Sparse Coding for Dictionary Learning in Context of Image De-noising

Dhaivat Deepak Shah
ds3267@columbia.edu

Gaurav Ahuja
ga2371@columbia.edu

Sarah Panda
sp3206@columbia.edu

Abstract

Sparse Coding is the modeling of data vectors as linear combination of basis elements or atoms which form the Dictionary. In this paper we study the behaviour of various sparse coding algorithms when used in K-SVD [1] for Dictionary learning. Dictionary learning technique has been used to denoise an image by learning a dictionary from the noisy image. We learn dictionaries of various sizes for images with different levels of noise and compare the performance of various sparse coding algorithms. This comparison is done in-terms of time per iteration and the SNR of the denoised image resulting from the dictionary learnt using these algorithms.

1 Introduction

Image restoration is a widely studied problem in image processing and the common approaches to solve the problem include filtering, minimizing variations in an image etc. The problem we address in this paper is to remove additive Gaussian noise from a given image using Dictionary Learning approach. The approach bases itself on representing an image as a sparse representation over a good dictionary. So our problem of restoring images boils down to learning a good dictionary and finding a good representation of any given patch of an image using the dictionary. Designing dictionaries to improve the above model can be done by either selecting one from a pre-specified set of linear transforms, or by adapting the dictionary to a set of training signals. Past research [1] has shown that dictionaries adapted to training signals lead to a lot better signal restoration than the pre-specified dictionaries. The training signal could either be a dataset of related images or the noisy image itself. In our case strive to learn a dictionary from the input noisy image for the purpose of restoration. We have used the K-SVD algorithm as our basis for dictionary learning and explore different sparse coding methods that can be used for the same. The next few sections give a review of the different components of Dictionary Learning and discuss sparse coding algorithms. Section 6 talks about our experimental setting. We present the results of our experiments in section 7 and conclude with section 8.

2 Intro to Dictionary learning

Dictionary Learning is a focus area in the Signal Processing domain. A dictionary is used for the sparse representation or approximation of signals. A dictionary D, is an NxK matrix that contains K prototype signals called atoms. A signal y_0 can be represented by a linear combination of a small number of these atoms. For any signal y_0 and dictionary D there must exist a sparse vector $x \in R^K$ such that:

$$y_0 = Dx \tag{1}$$

There will be cases where we cannot exactly represent the signal with a linear combinations of atoms from a finite dictionary. In such a case we find an approximation of the signal, y such that:

$$y = Dx \tag{2}$$

$$\|y - y_0\| < \epsilon \tag{3}$$

Dictionary Learning is a problem of finding a dictionary such that the approximation of many vectors in the training set are as good as possible.

For a finite number of training vectors, L each of length N the aim of Dictionary Learning is to find both a Dictionary D, of size NxK and the L corresponding coefficients vectors of length K.

2.1 Problem Statement

Classical dictionary learning techniques consider a finite training set of signals $Y = [y_1, y_2, \dots, y_L]$ int $R^{N\text{x}L}$ and optimize the empirical cost function:

$$f(D) = \frac{1}{L}\sum_{i=1}^{L} l(y_i, D) \tag{4}$$

where $D \in R^{N\text{x}K}$ is the dictionary and l is the loss function such that $l(y, D)$ is small if D is good at representing the signal x.

$l(y, D)$ can be defined as the optimal value of the l_1 or l_0 sparse coding problem. See section on Sparse coding for more details.

We consider the l_1 norm for illustration in this section.

$$l(y, D) = \min_{x \in R^K} \frac{1}{2}\|y - Dx\|^2 + \lambda\|x\|_1 \tag{5}$$

λ is the regularization parameter. The problem of minimizing the empirical cost function $f(D)$ is not convex with respect to D. It can be rewritten as a joint optimization problem with respect to dictionary D and the coefficients $x = [x_1, x_2, \dots, x_L]$, which is not jointly convex, but convex with respect to each of the two variables D and x when the other one is fixed.

$$\min_{D \in R^{NxK}, x \in R^{KxL}} \frac{1}{L}\sum_{i=1}^{L}\left(\frac{1}{2}\|y_i - Dx_i\|^2 + \lambda\|x_i\|_1\right) \tag{6}$$

