Circuits Syst Signal Process (2014) 33:309–320
DOI 10.1007/s00034-013-9630-3

SHORT PAPER

Single-Pass K-SVD for Efficient Dictionary Learning

Kuang-Yu Chang · Cheng-Fu Lin ·
Chu-Song Chen · Yi-Ping Hung

Received: 22 January 2013 / Revised: 19 June 2013 / Published online: 17 July 2013

Abstract Sparse representation has been widely used in machine learning, signal processing and communications. K-SVD, which generalizes k-means clustering, is one of the most famous algorithms for sparse representation and dictionary learning. K-SVD is an iterative method that alternates between encoding the data sparsely by using the current dictionary and updating the dictionary based on the sparsely represented data. In this paper, we introduce a single-pass K-SVD method. In this method, the previous input data are first summarized as a condensed representation of weighted samples. Then, we developed a weighted K-SVD algorithm to learn a dictionary from the union of this representation and the newly input data. Experimental results show that our approach can approximate K-SVD's performance well by consuming considerably less storage resource.

Keywords K-SVD · Sparse representation · Dictionary learning · Single-pass learning

K.-Y. Chang · C.-S. Chen (✉) · Y.-P. Hung
Institute of Information Science, Academia Sinica, Taipei, Taiwan
e-mail: song@iis.sinica.edu.tw

K.-Y. Chang
e-mail: kuangyu@iis.sinica.edu.tw

K.-Y. Chang · Y.-P. Hung
Department of Computer Science & Information Engineering, National Taiwan University, Taipei, Taiwan

C.-F. Lin · C.-S. Chen
Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan

C.-F. Lin
e-mail: chengfu@umich.edu

Y.-P. Hung
e-mail: hung@csie.ntu.edu.tw

1 Introduction

Codeword learning is fundamental in signal processing. A real-world signal can usually be represented as a linear combination of a few codewords in an overcomplete dictionary. This representation is beneficial to applications in audio/image processing, pattern recognition and image restoration. To construct the codewords, clustering or vector quantization has been used in the past decades, which separates the training data into disjoint groups. Among many clustering methods, k-means [5, 17] is known as the most widely adopted. Given the cluster number k, the goal of k-means is to choose k centers that minimizes the total squared distances between the data points to their closest centers. Recently, k-means has been extended to K-SVD [4], where each of the training data is not associated to only a single codeword but a linear combination of few codewords. K-SVD alternates between encoding the data sparsely with the presently calculated dictionary and then updating the dictionary atoms from the sparse representation of the data. This method is effective in finding the codewords and the associated sparse representation of a dataset, which has received much attention for applications such as de-noising [20, 21], source separation [2], restoration [20, 25], channel estimation [13], face and object-category recognition [16, 24], and so on. Details of K-SVD will be introduced in Sect. 2.

Although K-SVD has been extensively used for dictionary learning and sparse representation, it is hard to scale when the dataset is large. In this situation, we could only allow part of the data to be loaded, and thus are able to process only partial data at a time. Moreover, some data collected from the Internet or mobile devices are generated from time to time, which are not allowed to be kept for future use; hence, when dealing with them, we should drop out the old data when the new data are coming, and the old data cannot be retrieved again. In this paper, we propose a single-pass algorithm for K-SVD. In our algorithm, the data have to be scanned only once, and the data already processed can be abandoned. Moreover, for hardware implementations, data retrieval time is critical. Scanning the dataset multiple times increases the computational complexity and could increase the system overhead extensively.

As a special case of sparse representation, clustering also suffers from the scalability problem. Bradley et al. [7] and Farnstorm et al. [12] solved k-means clustering by scanning the database in a single pass. Similar idea has been adopted for soft clustering, where each datum could belong to multiple clusters with different membership degrees. In [14], previous data are categorized into several soft clusters, and each cluster is summarized as its center point with the weight equal to sum of the membership degrees of the data in this cluster. Then, the weighted fuzzy c-means algorithm is adopted to find clusters for the combination of the weighted samples and the current data.

