Compressed Nonnegative Sparse Coding

Fei Wang
Department of Statistical Science
Cornell University
Ithaca, NY 14853, USA
fw83@cornell.edu

Ping Li
Department of Statistical Science
Cornell University
Ithaca, NY 14853, USA
pingli@cornell.edu

***Abstract*—Sparse Coding (SC), which models the data vectors as sparse linear combinations over basis vectors, has been widely applied in machine learning, signal processing and neuroscience. In this paper, we propose a *dual random projection* method to provide an efficient solution to Nonnegative Sparse Coding (NSC) using small memory. Experiments on real world data demonstrate the effectiveness of the proposed method.**

I. Introduction

Recent years have witnessed a surge of interests in *Nonnegative Matrix Factorization* (*NMF*) [9], [10], [19], [26] and *Nonnegative Sparse Coding* (*NSC*) [5], [6], [7], [17].

NMF is an effective factorization method for decomposing multivariate data into nonnegative components. It is believed that NMF is capable of producing interpretable representations of the data owing to the additive combination of the components [3], [9]. One beneficial "side effect" of NMF is that it often produces *sparse* representations [7], encoding much of the data with only a few *active* components. This property further enhances the interpretability. In fact, it has been shown that sparse learned models are well adapted to natural signals [16], [22], [21]. Recently, *Nonnegative Sparse Coding* (*NSC*) has been proposed to extend the original NMF to explicitly control the sparseness.

A. Nonnegative Sparse Coding

We first focus on the NSC problem with the sparseness penalty only on the coefficients [6][5][17]. Later we will show that our framework can be easily adapted to NSC with penalties on both coefficients and basis vectors [7].

Consider a data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the i-th data vector. NSC aims to learn a basis matrix $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_r] \in \mathbb{R}^{d\times r}$, such that $\mathbf{f}_j \in \mathbb{R}^d$ is the j-th basis vector, together with a combination coefficient matrix $\mathbf{G} = [\mathbf{g}_1, \mathbf{g}_2, \cdots, \mathbf{g}_n] \in \mathbb{R}^{r\times n}$ by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{F},\mathbf{G}} \quad & \sum_i^n \|\mathbf{x}_i - \mathbf{F}\mathbf{g}_i\|^2 + \lambda |\mathbf{g}_i|_1 \\ s.t. \quad & \mathbf{F} \geqslant 0,\ \mathbf{G} \geqslant 0 \end{aligned} \tag{1}$$

Here, λ is a constant to trade off the reconstruction loss for the sparsity of $\mathbf{G}$. $\|\mathbf{z}\| = \sqrt{\sum_i z_i^2}$ is the ℓ_2 norm and $|\mathbf{z}|_1 = \sum_i |z_i|$ is the ℓ_1 norm. $\mathbf{F} \geqslant 0$ means $F_{ij} \geqslant 0\ \forall\ i, j$.

Without loss of generality, we assume $\|\mathbf{f}_j\| = 1$, $j = 1$ to r. This is to prevent $\mathbf{F}$ from being arbitrarily large (which would result in arbitrarily small $\mathbf{G}$) [5].

Note that the optimization problem (1) is in general difficult because it is non-convex over $\mathbf{F}$ and $\mathbf{G}$ jointly. A common strategy is to apply alternating optimization [10][5][17].

1) : In Problem (1), if $\mathbf{F}$ is fixed, then the optimization problem over $\mathbf{G}$ can be decomposed into n independent ℓ_1 constrained optimization problems. That is, for $i = (1, 2, \cdots, n)$,

$$\begin{aligned} \min_{\mathbf{g}_i} \quad & \|\mathbf{x}_i - \mathbf{F}\mathbf{g}_i\|^2 + \lambda |\mathbf{g}_i|_1 \\ s.t. \quad & \mathbf{g}_i \geqslant 0, \end{aligned} \tag{2}$$

which can be solved by the nonnegative LARS-LASSO algorithms [23], [4], [17].