A natural approach to solving this problem is to alternate between the two variables, minimizing one over the other while keeping the other fixed. There are two popular approaches for this

1. **K-SVD**: It is an iterative method that alternates between sparse coding of the examples based on the current dictionary and a process of updating the dictionary atoms to better fit the data. In each iteration of K-SVD all the training vectors are used. [1, 13]
2. **Online Dictionary Learning**: This dictionary learning method falls into the class of online algorithms based on stochastic approximations, processing one example at a time. [9]

We have used only K-SVD for the purpose of our paper.

3 K-SVD for dictionary learning

The goal of K-SVD is to adapt a dictionary to the training signals such that it ensures the best sparse representation for each of the signals, under constraints of reconstruction error and sparsity. It is a two-step iterative process, which begins with a dictionary initialization. The dictionary could be initialized with any pre-calculated well known dictionary, like DCT, or created from the noisy image by randomly selecting atoms from the patches of the given image. The first step of the process determines the sparse code for the given training signals with fixed dictionary. The second step assumed fixed sparse code and updates the dictionary to better fit the data. The second step further updates the sparse representation of the training signals to accelerate convergence.

K-SVD tries to optimize the following overall MSE under sparsity constraints in each iteration:

$$\min_{d,x} \|Y - DX\|_F{}^2 \text{ subj. to, } \|x_i\|_0 \leq S \forall i \tag{7}$$

The algorithm proposed in [1] is outlined below:

```
1. **Initialization:** Set dictionary matrix $D \in R^{N\text{x}K}$ with $l_2$ normalized columns
2. **while** not converged($j$ = 1,2,...) **do**
3.     **Sparse Coding:**
4.         Compute $x_i$ for every training signal $y_i$, by solving for
           $\min_{x_i} \|y_i - Dx_i\|_F{}^2$ subject to $\|x_k\|_0 \leq S$
5.     **Dictionary Update:**
6.         Update each atom k in 1,2,..K in $D_{j-1}$ as below:
   (a)         Let $w_k$ be the set of input signals that use this atom D(k).
   (b)         Find the overall error in representation, in matrix $E_k$,
               $E_k = Y - \sum(j \neq k) d_j x_j{}^2$
   (c)         Restrict $E_k$ by choosing columns corresponding to $w_k$ as $E_k^R$
   (d)         Apply SVD decomposition on the restricted error matrix :
              $E_k^R = U\delta V^T$
              Set $d_k = U[:1]$
              and $x_r^k = V[:1] * \delta(1,1)$
7.         j= j+1
8. **end while**
9. **Output:** $X^* \leftarrow X_j, D^* \leftarrow D_j$
```

In this paper, we focus on the first step of K-SVD, that is the sparse coding step. The following sections discuss the problem of sparse coding in detail along with the algorithms we have used.

4 Sparse Coding

As mentioned before, any signal y can be represented as a linear combination of atoms in a dictionary D. Sparse coding problem aims at finding the sparsest such representation. On a high level, those atoms summarize the most important features in the input.

$$\min \|x\|_0 \text{ subj. to } y = Dx \tag{8}$$

The above equation tries to find the sparsest coefficient vector that exactly reconstructs the input signal. However, this form of exact determination is NP-hard. So, to make it easier to solve, the problem needs to be relaxed by allowing for a small amount of reconstruction error, ϵ. The problem can be thus formulated as,

$$\min \|x\|_0 \text{ subj. to } \|y - Dx\|^2 \leq \epsilon \tag{9}$$

Still the problem is computationally expensive and following are two of the well known approaches to approximate the solution to the above.

4.1 Greedy (Matching pursuit)

Greedy methodology, solves the problem in equation (9) by sequentially selecting the atom that best explains the data. The coefficient vector(sparse code) is then updated to include the selected atom and its contribution to the signal is deducted from the residual. Upon convergence, the coefficient vector assumes a sparse code for the input signal over the given dictionary. We cover two important Greedy algorithms, Matching Pursuit and Orthogonal Matching pursuit in the section below and also, study their performance in K-SVD de-noising.