Some other works [18, 19] developed on-line algorithms to speed up dictionary learning. Previous data are aggregated for codewords construction and updating. However, for a given dataset, multiple passes of scans are required to obtain a satisfied convergent result (as suggested by their works). In this paper, we focus on scanning the dataset in a single pass, where each datum can be retrieved only once. In the interest of enhancing K-SVD's capability in dealing with on-line and large data, we propose an improved K-SVD algorithm that learns the dictionary by a single-pass scan of the whole dataset. Our method cannot only perform incremental learning based on an infinite data stream, but is also applicable to learning from a fixed-size dataset by scanning it only once. The principle introduced in this paper can be employed for extending other dictionary-learning methods to single-pass methods as well.

This paper is organized as follows. Section 2 reviews K-SVD. Section 3 introduces our algorithm. Experimental results are shown in Sect. 4. Finally, conclusions are given in Sect. 5.

2 Review of K-SVD

Recently, K-SVD [4] has become popular for sparse representation and codeword learning. Given an $l \times N$ data matrix as input,

$$X = [x_1, x_2, \dots, x_N],$$

which consists of N vectors, $x_i \in R^l$. K-SVD decomposes $X \cong WC$ with $W : l \times m$ and $C : m \times N$ by solving

$$\min_{W,C} \|X - WC\|_F^2, \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm of matrix. W is called the codeword matrix whose columns are the codewords and m is the number of codewords. C is the coefficient matrix. Let the jth column be c_j. The jth datum x_j is approximated by a linear combination of the codewords, $x_j \cong Wc_j$.

The decomposition is imposed with the constraint that the sparse degree of the m-dimensional vector c_j, i.e., the number of c_j's non-zero elements or its zero-norm, $\|c_j\|_0$, equals s for all $j = 1, \dots, N$,

$$\|c_j\|_0 = s, \quad j = 1, \dots, N. \tag{2}$$

Since we hope to represent each datum by using only very few codewords (so as to achieve the 'sparseness'), we usually set $s \ll m$.

K-SVD is effective in finding codewords and the associated sparse representation for a dataset. The codewords then serve as basis vectors for the data. Each datum is associated with only few codewords since most coefficients in the linear combination are zero.

The K-SVD algorithm finds W and C via iterations of the following two alternative steps:

(1) Given W, find the sparse coefficient matrix C by any pursuit algorithms. As the exact solution is NP-hard, these pursuit algorithms find approximate solutions instead.
(2) Given C, refine both W and C by multiple rank-one updates performed by singular value decompositions (SVDs).

Most K-SVD algorithms adopted orthogonal matching pursuit (OMP) [22] in Step 1, which reaches the zero-norm constraint by sequential forward search. In the K-SVD algorithm [4], the codeword matrix W is initialized by randomly selecting some training data, and we follow this approach for initialization too.

There are a variety of codeword learning studies for sparse representation, including the approaches employing p-norm penalty of distinct p and/or their variations [3, 9–11, 15, 23]. A thorough review is beyond the scope of this paper. Readers can refer to [8] for general information. This paper focuses on K-SVD and its extension to single-pass incremental learning.

3 Method

Since we are only allowed to scan the data once, we have to summarize previous data as a concise representation and seek to refine it via the updated data. In our approach, the old data having been processed were summarized into a fixed number of samples, where each sample is associated with a weight reflecting its importance.

At each run, there are two kinds of data for the proposed single-pass algorithm. One is the old data that are summarized as some representative samples and the corresponding importance weights, denoted as $X_{\text{Pre}} = \{(x_j, \sigma_j) \in R^l \times R^+ \mid j = 1, \ldots, n\}$, where σ_j is the weight of the importance sample x_j. The other is the new data, $X_{\text{New}} = \{(x_j, \sigma_j) \mid j = (n+1), \ldots, 2n\}$, $\sigma_j = 1$. The union of these two data,

$$X = X_{\text{Pre}} \cup X_{\text{New}} = \big\{(x, \sigma_j) \mid j = 1, \ldots, N\big\} \tag{3}$$

constitutes the training data at the current time, where $N = 2n$.