2) : Fixing $\mathbf{G}$, the optimization problem (1) becomes

$$\begin{aligned} \min_{\mathbf{F}} \quad & \sum_i^n \|\mathbf{x}_i - \mathbf{F}\mathbf{g}_i\|^2 = \|\mathbf{X} - \mathbf{F}\mathbf{G}\|_F^2 \\ s.t. \quad & \mathbf{F} \geqslant 0, \end{aligned} \tag{3}$$

which can be solved by the following normalization invariant multiplicative update rule [5]:

$$\mathbf{F} \longleftarrow \mathbf{F} \odot \frac{\mathbf{X}\mathbf{G}^{\mathrm{T}} + \mathbf{F}\mathrm{diag}\left(\mathbf{1}^{\mathrm{T}}\left(\mathbf{F}\mathbf{G}\mathbf{G}^{\mathrm{T}} \odot \mathbf{F}\right)\right)}{\mathbf{F}\mathbf{G}\mathbf{G}^{\mathrm{T}} + \mathbf{F}\mathrm{diag}\left(\mathbf{1}^{\mathrm{T}}\left(\mathbf{X}\mathbf{G}^{\mathrm{T}} \odot \mathbf{F}\right)\right)} \tag{4}$$

where the multiplication ($\odot$) and division are element-wise operations, $\mathbf{1}$ is an all-one column vector.

B. The Storage and Computational Bottleneck of NSC

Current solutions of NSC require that the data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ reside in the memory. This seriously limits the applicability of NSC to real world problems when both the number of samples (n) and dimensions (d) are very large. For example, 10 million images, each of size 1000×1000, would not fit in the memory (using pixel representations).

A somewhat secondary issue is that when the data $\mathbf{X}$ is large, the matrix-vector multiplications in the algorithms could be prohibitive. For example, $\mathbf{X}\mathbf{G}^{\mathrm{T}}$ in Eq. (4) would be expensive when both n and d are large.

C. CNSC: Compressed Nonnegative Sparse Coding

This paper proposes *Compressed Nonnegative Sparse Coding (CNSC)* to overcome the storage and computational problems of the current NSC solutions, based on a *dual random projection* method.

Random projection [24], [1], [11], [14], [12] is an effective randomized algorithm for solving many large scale computational problems. The basic idea is that, if one multiplies the data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ by a random matrix $\mathbf{R} \in \mathbb{R}^{n\times k}$ whose entries are sampled from i.i.d. standard normals $N(0,1)$, the resultant matrix $\mathbf{B} = \mathbf{X} \times \mathbf{R}$ fairly accurately preserves all pairwise l_2 distances and inner products of $\mathbf{X}$.[1]

For NSC, we have to conduct projections in both directions, that is, $\mathbf{X} \times \mathbf{R}_n$ and $\mathbf{R}_d \times \mathbf{X}$, using two independent random projection matrices, $\mathbf{R}_n$ and $\mathbf{R}_d$.

The prior work [25] developed an efficient random projection algorithm for NMF. This paper is a natural extension. We would like to point out that, although we focus on *nonnegative* sparse coding, our method will be applicable to sparse coding without the nonnegativity constraint.

II. Compressed Nonnegative Sparse Coding

As introduced in section I, *Nonnegative Sparse Coding (NSC)* proceeds by solving $\mathbf{G}$ and $\mathbf{F}$ alternatingly. The proposed *Compressed Nonnegative Sparse Coding (CNSC)* adopts a similar approach.

A. Solving $\mathbf{G}$ with $\mathbf{F}$ Fixed

In this case, we need to solve a "compressed" version of the optimization problem (2):

$$\begin{aligned} \min_{\mathbf{g}_i} \quad & \left\|\mathbf{R}_d\mathbf{x}_i - \mathbf{R}_d\mathbf{F}\mathbf{g}_i\right\|^2 + \lambda\left|\mathbf{g}_i\right|_1 \\ s.t. \quad & \mathbf{g}_i \geqslant 0 \end{aligned} \tag{5}$$

where $\mathbf{R}_d \in \mathbb{R}^{k_d\times d}$ is a random matrix whose entries are sampled from i.i.d. standard normals $N(0.1)$. Problem (5) is still a standard nonnegative least square regression problem with ℓ_1 penalty. We used the *NLARS* code [17][2]. At each step, we only need to solve a much smaller problem because $\mathbf{R}_d\mathbf{x}_i \in \mathbb{R}^{k_d\times 1}$ instead of $\mathbb{R}^{d\times 1}$. Our experiments will show that using about $k_d = 500$ (while the original problem may have d in millions or more) can provide a solution which is sufficiently close to the original solution.

The computational complexity of the original problem (2) can reach $O(d^2n)$. For high dimensional data sets, this would be computationally prohibitive, even assuming that one can store the original data matrix $\mathbf{X}$ in the memory.

