A CLUSTERING APPROACH TO OPTIMIZE ONLINE DICTIONARY LEARNING

Nikhil Rao[⋆]
University of Wisconsin-Madison

Fatih Porikli
Mitsubishi Electric Research Labs

ABSTRACT

Dictionary learning has emerged as a powerful tool for low level image processing tasks such as denoising and inpainting, as well as sparse coding and representation of images. While there has been extensive work on the development of online and offline dictionary learning algorithms to perform the aforementioned tasks, the problem of choosing an appropriate dictionary size is not as widely addressed. In this paper, we introduce a new scheme to reduce and optimize dictionary size in an online setting by synthesizing new atoms from multiple previous ones. We show that this method performs as well as existing offline and online dictionary learning algorithms in terms of representation accuracy while achieving significant speedup in dictionary reconstruction and image encoding times. Our method not only helps in choosing smaller and more representative dictionaries, but also enables learning of more incoherent ones.

Index Terms— Online Dictionary Learning, Clustering

1. INTRODUCTION

Dictionary learning (DL) based sparse coding has emerged as the state of the art in many low level image processing tasks such as denoising, inpainting, and demosaicking [3, 6]. It offers more robustness and data dependent representations compared to standard sparsifying transformations like DCT and wavelets.

DL aims to represent data as a linear combination of learned bases, which can be posed as the following optimization problem:

$$\{\hat{D}, \hat{A}\} = \underset{D,A}{\operatorname{argmin}} \ \frac{1}{2}\|X - DA\|_F^2 + \lambda\|A\|_p \qquad (1)$$

where X is a matrix with data points as columns, $\hat{D}$ is a dictionary to be learned , and $\hat{A}$ is the set of coefficients such that $X \approx \hat{D}\hat{A}$. λ is a regularization parameter. $0 \leq p \leq 1$, with $\|A\|_p$ defined as:

$$\|A\|_p = \left(\sum_i \sum_j |A_{ij}|^p \right)^{\frac{1}{p}}$$

The penalty term promotes sparsity in the coefficients $\hat{A}$. We consider $p = 1$, making the regularization term, and subsequently the entire equation (1) convex.

The function to be minimised in equation (1) is not jointly convex in A and D, but it becomes convex in one variable keeping the other fixed. Typical dictionary learning algorithms (e.g. KSVD [1], online [6]) consist of alternating between a sparse coding stage and a dictionary update stage. The authors in [6] show that the online dictionary learning procedure has a computational load of $\mathcal{O}(k^2m + km + ks^2) \approx \mathcal{O}(k^2)$, where k is the dictionary size (the number of atoms in the dictionary) and m and s are the dimension of data and sparsity of the coefficients (in the sparse coding stage) respectively. Moreover, in [9] the sample complexity of dictionary learning is derived to be $\mathcal{O}(\sqrt{k})$. So, the size of the dictionary has an impact on both the computational and sample complexity of the algorithm.

Typically, the size of the dictionary is fixed before learning. A trade-off has to be made between choosing a larger, slow learning, but better fitting dictionary and a fast learning, smaller dictionary. This motivates our problem: Can we efficiently learn a dictionary that is "optimal" in size, so that on the one hand it provides good representation of the data, and on the other it is not too large so as to burden computational resources? Moreover, can the small dictionary that is learned also provide a very good fit to the data?

Past work has addressed this problem in an offline setting [7], learning from a predefined superset [5], discarding atoms that vanish [11] or adding a new regularization term that promotes smaller dictionaries [12]. We propose a dictionary size adaptation method that learns a smaller dictionary and performs comparably to the state-of-the-art. Our work differs from the above in several aspects. Firstly, we assume an online setting, where we believe the need for gains in computational time and memory requirements is most needed. Secondly, we do not prune the dictionary by discarding atoms [7], but use a density based approach to synthesize new atoms from several atoms "close" to each other. We show in our simulations that the loss in redundancy arising from the clustering of atoms does not affect coding accuracy. Thirdly, we do not make restrictive assumptions on the dictionary or the data itself, except that the atoms in the dictionary lie on the unit sphere, a valid assumption so as to prevent the reconstruction coefficients from arbitrarily scaling.

Our method has the effect of preventing atoms of the dictionary from clumping together. This has another advantage: It is argued in [8] that incoherent dictionaries perform better in terms of image representation than coherent ones, by preventing overfitting to the data. Incoherence of the dictionary atoms also plays a role in determining the performance of the sparse coding algorithms. Since incoherence depends on the separation (and the ensuing dissimilarity) between the atoms, merging nearby dictionary elements into a single atom promotes incoherence.