The new data are real samples that should be more reliable but their amount is limited. On the other hand, despite the number of old data is unlimited, their importance samples are only approximations. Hence, both kinds of data have their pros/cons. To balance their influence, we keep their total weights the same at every time:

$$\sum_{j=1}^{n} \sigma_j = \sum_{j=n+1}^{2n} \sigma_j = n. \tag{4}$$

The rest of this section is organized as follows. Since the dataset X to be processed at each run consists of weighted samples, we first introduce weighted-samples K-SVD in Sect. 3.1, which is an extension of K-SVD when the training data are weighted. Then, we introduce the summarization of X by importance samples X_{Pre} in Sect. 3.2.

3.1 Weighted-Samples K-SVD

We follow the notation convention: if a matrix is denoted by an upper-case character, say A, then its low-case notation a_j denotes its jth column, and $a^{(i)}$ denotes its ith row, respectively. Let a_{ij} denote the ijth entry of A.

Consider the training data X defined in (3). Let B be the codeword matrix and D the coefficient matrix. We hope to represent each x_j as a sparse combination of the codewords,

$$x_j \cong \sum_{i=1}^{m} d_{ij} b_i, \tag{5}$$

such that

$$\|d_j\|_0 = s. \tag{6}$$

Since the data are associated with weights, we hope to minimize the sum of weighted errors for solving B and D:

$$\min_{B,D} \sum_{j=1}^{N} \sigma_j \left\| x_j - \sum_{i=1}^{m} d_{ij} b_i \right\|^2. \tag{7}$$

First, assume the codeword matrix B is fixed. Since each term in (7) is independent to the data index j, the terms

$$\left\| x_j - \sum_{i=1}^{m} d_{ij} b_i \right\|^2, \tag{8}$$

$j = 1, \dots, N$, can be minimized independently no matter what σ_j is. This is the same situation occurred in the original K-SVD when the codeword matrix is fixed. Hence, to find the coefficient matrix D (with B fixed), we also use OMP to find the coefficient vector d_j for all j in (8), so that the constraint $\|d_j\|_0 = s$ is satisfied.

Second, we seek to refine both the coefficients D and the codewords B. Let us re-formulate the objective in (7) as

$$\sum_{j=1}^{N} \left\| y_j - \sum_{i=1}^{m} \sqrt{\sigma_j} d_{ij} b_i \right\|^2, \tag{9}$$

where $y_j = \sqrt{\sigma_j} x_j$ and the data matrix is reweighted as $Y = [y_1, y_2, \dots, y_N]$. The equivalent matix-form of (9) is

$$\min_{B,D} \|Y - BD\Lambda\|_F^2, \tag{10}$$

where Λ is an $N \times N$ diagonal matrix with the jth entry being $\sqrt{\sigma_j}$. By denoting $E = D\Lambda$, (10) is expressed as

$$\min_{B,E} \|Y - BE\|_F^2. \tag{11}$$

Note that (11) becomes exactly the same form of the original unweighted K-SVD. Hence, we can also use iterated rank-one updates to solve B and E. Like K-SVD, we only modify the non-zero entries in E; that is, the zero entries in E obtained in the first step remain zeros. Also, only B's entries that are associated with the non-zero entries in E are processed when modifying B. When E is solved, D is uniquely determined since the weights σ_j are positive for all j and so the non-zero entries of E and D are the same. More specifically, let b_i and $e^{(i)}$ be the ith column and row of B and E, respectively, $i = 1, \dots, m$. Then (11) becomes

$$\min_{B,E} \left\| Y - \sum_{i=1}^{m} b_i e^{(i)} \right\|_F^2 . \tag{12}$$

Denote $\bar{e}^{(i)}$ to be the condensed vector of $e^{(i)}$, consisting of only the non-zero entries of $e^{(i)}$; let $\bar{b}_i$ be the condensed vector of b_i with respect to $\bar{e}^{(i)}$. When updating $\bar{b}_k$ and $\bar{e}^{(k)}$ for a particular k, we assume that the other b_i and $e^{(i)}$ are fixed for $i \neq k$, yielding

