 20(1):21–33
7. Cheng-Chieh C, Huei-Fang Y (2013) Quick browsing and retrieval for surveillance videos. Multimedia Tools Appl. doi:10.1007/s11042-013-1750-z
8. Youtian D, Feng C, Wenli X, Xueming Q (2013) Video content categorization using the double decomposition. Multimedia Tools Appl. doi:10.1007/s11042-012-1213-y
9. Mark Galassi, Jim Davies, James Theiler, Brian Gough, et al. (2003)GNU Scientific Library
10. Guan N, Tao D, Luo Z, Yuan B (2012) Online nonnegative matrix factorization with robust stochastic approximation. IEEE Trans Neural Netw Learning Syst 23(7):1087–1099
11. Guan N, Tao D, Luo Z, Yuan B (2012) NeNMF: an optimal gradient method for nonnegative matrix factorization. IEEE Trans Signal Process 60(6):2882–2898
12. Guan N, Tao D, Luo Z, Yuan B (2011) Manifold regularized discriminative nonnegative matrix factorization with fast gradient descent. IEEE Trans Image Process 20(7):2030–2048
13. Guan N, Tao D, Luo Z, Yuan B (2011) Non-negative patch alignment framework. IEEE Trans Neural Netw 22(8):1218–1230
14. Hull JJ (1994) A database for handwritten text recognition research. IEEE Trans Pattern Anal Mach Intell 16(5):550–554
15. Jolliffe IT (1986) Principal component analysis. Springer Verlag, New York
16. Jolliffe IT (1995) Rotation of principal components: choice of normalization constraints. J Appl Stat 22:29–35
17. Jolliffe IT, Trendafilov NT, Uddin M (2003) A modified principal component technique based on the LASSO. J Comput Graph Stat 12(3):531–547
18. Journée M, Nesterov Y, Richtárik P, Sepulchre R (2010) Generalize power method for sparse principal component analysis. J Mach Learn Res 11:517–553

19. Li J, Allinson NM, Tao D, Li X (2006) Multitraining support vector machine for image retrieval. IEEE Trans Image Process 15(11):3597–3601
20. Liu W, Tao D (2013) Multiview hessian regularization for image annotation. IEEE Trans Image Process 22: 2676–2687
21. Liu W, Tao D, Cheng J, Tang Y (2014) Multiview hessian discriminative sparse coding for image annotation. Comput Vis Image Underst 118:50–60
22. Fanty, Mark, and Ronald Cole. (1990) "Spoken Letter Recogniitiion
23. Moghaddam B, Weiss Y, Avidan S (2006) Spectral bounds for sparse PCA: exact and greedy algorithms. Advances in neural information processing systems, vol 18. MIT Press, Cambridge, pp 915–922
24. S. A. Nene, S. K. Nayar and H. Murase (1996) Columbia Object Image Library (COIL-20). Technical Report CUCS-005-96
25. NVIDIA, CUDA C Programming Guide (version 4.0), (2011)
26. NVIDIA, CUBLAS Library (2011)
27. J. Sun, D. Tao, C. Faloutsos (2006) Beyond streams and graphs: dynamic tensor analysis. KDD: 374–383
28. Tao D, Li X, Wu X, Maybank SJ (2007) General tensor discriminant analysis and Gabor features for gait recognition. IEEE Trans Pattern Anal Mach Intell 29(10):1700–1715
29. Tao D, Tang X, Li X, Wu X (2006) Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. IEEE Trans Pattern Anal Mach Intell 28(7):1088–1099
30. Tao D, Tang X, Li X, Rui Y (2006) Direct kernel biased discriminant analysis: a new content-based image retrieval relevance feedback algorithm. IEEE Trans on Multimed 8(4):716–727
31. Tao D, Li X, Wu X, Maybank SJ (2009) Geometric mean for subspace selection. IEEE Trans Pattern Anal Mach Intell 31(2):260–274
32. Xu C, Tao D, Xu C (2014) Large-margin multi-view information bottleneck. IEEE Trans Pattern Anal Mach Intell. doi:10.1109/TPAMI.2013.2296528
33. Zha Z-J, Wang M, Zheng Y-T, Yang Y, Hong R (2012) Tat-seng Chua: interactive video indexing with statistical active learning. IEEE Trans Multimedia 14(1):17–27
34. Zheng-Jun Zha, Xian-Sheng Hua, Tao Mei, Jingdong Wang, Guo-Jun Qi, Zengfu Wang (2008) Joint multi-label multi-instance learning for image classification. CVPR
35. Yan-Tao Z, Zheng-Jun Z, Tat-Seng C (2011) Research and applications on georeferenced multimedia: a survey. Multimed Tools Appl 51(1):77–98
36. Zou H, Hastie T, Tibshirani R (2006) Sparse principal component analysis. J Comput Graph Stat 15(2):265–286

Weifeng Liu(M'12) received the double B.S. degree in automation and business administration and the Ph.D. degree in pattern recognition and intelligent systems from the University of Science and Technology of China, Hefei, China, in 2002 and 2007, respectively. He is currently an Associate Professor with the College of Information and Control Engineering, China University of Petroleum, (East China), China. He was a Visiting Scholar with the Centre for Quan-tum Computation & Intelligent Systems, Faculty of Engineering & Information Technology, University of Technology Sydney, Sydney, Australia, from 2011 to 2012. His current research interests include computer vision, pattern recognition, and machine learning.

Huimin Zhang received the B.S degree in electronic and information engineering from China University of Petroleum (East China) in 2013. She is currently pursuing her master degree in electronic and information engineering. Her research focuses on the applications of sparse learning in computer vision.

Dapeng Tao received a BEng degree from Northwestern Polytechnical University and a PhD degree from South China University of Technology, respectively. Over the past years, his research interests include machine learning, computer vision and cloud computing.

Yanjiang Wang received his Ph.D degree from Beijing Jiaotong University in 2001. He is currently a professor with the College of Information and Control Engineering, China University of Petroleum (East China), China. His research focuses on intelligent information processing, computer vision and pattern recognition.

Ke Lu was born in Ningxia on March 13, 1971, He received Master degree and PH.D degree from Department of Mathematics and Department of Computer science at Northwest University in July 1998 and July 2003, respectively. He as Postdoctoral Fellow in Institute of Automation Chinese Academy of Sciences from July 2003 to April 2005. Currently he is a professor of University of the Chinese Academy of Sciences. He research focuses on curve matching, 3D image reconstruction and computer graphics.