Online dictionary learning [6] has been proven to outperform standard batch learning methods in terms of speed [1]. We provide a framework that further speeds up the online learning method, and hence our method is inherently several orders of magnitude faster than batch processing methods.

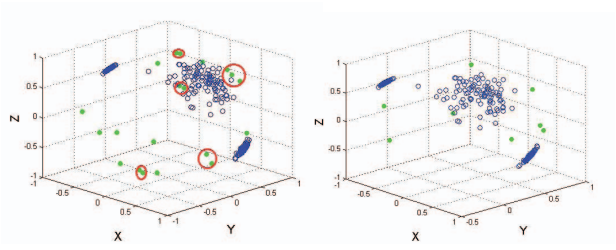
1.1. Notation

Let $x^i \in \mathbb{R}^n$ be the i^{th} sample of training data $X \in \mathbb{R}^{n \times p}$. The dictionary is denoted by D. We suppose we start with an initial dictionary of size k_0, so that $D_0 \in \mathbb{R}^{n \times k_0}$. We assume we use a kernel $K(\cdot)$ with an associated bandwidth h for the dictionary resizing step. After resizing, the dictionary will be denoted by $\overline{D}$. Subscripted variables x_t, D_t indicate the corresponding variables at the t^{th} iteration. Superscripts d^i, a^i indicate columns of the corresponding matrices.

[⋆]This work was performed at Mitsubishi Electric Research Labs.

2. OUR ALGORITHM

Our algorithm, Clustering based Online Learning of Dictionaries (COLD), employs the mean shift clustering procedure [2] to discover modes in the distribution of the atoms. Being nonparametric, it precludes the need to know the number of clusters in advance. It is important to note here that the clustering is done on the (empirical) distribution of the dictionary atoms, and not on the data. We outline our algorithm in Algorithm 1 after providing an intuitive explanation as to how the scheme works and why it is beneficial.



(a) (b)

Fig. 1. Fig. 1(a) shows the distribution of the data (blue circles) and learned dictionary of size 20 (green) for the data. Note how some atoms are very close to each other, depicted by red circles. Fig. 1(b): learned dictionary of size 9 for the data. Now, the atoms are further spaced apart (best seen in color).

Consider data X distributed as in Fig. 1(a). We start with an initial dictionary of size $k0$ distributed randomly over the unit sphere. We see that, after training, the atoms of the dictionary align themselves according to the data as in Fig. 1(a). After alignment, one can see that some atoms are clumped together in pairs or triplets.

When two or more atoms are very close to each other, it is highly unlikely that more than one of them will be used to represent a data point simultaneously due to the sparsity constraint on the representation coefficients. Referring to Fig. 1(a), only one atom per "clump" of atoms will be used per data point for representation.Hence, whenever dictionary atoms get "too close", we can merge them, leading to a situation more akin to that seen in Fig. 1(b).

Since the setting is online, we receive data points in a sequential manner. In the sparse coding step, we have a single data point x_t and we obtain the corresponding coefficients α^t

$$\alpha_t = \underset{\alpha}{\operatorname{argmin}} \ \frac{1}{2}\|x_t - D\alpha\|^2 \ + \ \lambda\|\alpha\|_1$$

Then, to update the dictionary for known $\alpha_t's$, one needs to obtain the solution for

$$D_t = \underset{D}{\operatorname{argmin}} \ \frac{1}{t}\sum_{i=1}^{t}\left[\frac{1}{2}\|x_i - D\alpha_i\|^2 \ + \ \lambda\|\alpha_i\|_1\right].$$

In the dictionary update stage (refer Algorithm 1), we restrict the atoms to lie *on* the unit ball unlike [6] so as to make the clustering easier.

$$d_t^j = \frac{u^j}{\|u^j\|} \tag{2}$$

Note here that we restrict the dictionary atoms to lie *on* the unit Eucledian ball, and not *in* it. This makes the clustering easier, by ensuring the vectors are normalized.

We do not apply the mean-shift until $minIters$ iterations has passed. This is because, in the online case, we need to wait until the dictionary learning procedure has adapted itself to a sufficient amount of data, before we start modifying it. One can think of a degerate case where , after the first iteration, all the dictionary atoms are perfectly aligned, so that all atoms are in the same cluster, leading to a dictionary of size 1 after clustering. To prevent this, we wait for $minIters$ iterations. In most cases, waiting for $k0$ iterations before running the mean-shift procedure constitutes a sufficient waiting time.