$$\min_{B,E} \left\| Z - \bar{b}_k \bar{e}^{(k)} \right\|_F^2, \tag{13}$$

where $Z = \bar{Y} - \sum_{i \neq k} \bar{b}_i \bar{e}^{(i)}$ and $\bar{Y}$ is the submatrix of Y associated with the non-zero entries of $e^{(k)}$. This rank-one fitting problem in (13) is solved by the SVD of Z with $(\bar{b}_k, \bar{e}^{(k)}) = (u_1, \gamma_1 v_1^T)$, where γ_1 is Z's largest singular value and u_1, v_1 are its left and right singular vectors, respectively. Then, via back substitution, we obtain $(\bar{b}_k, \bar{d}^{(k)}) = (u_1, \gamma_1 \bar{\Lambda}^{-1} v_1^T)$, where $\bar{d}^{(k)}$ is the condensed vector associated with $d^{(k)}$ and $\bar{\Lambda}$ is defined accordingly. By modifying $(\bar{b}_k, \bar{d}^{(k)})$ for every k in a random order, B and D are refined in the second step.

Both of the above two steps reduce the objective function in (7) and are iterated alternatively. The procedure stops when the error converges (the error of the current iteration is larger than that of the previous iteration) or when it achieves a pre-set maximal number of allowed iterations.

3.2 Incremental Importance Sampling

Let the weighted data at time t be $X^{[t]} = \{(x_j^{[t]}, \sigma_j^{[t]}) \mid j = 1, \dots, 2n\}$. Via the weighted-samples K-SVD in Sect. 3.1, we can obtain the codewords at time t. After that, we condense $X^{[t]}$ into n importance samples, so that the old data from time 1 to time t are summarized as $X_{\mathrm{Pre}}^{[t+1]} = \{(x_r^{[t+1]}, \sigma_r^{[t+1]}) \mid r = 1, \dots, n\}$ for time $t+1$. To find the importance samples $X_{\mathrm{Pre}}^{[t+1]}$ from $X^{[t]}$, we simply adopt the k-means clustering algorithm [5, 17].

More specifically, since each $x_j^{[t]}$ is weighted by $\sigma_j^{[t]}$, the k-means algorithm employed in our work is the one that can deal with weighted samples, which is simply a special case of the expectation-maximization (EM) algorithm for learning the Gaussian-mixture model when the variance of each Gaussian component approaches to zero [6].

After performing weighted k-means on $X^{[t]}$, the n cluster centers are obtained and serve as the important samples $x_r^{[t+1]}$, $r = 1, \dots, n$. The associated weights of important samples, $\sigma_r^{[t+1]}$, are then set by summing up the weights of the data contained in the rth cluster,

$$\sigma_r^{[t+1]} = \sum_{x_j^{[t]} \in \text{ the } r\text{th cluster}} \sigma_j^{[t]}/2, \tag{14}$$

$r = 1, \dots, n$. The division by 2 in (14) is to normalize the total weight of the summarization samples to n, so that the constraint in (4) is satisfied. It can be verified as follows:

$$\sum_{r=1}^{n} \sigma_r^{[t+1]} = \frac{1}{2} \sum_{\text{all clusters}} \left\{ \sum_{x_j^{[t]} \in \text{the } r\text{th cluster}} \sigma_j^{[t]} \right\} = \frac{1}{2} \sum_{j=1}^{2n} \sigma_j^{[t]} = n. \tag{15}$$

The $x_r^{[t+1]}$ and $\sigma_r^{[t+1]}$ obtained then constitute the importance samples $X_{\text{Pre}}^{[t+1]}$ for the old data at time $t + 1$.

