# A Fast Dictionary Learning Algorithm via Codeword Clustering and Hierarchical Sparse Coding

**By Tao Xu[1], Wenwu Wang[1], Wei Dai[2]**

[1] Department of Electronic Engineering, University of Surrey, UK
[2] Department of Electrical and Electronic Engineering, Imperial College London, UK
Emails: [t.xu; w.wang]@surrey.ac.uk; wei.dai1@imperial.ac.uk

**Abstract**

Dictionary learning algorithms, aiming to learn a sparsifying transform from training data, are often established on an optimization process involving the iterations between two stages: sparse coding and dictionary update. In practice, however, these algorithms are often computationally demanding especially when dealing with large scale data or high dimensional signals. In this paper, we propose new methods for improving the computational efficiency of dictionary learning algorithms. Specifically, we develop a tree-structured multi-level representation of dictionary based on clustering, which is used to derive a hierarchical algorithm in the sparse coding stage. The proposed idea is then applied to the simultaneous codeword optimisation (SimCO) algorithm, a dictionary learning algorithm that we developed recently, resulting in a new algorithm: fast SimCO. Numerical examples are provided to show its computational efficiency and the performance for image denoising.

## 1. Introduction

Sparse representation is an emerging technique in signal processing, aiming to address the problem of how to represent a natural signal as the combination of only a few elementary components (i.e. codewords or atoms) chosen from a dictionary (i.e. the collection of all the atoms), assuming that the signals are naturally sparse in one domain or can be made sparser in another domain using e.g. a transform. This technique can be used in a number of applications, e.g. image denoising and source separation, and was demonstrated to be useful for large scale and highly redundant data.

The performance of sparse representation relies greatly on the fit between the signals of interest and the dictionary that is used to sparsify the signals. Either a pre-defined dictionary (obtained by an analytical transform) or a learned dictionary (based on a machine learning or optimisation algorithm) has been used in the literature to serve for this purpose. Learning a dictionary from training data, as shown recently, may give a better fit to the signals, and therefore has the potential to offer better performance than a predefined dictionary. The learning process, however, may involve a higher computational complexity, rendering the algorithms to be less practical in computation extensive applications, for example, when dealing with large scale or high-dimensional data.

We propose a new method to improve the computational efficiency of dictionary learning algorithms based on codeword clustering and hierarchical sparse coding, and we apply this method to our recently proposed dictionary learning algorithm, i.e. simultaneous

codeword optimisation (SimCO) in Dai et al. (2012). The details of the proposed method, i.e. fast SimCO, are given in Section 2, and simulations together with the analysis of the results are provided in Section 3.

## 2. The Proposed Method

The dictionary learning task is usually achieved by alternate iterations between sparse coding and dictionary update. First, given an initial dictionary, a signal is decomposed as a linear combination of only a few atoms. The atoms of the dictionary are trained with fixed or sometimes unfixed weighting coefficients. The trained dictionary is then used to compute the new weighting coefficients. The process is iterated until the most suitable dictionary is obtained eventually based on a pre-defined optimisation criterion. When developing the SimCO algorithm, we also adopted this process, and used the OMP algorithm by Pati et al. (1993) for the sparse coding stage, which aims to solve the optimisation problem: Given data matrix $\mathbf{Y}$, find a sparse coefficient matrix $\mathbf{X}$ to minimize $\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2$ for a given overcomplete dictionary $\mathbf{D}$. In OMP, this is achieved by finding the column vector in $\mathbf{D}$ which most closely resembles a residual vector $\mathbf{r}$, which is initialised to $\mathbf{y}$, then adjusted at each iteration to take into account the vectors previously chosen.