After clustering, if the dictionary is shrunk, *i.e.* $\overline{D_t} \neq D_t$, matrices A and B are reset. This is because the "merged" atoms give rise to a new dictionary, and it makes sense to treat the procedure as if restarting the learning algorithm with the new dictionary as an initialization. This can be seen as analogous to periodically flushing the history in the usual online learning scenario. The procedure can be improved by not discarding history corresponding to the atoms in the dictionary that are retained, but a search will require more computations, possibly obliviating the complexity gains acquired by clustering.

Algorithm 1 Algorithm: COLD

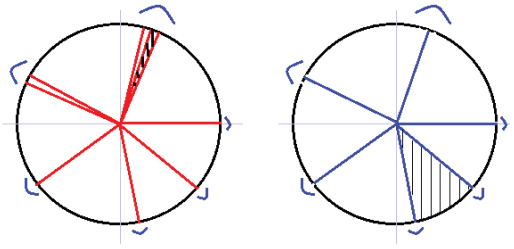
Inputs : $x \in \mathbb{R}^n \overset{iid}{\sim} p(x)$, λ, $D_0 \in \mathbb{R}^{n\times k_0}$, $maxIters$, $minIters$, h
Initialize : $A \in \mathbb{R}^{k_0\times k_0} \leftarrow 0$, $B \in \mathbb{R}^{m\times k_0} \leftarrow D_0$, $t = 1$,$\overline{D_0} \leftarrow D_0$
while $t \leq maxIters$ **do**
  Draw $x_t \sim p(x)$
  $\alpha_t = \underset{\alpha}{\operatorname{argmin}} \left[\frac{1}{2}\|x_t - D_{t-1}\alpha\|^2 + \lambda\|\alpha\|_1\right]$
  $A \leftarrow A + \alpha_t\alpha_t^T$
  $B \leftarrow B + x_t\alpha_t^T$
  if $t \geq minIters$ **then**
    $\overline{D_{t-1}} = MeanShift(D_{t-1}, h)$
    if $\overline{D_{t-1}} \neq D_{t-1}$ **then**
      $A \leftarrow 0$, $B \leftarrow \overline{D_{t-1}}$ {resetting past information}
      $h = h/(t - minIters + 1)$ {for proving convergence, can be omitted}
    end if
  end if
  Compute D_t using Algorithm 2 of [6], with $\overline{D_{t-1}}$ as warm restart
  $t \leftarrow t + 1$
end while
Output : D_t

3. COHERENCE, COMPLEXITY, CONVERGENCE

COLD learns a more incoherent dictionary, achieves speedups in computational complexity and also converges to a local minimum almost surely. Due to space constraints, we defer detailed proofs and analysis exist for a longer version of our work.

Fig. 2 pictorially shows how the coherence is reduced as nearby atoms are clustered and replaced by a single atom. For representation purposes, atoms are assumed to lie on the unit circle in two dimensions

Smaller dictionaries have lower sample complexity. The amount of reduction in computational complexity depends on the version of mean-shift clustering used. Traditional mean shift has complexity superlinear in the number of data points [4], generally $\mathcal{O}(k^2)$. We consider here this setting and show that even this achieves a reduction in complexity. So by default, faster mean-shift algorithms (e.g. [10]) will perform much better.



(a) D (b) $\bar{D}$

Fig. 2. Initial dictionary on the unit disc (Fig. 2(a)). The small shaded area corresponds to the angle between the atoms, which decides the initial coherence $\mu(D)$. The bumps outside the disc correspond to the modes of the kernel density estimate over the atoms. Fig. 2(b) shows the atoms after clustering, and the corresponding shaded region indicates the angle determining the new coherence. Clearly $\mu(\bar{D}) < \mu(D)$

It is natural that *every* mean-shift will not result in a reduction of dictionary size. Suppose M of them do, so that for every m_j iterations, $j = 0, 1, \cdots M$, the dictionary size reduces sequentially from k_0 to k_j. Of course, $\sum_{j=0}^{M} m_j = n$, where n is the number of iterations. Considering the mean-shift itself to have a (maximum) complexity of $\mathcal{O}(k_j^2)$, we have the total complexity of COLD to be less than ODL (the method in [6]) provided that we have

$$2\sum_{j=0}^{M} m_j k_j^2 \leq n k_0^2 \tag{3}$$

This inequality will strictly hold as long as m_j is large and $k_j << k_0$ for $j \approx M$. That this holds in most cases has been supported by experimental validation, as we will see in section 4.