The single-pass K-SVD algorithm is summarized in Algorithm 1

Algorithm 1 Single-pass K-SVD

```
Input: the data matrix $X_{\text{input}} = [x_1, x_2, \dots, x_N, \dots]$
Output:
1: Initially, $t \leftarrow 1$. $X_{\text{Pre}}^{[t]} = \{(x_j, 1) \mid j = 1, \dots, n\}$.
2: Input $X_{\text{New}}^{[t]} = \{(x_j, 1) \mid j = (tn + 1), \dots, (t + 1)n\}$.
3: Let $X^{[t]} = X_{\text{Pre}}^{[t]} \cup X_{\text{New}}^{[t]}$.
4: Run weighted K-SVD (Sect. 3.1) for $X^{[t]}$; obtain the codeword matrix $B$ and coefficient matrix $D$.
5: Run weighted k-means (Sect. 3.2) on $X^{[t]}$ and set the weights by (14); obtain $X_{\text{Pre}}^{[t+1]} = \{(x_r^{[t+1]}, \sigma_r^{[t+1]}) \mid r = 1, \dots, n\}$.
6: If no data are available, stop; else $t \leftarrow t + 1$ and goto Step 2.
7: return the codeword matrix $B$;
```

4 Experimental Results

4.1 Datasets

To validate our approach, we apply it to several datasets as shown in Table 1. For each dataset, we permute its data randomly and divide it into p disjoint subsets. Each subset contains $n = N_{\text{total}}/p$ samples. For example, for the MNIST dataset, the data amount of each subset is $n = 7000$ for $p = 10$ and $n = 700$ for $p = 100$. We denote

Table 1 Datasets in our experiments

Database	Dimension	Number of data
MIT-CBCL	361 (19 × 19)	2,429
COIL	1,024 (32 × 32)	7,200
Extended Yale B	32,256 (192 × 168)	2,414
MNIST	784 (28 × 28)	70,000

Table 2 Reconstruction errors for 10 % downsize ratio and sparse degree $s = 3$, where the format in each blank is mean ± standard deviation (*rel_err*)

One-patch	Our single pass	Batch K-SVD
MIT-CBCL	406.45 ± 2.91 (2.0 %)	398.58
COIL	1781.81 ± 11.02 (1.5 %)	1755.41
Extended Yale B	706.18 ± 2.92 (2.1 %)	691.49
MNIST	35469.48 ± 80.71 (0.4 %)	35341.00

$1/p$ as the *downsize ratio*, which are 10 % and 1 % , respectively. Note that each subset is used only once in our single-pass setting. They are not allowed to be retrieved repeatedly.

4.2 Results

First, we run the K-SVD algorithm for them, where the K-SVD Matlab code available at [1] is used to generate the results. We set the number of codewords $m = 50$ and the sparse degree $s = 3$. The reconstruction errors (1) obtained for MIT-CBCL, COIL, Extended Yale B and MNIST are 398.58, 1755.41, 691.49 and 35341.00, respectively. They are referred to as the *batch K-SVD* results since the entire data are used at the same time.

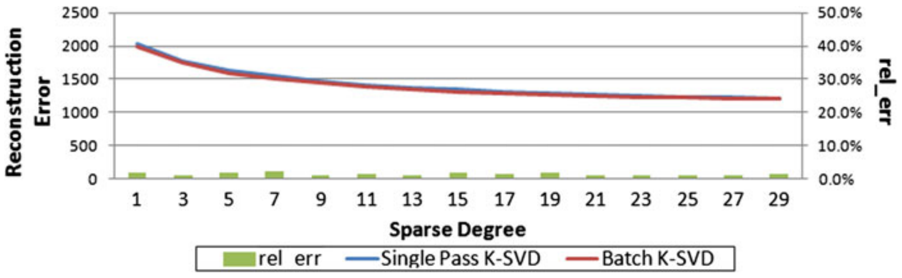
Our approach is a single-pass version of K-SVD. Therefore, what we are interested is how well it can approximate original K-SVD's performance. When the downsize ratio is 10 % and under the same settings that the sparse degree $s = 3$, the reconstruction errors obtained by our method are shown in Table 2. We define the relative error to reflect the increment when treating the error of the batch K-SVD as one:

$$rel\_err = 100 \times (e_{\text{obtained}} - e_{\text{batch}})/e_{\text{batch}} \ \%, \tag{16}$$