In this work, we propose to make the following improvements to the SimCO algorithm. First, the dictionary atoms obtained in the dictionary update stage of SimCO are clustered using a K-means algorithm. The cluster centers represent a high-level representation of the dictionary, with the atoms in their neighborhoods representing the low-level dictionary. In the sparse coding stage, the closest centroid from the higher level dictionary to the signal under consideration is found, and their neighbors are then used to code this signal with a dimension reduced orthogonal matching pursuit (OMP), based on the nearest neighbor search. A tree structure with multi-level dictionary, obtained by multi-level K-mean clustering, is applied for sparse coding process. We call this new algorithm as the tree-OMP (TOMP) method, summarised in Algorithm 1. The atoms of the dictionary are organized by a tree structure to improve the computational efficiency in the coding stage. In each iteration, the atom that is closest to the residual vector, is selected from the cluster centroids of the atoms in the dictionary, instead of all the atoms in the dictionary, and the coding is performed in the neighborhood of each centroid. Simulations are given in Section 3 to demonstrate its advantage in computational efficiency.

We have also tested an approximate version of TOMP, called the centralized OMP method (COMP), where only the centroid that is closest to the residual vector is selected in each iteration. The only difference is that the sub-optimization problem in step ($d$) of TOMP is now replaced by using the centroid directly in COMP. This apparently improves the computational efficiency but may degrade the sparse coding performance, as shown in the next section.

## 3. Simulation Results

Firstly, we evaluate the proposed fast SimCO algorithm (i.e. using TOMP in the sparse coding stage, but the same dictionary update stage as the original SimCO algorithm) on synthetic data. We compare TOMP with OMP and COMP in the coding stage. As in Dai et al. (2012), we refer to the iterations between sparse coding and dictionary learning stages as outer-iterations, and the iterations within the dictionary update stage

**Algorithm 1** TOMP

**Task**: To find the sparse representation $\mathbf{X}$ from the data sample matrix $\mathbf{Y}$.
**Input**: The initial $m * d$ dictionary $\mathbf{D}$, the $m * n$ matrix $\mathbf{Y}$, and the sparsity $L$.
**Output**: The $d * n$ coefficient matrix $\mathbf{X}$
**Initialize**: The residual $\mathbf{r}_0 = \mathbf{y}$, the index set $\Lambda$ is empty and the iteration counter $j = 1$.
**Do**:

($a$) Cluster the atoms in dictionary $\mathbf{D}$ by K-means algorithm to obtain the centroids of the atoms $\mathbf{d}_{c_1}, ..., \mathbf{d}_{c_e}$, and $e$ is the number of centers.

($b$) Find the index of one of the centers $c_b$ that solves the optimization problem $c_b = \arg\max_{i=1,...e} | < \mathbf{r}_{j-1}, \mathbf{d}_{c_i} > |$.

($c$) Find the index $\lambda_j$ that solves the sub-optimization problem $\lambda_j = \arg\max_j | < \mathbf{r}_{j-1}, \hat{\mathbf{d}}_{c_b} > |$.

($d$) Combine the index set and the chosen atom: $\Lambda_j = \Lambda_{j-1} \bigcup \{\lambda_j\}$ and $\mathbf{D}_j = [\mathbf{D}_{j-1} \mathbf{d}_{\lambda_j}]$.

($e$) Solve the least squares problem to obtain a new coefficient estimate: $\mathbf{x}_j = \arg\max_{\mathbf{x}} \|\mathbf{y} - \mathbf{D}_j \mathbf{x}\|_2$.

($f$) Calculate the new approximation of the data and the new residual $\hat{\mathbf{y}}_j = \mathbf{D}_j \ \mathbf{x}_j$, $\mathbf{r}_j = \mathbf{y}_j - \hat{\mathbf{y}}_j$.

($g$) Increase $j$ and return to Step ($b$) if $j$ is smaller than $L$. This leads to the search of another best centroid.

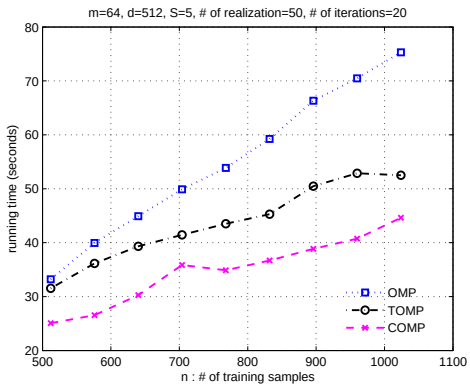


FIGURE 1. Average running time comparison between fast SimCO and the baselines.