[1]Random projection is particularly effective for preserving the l_2 distances, with a strong guarantee known as the *Johnson-Lindenstrauss (JL)* Lemma [8]. The guarantee is weaker in terms of preserving the inner products; see [11], [13] for the detailed analysis of the estimation variances.

[2]http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=5523.

B. Solving $\mathbf{F}$ with $\mathbf{G}$ Fixed

Instead of solving the original problem (3), we solve the compressed version:

$$\begin{aligned} \min_{\mathbf{F}} \quad & \left\|\mathbf{X}\mathbf{R}_n - \mathbf{F}\mathbf{G}\mathbf{R}_n\right\|_F^2 \\ s.t. \quad & \mathbf{F} \geqslant 0 \end{aligned} \tag{6}$$

where $\mathbf{R} \in \mathbb{R}^{n\times k_n}$ is another random matrix whose entries are sampled from i.i.d. $N(0,1)$.

However, after the compression, we cannot directly use the update formula Eq. (4), because $\mathbf{F}\mathbf{R}_n$ and $\mathbf{G}\mathbf{R}_n$ are no longer nonnegative. While we can apply strategies like active set [20] or projected gradient [15], we find that these methods are slow when the problem size r is relatively large (e.g., $r = 25$). Here we adopt the following rule for updating $\mathbf{F}$:

$$\mathbf{F} \longleftarrow \mathbf{F} \odot \sqrt{\frac{\mathbf{\Gamma}_+ + \mathbf{F}\mathbf{\Theta}_- + \mathbf{F}\mathrm{diag}\left[\mathbf{1}^{\mathrm{T}}((\mathbf{\Gamma}_- + \mathbf{F}\mathbf{\Theta}_+)\odot\mathbf{F})\right]}{\mathbf{\Gamma}_- + \mathbf{F}\mathbf{\Theta}_+ + \mathbf{F}\mathrm{diag}\left[\mathbf{1}^{\mathrm{T}}((\mathbf{\Gamma}_+ + \mathbf{F}\mathbf{\Theta}_-)\odot\mathbf{F})\right]}} \tag{7}$$

where

$$\mathbf{\Gamma} = \quad \mathbf{X}\mathbf{R}_n\mathbf{R}_n^{\mathrm{T}}\mathbf{G}^{\mathrm{T}} \tag{8}$$

$$\mathbf{\Theta} = \quad \mathbf{G}\mathbf{R}_n\mathbf{R}_n^{\mathrm{T}}\mathbf{G}^{\mathrm{T}} \tag{9}$$

and $\mathbf{A}_+ = (|\mathbf{A}| + \mathbf{A})/2$, with $|\cdot|$ being the element-wise absolute value, is the positive part of matrix $\mathbf{A}$; and $\mathbf{A}_- = (|\mathbf{A}| - \mathbf{A})/2$ is the negative part of $\mathbf{A}$.

In other words, Eq. (7) is a simple variation of Eq. (4) by separating the positive and negative parts. Eq. (7) is derived using the same strategy as in *Semi-NMF* [2].

Thus, we can update $\mathbf{G}$ and $\mathbf{F}$ alternatingly until a local equilibrium is reached. The overall CNSC algorithm is summarized in Algorithm 1.

Algorithm 1 Compressed NSC

Require: Data Matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, Projection Matrix $\mathbf{R}_d \in \mathbb{R}^{k_d\times d}$, Projection Matrix $\mathbf{R}_n \in \mathbb{R}^{n\times k_n}$, Positive integer r, Number of iterations T

1: Construct the compressed data matrix $\widetilde{\mathbf{X}} = \mathbf{R}_d\mathbf{X}$ and $\bar{\mathbf{X}} = \mathbf{X}\mathbf{R}_n$.
2: Randomly initialize $\mathbf{F}^{(0)} \in \mathbb{R}^{d\times r}$ to be a nonnegative matrix
3: **for** $t = 1 : T$ **do**
4: Compress $\mathbf{F}$ by $\widetilde{\mathbf{F}} = \mathbf{R}_d\mathbf{F}$
5: **for** $i = 1 : n$ **do**
6: Solve the i-th column of $\mathbf{G}$, Problem (5), by the NLARS Algorithm
7: **end for**
8: Compress $\mathbf{G}$ by $\bar{\mathbf{G}} = \mathbf{G}\mathbf{R}_n$
9: Construct $\mathbf{\Gamma}$ and $\mathbf{\Theta}$ as in Eq. (8) and Eq. (9), and update $\mathbf{F}$ using Eq. (7).
10: **end for**
11: Output $\mathbf{G}$ and $\mathbf{F}$