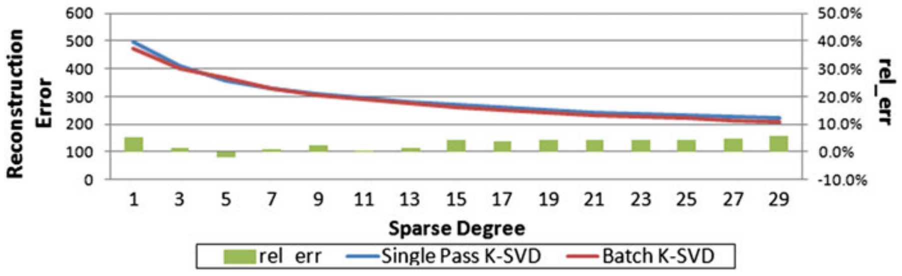
where e_{batch} and e_{obtained} are the errors obtained by using batch K-SVD and single-pass K-SVD, respectively. Compared to the reconstruction errors of batch K-SVD, the single-pass K-SVD can achieve the performance with the error-increasing (*rel_err*) less than 2.5 % for all datasets by using only 10 % data per each time.

In another experiment, we vary the sparse degree from $s = 1$ to 29 when the downsize ratio is kept as 10 %, and show the reconstruction error and *rel_err* for MIT-CBCL and COIL in Fig. 1. We can observe that the relative error remains low when the sparse degree is getting larger. Sometimes single-pass K-SVD is even more accurate than batch K-SVD.

Figure 2 shows the reconstruction error and *rel_err* when the downsize ratio is getting decreased. For the smaller dataset, Extended Yale B (size < 2500), we vary the downsize ratio from 10 % to 2.5 %. For the larger one, MNIST (size > 35000), we vary the downsize ratio from 10 % to 1 %. Basically, the results obtained follow the principle: the smaller is the downsize ratio, the larger is the error. However, the *rel_err* are still within a satisfied range when the downsize ratio is decreased for both datasets. In particular, for the larger dataset, the relative error obtained is much smaller by our single-pass approach.

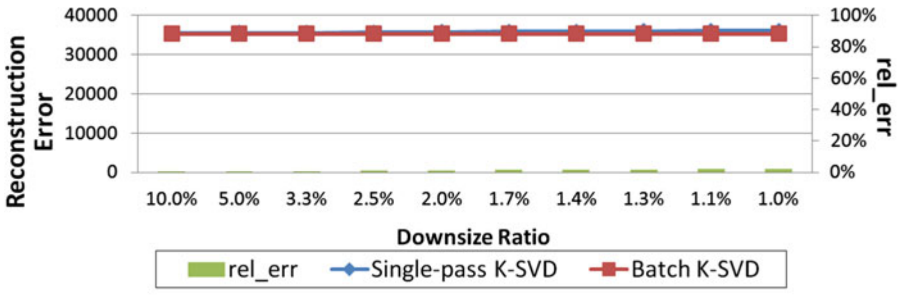


(a) COIL

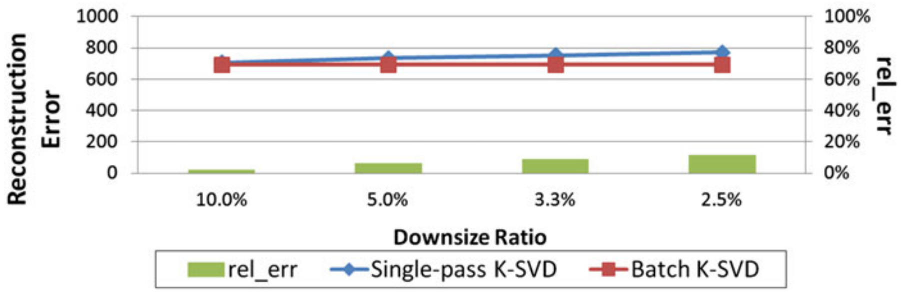


(b) MIT-CBCL

Fig. 1 Reconstruction errors of various sparse degree (downsize ratio = 10 %)