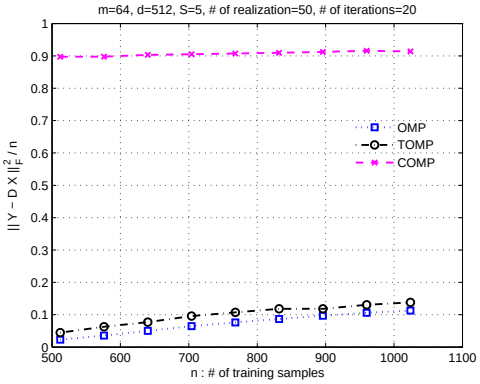


FIGURE 2. Average approximation error comparison between fast SimCO and the baselines.

as inner-iterations. In our tests, all results are averaged over 50 realizations with a random initialization for each realization. The numbers of outer-iterations are set to 20 for all of three algorithms, and in each outer iteration, the numbers of inner-iterations of all the algorithms are set to 1. Furthermore, in dictionary update stage, the regularized constant, as in Dai et al. (2012), is set to $\mu = 1e-1$ during the first 10 outer-iterations, and $\mu = 0$ during the remaining 10 outer-iterations. The average running time by OMP, TOMP and COMP respectively is shown in Figure 1. The approximation error $\|\mathbf{Y} - \mathbf{DX}\|_F^2 / n$ versus $n$ are depicted in Figure 2. It can be observed that TOMP improves computational efficiency over OMP, while maintaining a similar reconstruction performance, and COMP, despite being most efficient, gives the worst performance among the three algorithms.

Secondly, we test the fast SimCO algorithm for image denoising, and compare it with baseline algorithms: MOD by Engan et al. (1999), KSVD by Aharon et al. (2006), and


FIGURE 3. Image denoising examples using the fast SimCO compared with other baseline algorithms. PSNR values in dB are given in the sub-figure titles.

the original SimCO in Dai et al. (2012), using the examples presented in Elad and Aharon (2006). Here, an image corrupted by noise was used to train the dictionary. We take 1000 blocks (with each of size $8 \times 8$ pixels) from the corrupted image as training samples. The number of codewords in the dictionary is set to $d = 256$. The number of outer-iterations and inner-iterations is 10 and 50 respectively. The TOMP algorithm is used in the sparse coding stage. The regularization constant is set to $\mu = 0.05$. We use the learned dictionary to reconstruct the images, and the results are shown in Figure 3. The proposed fast SimCO algorithm offers a similar PSNR to those by the baselines, but only needs 11.56 seconds, as opposed to 14.68 and 18.13 seconds required by the original SimCO and K-SVD respectively. The MOD algorithm needs the shortest running time 7.50 seconds thanks to its second-order operations.

## REFERENCES

Aharon, M., Elad, M., Bruckstein, A., 2006. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans. Signal Process. 54 (11), 4311–4322.

Dai, W., Milenkovic, O., 2009. Subspace pursuit for compressive sensing signal reconstruction. IEEE Trans. Inform. Theory 55, 2230–2249.

Dai, W., Xu, T., Wang, W., 2012. Simultaneous codeword optimization (simco) for dictionary update and learning. IEEE Trans. on Signal Processing.

Elad, M., Aharon, M., dec. 2006. Image denoising via sparse and redundant representations over learned dictionaries. IEEE Transactions on Image Processing 15 (12), 3736 –3745.

Engan, K., Aase, S., Husøy, J. H., 1999. Method of optimal directions for frame design. In: IEEE Int. Conf. Acoustics, Speech, and Signal Processing. Vol. 5. pp. 2443–2446.

Pati, Y. C., Rezaiifar, R., Rezaiifar, Y. C. P. R., Krishnaprasad, P. S., 1993. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In: Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers. pp. 40–44.