Table I
THE BASIC INFORMATION OF THE DATA SETS

	Dimensionality (d)	Size (n)
Yale	1024	165
YaleB	1024	2,124
COIL	16384	7,200
PIE	1024	11,554
SecStr	315	1,273,151

III. EXPERIMENTS

This section presents a set of experiments to demonstrate the effectiveness of the proposed CNSC method. Table I summarizes the information about the data sets[3].

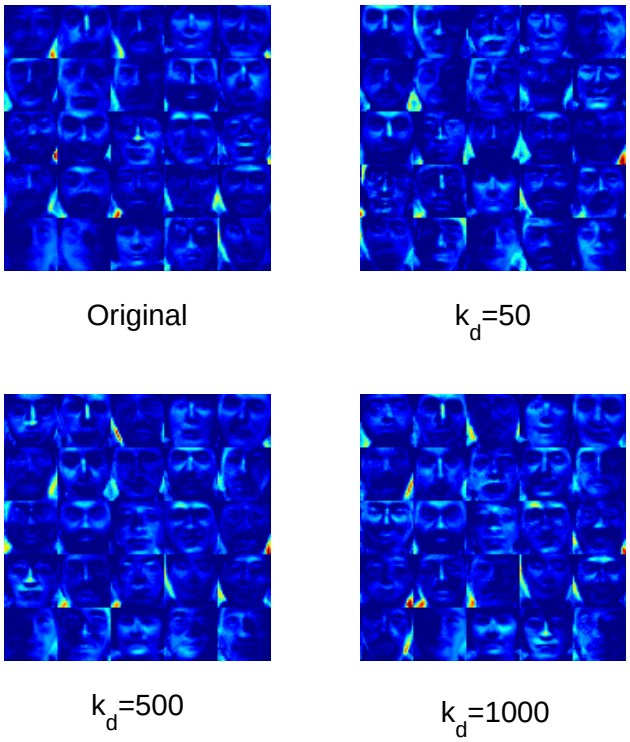


Figure 1. **Yale**: The learned face dictionary. Different figures correspond to different compressed dimensions with the same initialization of $\mathbf{F}$.

A. Yale Face Data Set

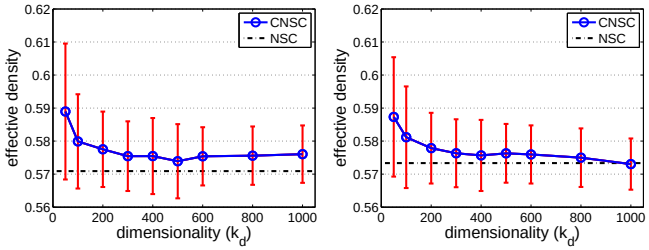
The Yale Face data set contains 165 gray scale of 15 individuals. There are 11 images per subject, one per different facial expression or configuration. The faces have been cropped from the original images and resized to 32×32. We compare our CNSC algorithm with the sparse coding algorithm in [17]. For both algorithms, the dictionary $\mathbf{F}$ is randomly initialized, and the number of iterations is set to 200. The objective function loss at step t is computed as

$$J(\mathbf{F}^{(t)}, \mathbf{G}^{(t)}) = \|\mathbf{X} - \mathbf{F}^{(t)}\mathbf{G}^{(t)}\|_F^2 + \lambda\|\mathbf{G}^{(t)}\|_1 \qquad (10)$$

[3]http://www.zjucadcg.cn/dengcai/Data/FaceData.html

We set $r = 25$ and let the *NLARS* code choose λ (for the smallest function loss). For this data set, as $n = 165$ is very small, we only compress $\mathbf{F}$ when solving $\mathbf{G}$ in Problem (5). We construct a normal random matrix $\mathbf{R}_d \in \mathbb{R}^{k_d \times d}$ with $k_d = 50, 100, 200, \cdots, 1000$, and run algorithm 1 with $\mathbf{R}_n = \mathbf{I}_{n\times n}$. For each k_d value, we conduct 100 independent runs with the same initialization and report the statistical performance.