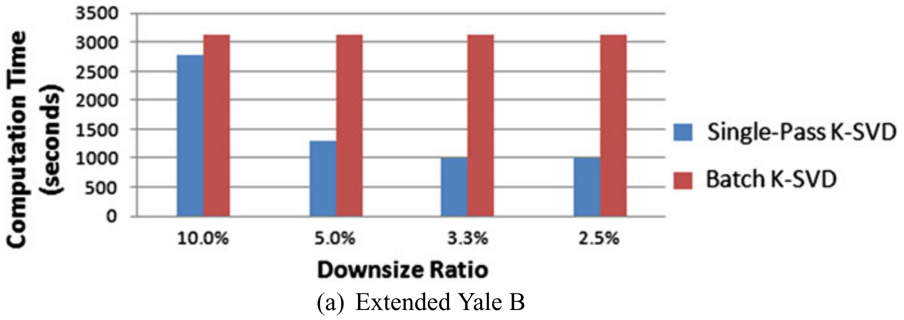


(a) MNIST



(b) Extended Yale B

Fig. 2 Reconstruction errors and relative errors of various downsize ratios (under $s = 3$)



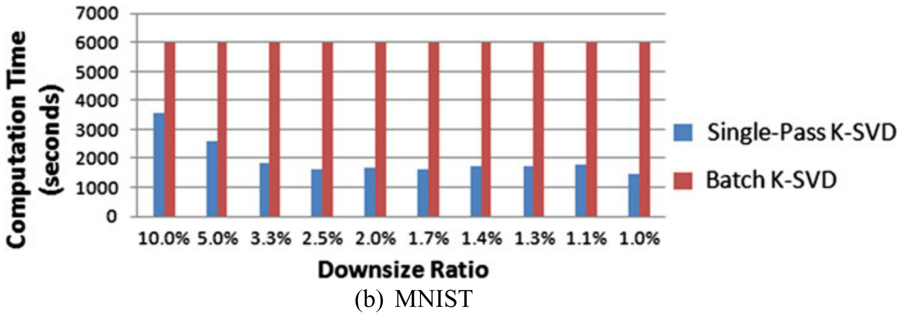


Fig. 3 Computation time when downsize ratio is getting decreased

4.3 Computation Time

Our approach has the advantage that it requires only one scan of the dataset (with no re-scans needed). The process is suspendible since resuming a stopped job is allowed. It is thus helpful for scaling K-SVD to large databases and works within a given limited RAM buffer.

The total computation time of single-pass K-SVD is the sum of the times of all runs. Since the time complexities of both OMP and rank-one-SVDs are linear with the data size (for either batch or single-pass K-SVD), the computation time depends mainly on the number of iterations which is data-dependent. Figure 3 shows the computation times for Extended Yale B and MINST when a 3.07 GHz PC is used. It can be seen that the speed of single-pass K-SVD is faster in general. In particular, for the larger dataset and smaller downsize ratio, the computation time is decreased in overall. For example, when the downsize ratio is smaller than 2.5 %, the computational speeds are more than three-times faster than that of batch K-SVD for the MNIST dataset.

5 Conclusions and Discussion

We introduce a weighted single-pass K-SVD that can learn an approximation of K-SVD well by one scan of a dataset. It can incorporate additional data with existing learned K-SVD models efficiently, and also has the on-line behavior where the most appropriate dictionary can be found by updating weighted new data. Experimental results show that our approach is effective in dictionary learning by approximating K-SVD's performance with limited storage resource. Despite K-SVD is extended to single-pass K-SVD in this work, the same principle can be applied to the extension of other sparsity-based dictionary-learning approaches as well.

Acknowledgements This work was supported in part by the National Science Council, Taiwan, under the grants NSC101-2221-E-001-011.