The proof of almost sure convergence follows closely along the lines of that in [6]. We omit it here due to space constraints. The key idea is that we can show that as $h \to 0$, mean shift clustering stops modifying the dictionary, and so the proof in [6] can be applied. A similar situation holds for constant h, though the analysis is harder.

4. EXPERIMENTS AND RESULTS

We compare ODL and COLD in terms of speed. Speed is measured in terms of the number of seconds it takes for the algorithm to both learn as well as perform sparse coding after the dictionary is learnt. Table 1 compares the performance of the methods, as the initial dictionary size is varied. The test images were taken from the "background" dataset at Caltech [1]. The dataset contains 450 grayscale images of various scenes. The bandwidth for the mean shift algorithm was set at 0.5. The images were broken down into 5×5 overlapping patches, which yields the size of each dictionary atom to be 25.

Note that, the increase in speed does not compromise the accuracy of encoding. Another thing is that, regardless of the initial number of dictionary atoms we start with, COLD returns nearly the same dictionary size all the time ($\approx 2\times$ overcomplete). This underscores the underlying idea of looking to find an "optimum" dictionary size, which is data dependent. We see that for a small dictionary, COLD is slower than ODL, underscoring the need for the condition in equation (3) to hold. It also suggests that if the "correct" dictionary size is already known, then one could initialize the initial dictionary of this size, and clustering of the atoms will not further reduce the size of the dictionary significantly.

[1] http://www.vision.caltech.edu/html-files/archive.html

Dsize	method	time(sec)	MSE(std.dev)
50	ODL	11.855	0.00474(0.0030)
41	COLD	17.748	0.00475(0.0029)
100	ODL	20.902	0.00465(0.0030)
40	COLD	18.201	0.00471(0.0029)
150	ODL	34.913	0.00462(0.0030)
50	COLD	23.445	0.00473(0.0029)
200	ODL	49.840	0.00461(0.0029)
45	COLD	24.148	0.00472(0.0029)

Table 1. Comparison of ODL and COLD on the "background" dataset. The first column indicates the final dictionary size after learning. Note that in case of ODL, final size = initial size. We can see that, as the initial dictionary size increases, COLD is much faster, while the loss in MSE is negligible

In single images, even after a single iteration, the dictionary size reduces and we obtain a significant speedup in the algorithm. For the plot in Fig 3, we used 6000 patches for training, merely to display the speedup of the algorithm. The remainder of the patches constituted the test set. We see that ODL takes $4\times$ in training than COLD. The "jump" observed in the plot is due to the fact that the dictionary size *drastically* reduces when the mean-shift is applied, and the entire learning process has to be restarted.

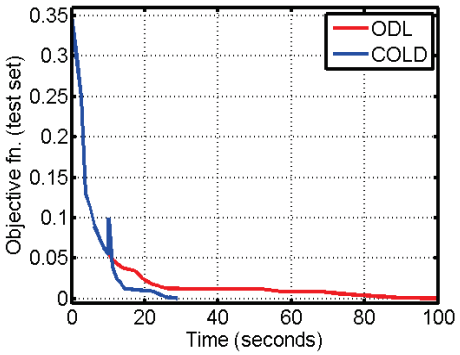


Fig. 3. Speedup obtained on 6000 patches dataset (best seen in color)

ODL		COLD	
Training	Denoising	Training	Denoising
260	105	98	24

Table 2. Comparison of ODL and COLD denoising times (Fig. 4.

Fig. 4 compares COLD and ODL in a denoising setting. We used 10000 patches of size 8×8 to train the dictionary. We then used the learned dictionary to denoise the entire image (a total of 255K 8×8patches). We see that COLD significantly speeds up the learning and reconstruction process (See Table 2).

Figures 5 and 6 compare the ability of COLD and ODL to learn dictionaries from a small fraction of the data. We start with an original image of size 512×512, extract 8×8 overlapping patches from it. Of these, we consider only 10000 patches selected randomly for learning a dictionary. We then use the learned dictionary to reconstruct the entire image. The initial dictionary was sized 64×256 in all cases. It can be seen that COLD gives nearly identical results

Fig. 4. Image denoising samples. COLD only needs less than 20% of the computations required by ODL, as seen in Table 2.