Fig.1 illustrates the learned face dictionary $\mathbf{F}$ with a specific initialization for the original algorithm in [17] and our CNSC algorithm with $k_d = 50, 500, 1000$. From the figure we can see that with increasing k_d, the learned dictionary would become more like the dictionary learned from the original uncompressed problem, and the learned $\mathbf{F}$ with $k_d = 500$ and $k_d = 1000$ are quite similar, which indicates that $k_d = 500$ is enough for this data set. Moreover, we also compute the effective density of $\mathbf{G}$ (which is the number of nonzero elements in $\mathbf{G}$ divided by $r \times n$), for comparing CNSC with NSC. The results are shown in Fig. 2, which illustrates that the sparsity of $\mathbf{G}$ are well preserved for CNSC. Moreover, the larger the compressed dimension, the better the preservation would be.

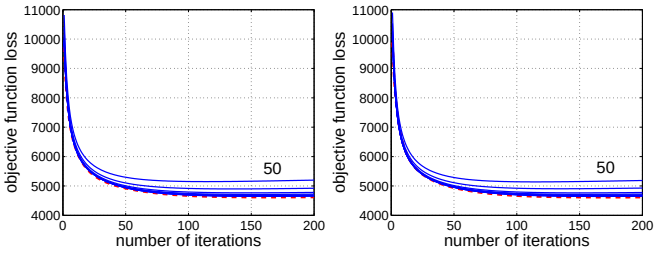


(a) Initialization 1 (b) Initialization 2

Figure 2. **Yale**: Effective density of $\mathbf{G}$ using CNSC and 2 different random initializations of $\mathbf{F}$. The y-axis corresponds to the (normalized) ℓ_0 norm of $\mathbf{G}$ after 200 iterations divided by its size, and the x-axis represents different projected dimensions k_d (50 to 1000). The solid lines are averaged 100 independent runs with the standard deviation shown as error bars. The dashed line is the effective density of $\mathbf{G}$ resulting from original NSC.

Fig. 3 plots the variation of the objective function loss with respect to the number of iterations for the CNSC method with 2 different random initializations of $\mathbf{F}$. The solid lines correspond to different projected dimensionalities, which are averaged over 100 independent runs. The dashed lines correspond to the original NSC without random projections. The initializations of $\mathbf{F}$ are set to be the same for NSC with or without random projections. As the projected dimensionality increases, the Frobenius loss curves of CNSC become closer to the original NSC curves.

Fig. 4 shows the relative loss (averaged over 100 independent runs with standard deviation) vs. projected dimensionality plots after 200 iterations, for two different initializations of $\mathbf{F}$. Here relative loss after T iterations at a specific



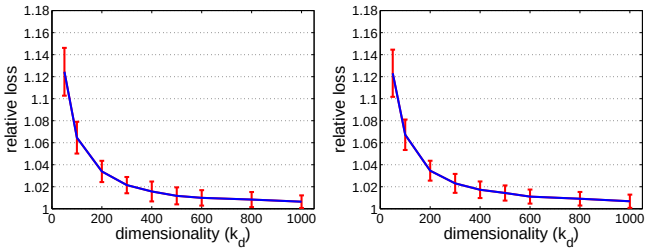
(a) Initialization 1 (b) Initialization 2

Figure 3. **Yale**: Objective function loss variation over 200 iterations using CNSC with 2 different random initializations of $\mathbf{F}$. The dashed line is for the original NSC. The solid lines are the averaged (over 100 independent runs) plots of CNSC ($k_d = 50$ to 1000 from top to bottom).

projected dimension is computed as

$$RL(T) = J(\widetilde{\mathbf{F}}^{(T)}, \widetilde{\mathbf{G}}^{(T)})/J(\mathbf{F}^{(T)}, \mathbf{G}^{(T)}) \qquad (11)$$

where $\widetilde{\mathbf{F}}^{(T)}$ and $\widetilde{\mathbf{G}}^{(T)}$ are the matrices learned by CNSC after $T = 200$ iterations, while $\mathbf{F}^{(T)}$ and $\mathbf{G}^{(T)}$ are learned from original NSC. Clearly, the closer $r(T)$ to 1, the better the approximation will be. Fig.4 shows that larger projected dimensions will lead to better and more stable approximations. When $k_d = 200$, the relative error would become less than 5% (i.e., 1.05 in the y-axis in Fig. 4). We also compare the computational time of CNSC in each round of updating $\mathbf{F}$ and $\mathbf{G}$, with the updating time of the original NSC, across different projected dimensions. The results are shown in Fig. 5(a), which clearly demonstrates the computational efficiency of our compression strategy.