References

1. The K-SVD Matlab software available at: http://www.cs.technion.ac.il/~elad/software/
2. V. Abolghasemi, S. Ferdowsi, S. Sanei, Sparse multichannel source separation using incoherent K-SVD method, in *Statistical Signal Processing Workshop* (2011)
3. M. Aharon, M. Elad, A. Bruckstein, K-SVD and its non-negative variant for dictionary design, in *Proceedings of the SPIE Conference Wavelets* (2005)
4. M. Aharon, M. Elad, A. Bruckstein, K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans. Signal Process. **54**(11), 4311–4322 (2006)
5. D. Arthur, S. Vassilvitskii, k-Means++: the advantages of careful seeding, in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2007)
6. C. Bishop, *Pattern Recognition and Machine Learning* (2006)
7. P.S. Bradley, U. Fayyad, C. Reina, Scaling clustering algorithms to large databases, in *Knowledge Discovery and Data Mining* (1998), pp. 9–15
8. A. Bruckstein, D. Donoho, M. Elad, From sparse solutions of systems of equations to sparse modeling of signals and images. SIAM Rev. **51**(1), 34 (2009)
9. E. Candes, T. Tao, Decoding by linear programming. IEEE Trans. Inf. Theory **51**(12), 4203–4215 (2005)
10. S. Chen, D. Donoho, M. Saunders, Atomic decomposition by basis pursuit, in *Society for Industrial and Applied Mathematics Review* (2001), pp. 129–159
11. W. Dai, T. Xu, W. Wang, Dictionary learning and update based on simultaneous codeword optimization (SIMCO), in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (2012)
12. F. Farnstrom, J. Lewis, C. Elkan, Scalability for clustering algorithms revisited. ACM SIGKDD Explor. Newsl. **2**(1), 51–57 (2000)
13. X. He, R. Song, W. Zhu, Optimal pilot pattern design for compressed sensing-based sparse channel estimation in OFDM systems, in *Circuits, Systems, and Signal Processing* (2012), pp. 1–17
14. P. Hore, L.O. Hall, D.B. Goldgof, Single pass fuzzy C means, in *IEEE International Conference on Fuzzy Systems* (2007)
15. M. Hyder, K. Mahata, An improved smoothed approximation algorithm for sparse representation. IEEE Trans. Signal Process. **58**(4), 2194–2205 (2010)
16. Z. Jiang, Z. Lin, L. Davis, Learning a discriminative dictionary for sparse coding via label consistent K-SVD, in *IEEE Conference on Computer Vision and Pattern Recognition* (2011)
17. S. Lloyd, Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
18. J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online dictionary learning for sparse coding, in *Proceedings of the 26th Annual International Conference on Machine Learning* (2009)
19. J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online learning for matrix factorization and sparse coding. J. Mach. Learn. Res. **11**, 19–60 (2010)
20. J. Mairal, M. Elad, G. Sapiro, Sparse learned representations for image restoration, in *The 4th World Conference of the IASC and 6th Conference of the Asian Regional Section of the IASC on Computational Statistics and Data Analysis* (2008)
21. V. Patel, Y. Shi, P. Thompson, A. Toga, K-SVD for HARDI denoising, in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro* (2011)
22. Y. Pati, R. Rezaiifar, P. Krishnaprasad, Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition, in *Asilomar Conference on Signals, Systems and Computers* (1993)

23. R. Rubinstein, M. Zibulevsky, M. Elad, Double sparsity: learning sparse dictionaries for sparse signal approximation. IEEE Trans. Signal Process. **58**(3), 1553–1564 (2010)
24. Q. Zhang, B. Li, Discriminative K-SVD for dictionary learning in face recognition, in *IEEE Conference on Computer Vision and Pattern Recognition* (2010)
25. Y. Zhao, X. Zhuang, H. Wang, Z. Dai, Model-based multichannel compressive sampling with ultra-low sampling rate, in *Circuits, Systems, and Signal Processing* (2012), pp. 1–12