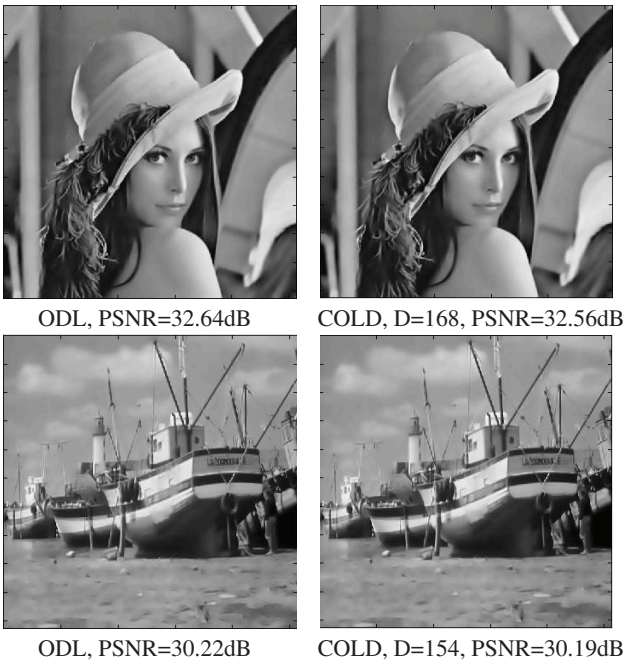


Fig. 5. Images on the left are the reconstruction obtained from ODL and the one on the right is the reconstruction using COLD. D denotes the dictionary size.

to ODL. "Training" refers to learning the dictionary, and "encoding" refers to creating the sparse coefficients once the dictionary is learned.

5. CONCLUSIONS

We introduced COLD, a method to achieve speedups in online dictionary learning by employing clustering of dictionary atoms. We show that one can achieve a considerable speedup in computation times when there is an inherent clustering step in the dictionary learning framework. The method also aids in learning more incoherent dictionaries.

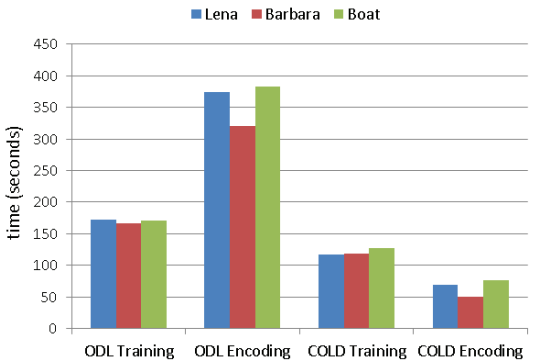


Fig. 6. Reconstruction times taken by ODL and COLD.

6. REFERENCES

[1] M. Aharon, M. Elad, and A. Bruckstein. The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Processing*, 54(11), 2006.

[2] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(5), 2002.

[3] M. Elad and M. Aharon. Image denoising via learned dictionaries and sparse representation. *IEEE Computer Vision and Pattern Recognition*, 2006.

[4] D. Freedman and P. Kisilev. Fast mean shift by compact density representation. *CVPR*, 2009.

[5] A. Krause and V. Cevher. Submodular dictionary selection for sparse representation. *Internatinal Conference on Machine Learning (ICML)*, 2010.

[6] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11, 2010.

[7] R. Mazhar and P. Gader. EK-SVD: Optimized dictionary design for sparse representations. *International Conference on Pattern Recognition*, 2008.

[8] I. Ramirez, F. Lecumberry, and G. Sapiro. Sparse modeling with universal priors and learned incoherent dictionaries. *IMA preprint Series 2279*, 2009.

[9] D. Vainsencher, S. Mannor, and A. Bruckstein. The sample complexity of dictionary learning. *COLT*, 2011.

[10] P. Wang, D. Lee, A. Gray, and J. Rehg. Fast mean shift with accurate and stable convergence. *AISTATS*, 2007.

[11] M. Yaghoobi, T. Blumensath, and M. Davies. Regularized dictionary learning for sparse approximation. *EUSIPCO*, 2008.

[12] M. Yaghoobi, T. Blumensath, and M. Davies. Dictionary learning for sparse approximations with the majorization method. *IEEE Trans. Signal Processing*, 57, 2009.