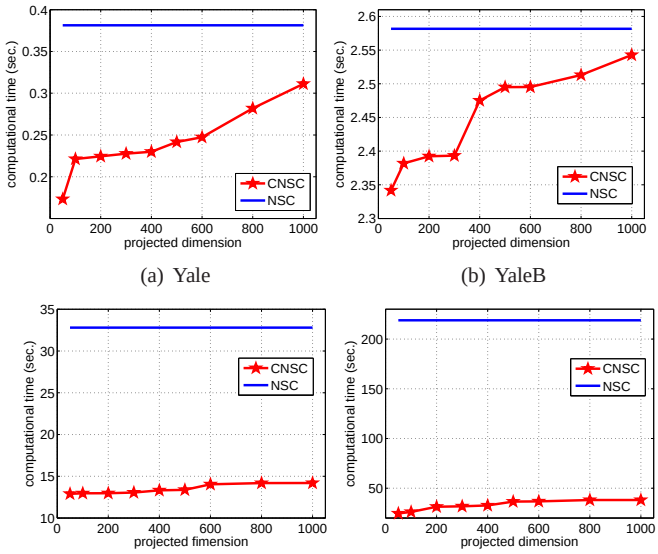
(a) Initialization 1 (b) Initialization 2

Figure 4. **Yale**: Relative loss using CNSC using 2 different random initializations of $\mathbf{F}$. The y-axis corresponds to the final relative loss after 200 iterations, and the x-axis represents different projected dimensions k_d (50 to 1000). The solid lines are averaged 100 independent runs with the standard deviation shown as error bars.

B. Experiments on YaleB Face Data Set

The YaleB data set we used is a sub data set from the extended Yale face database[4]. It has 38 individuals and around 64 near frontal images under different illuminations per individual. In total there are 2414 face images, each of size 32×32 (i.e., the dimensionality = 1024).

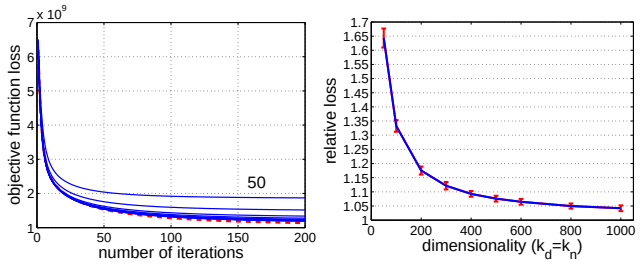
[4]http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html



(a) Yale (b) YaleB

(c) PIE (d) COIL

Figure 5. Computational time comparison of CNSC and original NSC. The x-axis corresponds to the projected dimensionality, y-axis represents the averaged computational time (over 50 independent runs) for each updating iteration in seconds. The figure shows that the larger the original data matrix, the more significant the speedup of CNSC is.



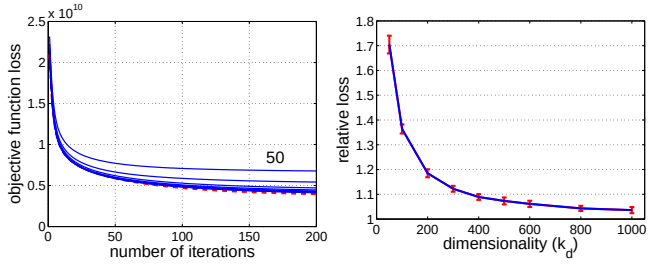
(a) Function Loss (b) Relative Loss

Figure 6. **YaleB**: Objective function loss and Relative loss using CNSC and one random initialization of $\mathbf{F}$. (a) shows the objective function loss vs. number of iterations plot, The dashed line is the plot of original NSC. The solid lines are the averaged (over 100 independent runs) plots of CNSC with random projections ($k_d = k_n = 50$ to 1000 from top to bottom). (b) shows the final relative loss after 200 iterations vs. compressed dimensionality. The solid lines are averaged 100 independent runs with the standard deviation shown as error bars.

In our experiments, we also set the number of the basis face vectors $r = 25$. For CNSC, we implemented algorithm 1 with $k_d = k_n = 50, 100, 200, \cdots, 1000$ for simplicity, i.e., the compressed dimensionalities for Problem (5) and Problem (6) are set to be the same when solving $\mathbf{G}$ and $\mathbf{F}$.

Fig. 6 (a) shows the variations of the objective function loss with respect to the number of iterations for original NSC (in dashed line) and CNSC (in solid lines, which are averaged over 100 independent runs). From the figure we can clearly see that with the increase of the compressed dimensionality (k_n and k_d), the resultant curve would become closer to the curve derived from original NSC. Fig. 6(b)

suggests that when $k_d = k_n$ becomes larger than 400, the final relative loss would be within 1.1. We also record the effective density of the final $\mathbf{G}$ after 200 iterations as shown in Fig. 8(a), from which we can observe that the sparsity of $\mathbf{G}$ is well preserved. Also we can see that the larger the compressed dimensionality, the better the preservation. The computational time comparison is provided in Fig. 5(b), which also shows the computational advantage of CNSC over NSC.

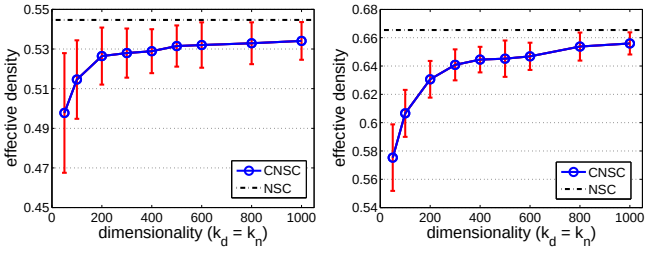


(a) Function Loss (b) Relative Loss

Figure 7. **PIE**: Objective function loss and relative loss using CNSC and one random initialization of $\mathbf{F}$. The meanings of the axes and curves are the same as in Fig. 6.

C. Experiments on PIE Data Set

The data set we used is a subset from the PIE face database[5], which contains the near frontal face images of 68 people, with a total of 11554 images. Each image is resized to 32×32. In our experiments, we also set $r = 25$ and $k_d = k_n = 50$ to 1000, and we report the objective function loss, final relative loss and final sparsity of $\mathbf{G}$ in Fig. 7(a), Fig. 7(b) and Fig. 8(b). From these figures we can observe similar pattern on the approximation of CNSC to NSC as what we see for the Yale and YaleB data sets. The computational time comparison of CNSC and NSC is shown in Fig. 5(c), where there is a large gap between the red curve and blue line, which suggests a significant speedup when applying the compression strategy on this data set.



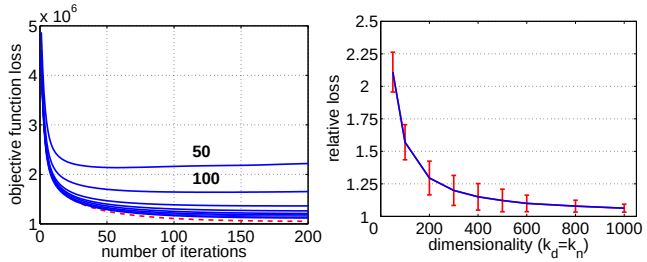
(a) YaleB (b) PIE

Figure 8. Effective density of G on YaleB and PIE face data using CNSC and one random initializations of $\mathbf{F}$. The meanings of the axes are the same as in Fig. 2. The solid lines are averaged 100 independent runs with the standard deviation shown as error bars. The dashed line is the effective density of $\mathbf{G}$ resulting from original NSC.

[5]www.ri.cmu.edu/research_project_detail.html?project_id=418&menu_id=261

D. Experiments on COIL Data Set

COIL-100 [18] is an object recognition data set, containing pictures of 100 different objects. Each object has 72 pictures taken from different angles. All pictures are of size 128×128, with a total of 16384 pixels. In our experiments, we also set $r = 25$ and $k_d = k_n = 50$ to 1000, and we report the objective function loss and final relative loss in Fig. 9(a) and Fig. 9(b). These figures exhibit similar trends as those on the Yale and YaleB data Sets. The computational time comparison of CNSC and NSC is shown in Fig. 5(d). The speedup is very significant owing to the large compression ratio on this data set.

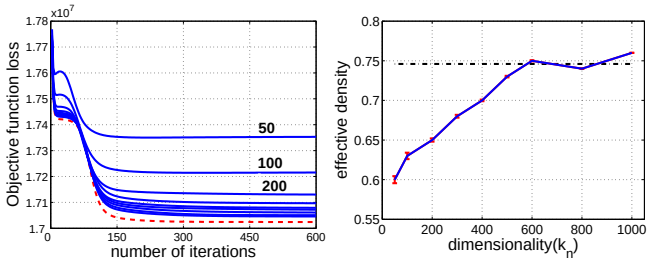


(a) Function Loss (b) Relative Loss

Figure 9. **COIL**: Objective function loss and Relative loss using CNSC and one random initialization of $\mathbf{F}$. The meanings of the axes and curves are the same as in Fig. 6.

E. Experiments on Secstr Data Set

Secstr is a bioinformatics data set for predicting the secondary structure of a given amino acid in a protein based on a sequence window centered around that amino acid.[6]. As the data scale is very large (over 1 million) and the data dimensionality is very small (315), we adopt a one-side compression, i.e., we only compress on the data scale side but leave the data dimension side unchanged. In our experiments, we set $r = 100$ and $k_n = 50$ to 1000, and we report the objective function loss and the effective density variation in Fig. 10(a) and Fig.10(b), respectively. The results on this data set again verify the effectivness of the proposed method.



(a) Function Loss (b) Relative Loss

Figure 10. **SecStr**: Objective function loss and effective density using CNSC and one random initialization of $\mathbf{F}$.

[6]http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.html

IV. Conclusion

In this paper, we propose *Compressed Nonnegative Sparse Coding (CNSC)*, a *dual random projection* strategy to significantly overcome the storage and computational bottlenecks of *Nonnegative Sparse Coding (NSC)*, a method that has been widely applied in machine learning, signal processing and neuroscience. With CNSC, we only need to store compressed versions of the original data matrix, whose size in these days may well exceed the memory capacity. Experimental results on real world data sets demonstrate the effectiveness of the proposed CNSC algorithm.

Acknowledgement

This work is partially supported by NSF (DMS-0808864), ONR (YIP-N000140910911), and a grant from Microsoft.

References

[1] D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003.

[2] C. Ding, T. Li, and M. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:45–55, 2009.

[3] D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts. In *NIPS 17*, 2004.

[4] B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[5] J. Eggert and E. Korner. Sparse coding and nmf. In *IJCNN*, volume 4, pages 2529–2533, 2004.

[6] P. O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing, 2002. Proceedings of the 12th IEEE Workshop on*, pages 557–565, 2002.

[7] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[8] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[9] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[10] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[11] P. Li, T. Hastie, and K. W. Church. Very sparse random projections. In *KDD 12*, pages 287–296, 2006.

[12] Ping Li. Computationally efficient estimators for dimension reductions using stable random projections. In *ICDM*, Pisa, Italy, 2008.

[13] Ping Li, Trevor J. Hastie, and Kenneth W. Church. Improving random projections using marginal information. In *COLT*, pages 635–649, Pittsburgh, PA, 2006.

[14] Ping Li, Trevor J. Hastie, and Kenneth W. Church. Nonlinear estimators and tail bounds for dimensional reduction in l_1 using cauchy random projections. *Journal of Machine Learning Research*, 8:2497–2532, 2007.

[15] C. J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, 19:2756–2779, 2007.

[16] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *CVPR*, pages 1–8, 2008.

[17] M. Mørup, K. H. Madsen, and L. K. Hansen. Approximate l0 constrained non-negative matrix and tensor factorization. In *Proceedings of The IEEE International Symposium on Circuits and Systems*, pages 1328–1331, 2008.

[18] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-100). *Technical Report CUCS-006-96*, 1996.

[19] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[20] H. Park and H. Kim. Nonnegative matrix factorization based on alternating non-negativity-constrained least squares and the active set method. *SIAM Journal on Matrix Analysis and Applications*, 30(2):713–730, 2008.

[21] M. Protter and M. Elad. Image sequence denoising via sparse and redundant representations. *IEEE Transactions on Image Processing*, 18(1):27–36, 2009.

[22] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML 24*, pages 759–766, 2007.

[23] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1):267–288, 1996.

[24] Santosh Vempala. *The Random Projection Method*. American Mathematical Society, Providence, RI, 2004.

[25] Fei Wang and Ping Li. Efficient non-negative matrix factorization with random projections. In *Proceedings of The 10th SIAM International Conference on Data Mining*, pages 281–292, 2010.

[26] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *SIGIR 26*, pages 267–273, 2003.