4.2 Relaxation (Basis pursuit)

In the relaxation methodology, the l_0 norm in equation (9) is replaced with the l_1 norm, resulting in the following equivalent objective,

$$\min \|y - Dx\|^2 + \lambda \|x\|_1 \tag{10}$$

l_1 regularization penalizes the non-sparse terms based on their actual values, which may seem unfair, but past research [16] has shown that it leads to a very good approximation and often leads to the sparsest representation. The resulting optimization problem is convex and unconstrained, and hence can be solved using several methods like gradient descent.

5 Summary of sparse coding techniques used:

5.1 Matching Pursuit (MP)

Matching pursuit, proposed by [10] is a well known greedy algorithm for learning sparse code of a signal over a given dictionary. The basic idea of the algorithm is outlined below.
At every step of refining the sparse code, the algorithm picks an atom of the dictionary that best represents a portion of the input signal.

The following pseudo-code briefly describes the algorithm:

```
1. Input: Training signal y, Dictionary - D
2. Initialization: $R_1 = y, x = 0, n = 1$
3. while $R_n >$ Threshold do
4.     Find $d_i \in$ D with max correlation with residual i.e. $< R_n, d_i >$
5.     $x_i = x_i + < R, d_i >$
6.     $R_{n+1} = R_n - x_i d_i$
7.     $n = n + 1$
8. end while
9. Output: Sparse code of y over D, $x^* \leftarrow x_n$
```

The algorithm evaluates the best matching atom at every step by evaluating the inner product between the Residual, R_n and the individual atoms from the dictionary. The coefficient vector x is updated with the atom that has maximum inner product and its contribution to the signal is deducted from the signal. The process repeats until the residual falls below a threshold. The sparse coefficients thus learned is believed to be the sparsest approximation of the decomposition of the signal over the dictionary.

5.2 Orthogonal Matching Pursuit (OMP)

A variation of the Matching Pursuit is the Orthogonal Matching Pursuit [11], also a greedy method that iteratively selects the atom having highest correlation with the residual. The difference is that after adding the best matched atom to the coefficient vector, the residual is updated with the orthogonal projection of the input signal y onto the subspace of previously selected atoms. This set of selected atoms is called the active set and since the resulting residual is orthogonal to each of the atoms in the set, none of those atoms are selected in the future iterations.

The OMP algorithm is as below:

```
1. Input: Training signal y, Dictionary - D
```

```
2. Initialization: $R_0 = y, X(c_0) = 0, n = 0, c_0 = \{\}$
3. while not converged do
4.     Solve for $X_{t_i}$ by maximizing,
       $\max_t {X_t}^T R_{n-1}$
       Add $X_{t_i}$ to the set of selected atoms; $c_n = c_{n-1} \cup t_i$
5.     Let $P_n = X(c_n)(X(c_n)^T X(c_n))^{-1} X(c_n)^T$ denote the projection onto the linear space
   spanned by the elements of $X(c_n)$.
6.     Update $R_n = (I - P_n)y$.
7.     $n = n + 1$
8. end while
9. Output: Sparse code of y over D, $x^* \leftarrow x_n$
```

5.3 Fast Iterative Shrinkage Thresholding Algorithm (FISTA)

The objective function that we have as defined above is:

$$F(x) = \frac{1}{2}\|y - Dx\|^2 + \lambda\|x\|_1 \tag{11}$$

Here the first term, $f(x)$ is a smooth, convex function with Lipschitz continuous gradient $D^T(Dx - y)$ and $L_f = \|D^T D\|$.

FISTA, proposed by [2] has a faster convergence rate as compared to ISTA. The main difference between the two is that the iterative shrinkage operator is not applied to the previous point alone but to another point which uses a specific linear combination of the previous two points. The algorithm in this case becomes:

```
1. Input: $L_f$
2. Step 0. $j_1 = x_0 \,\epsilon\, \mathbb{R}^n, t_1 = 1$
3. Step $k$.($k \geq 1$)
4.     $x_k = soft(j_k - \frac{1}{L_f}\nabla f(j_k), \frac{\lambda}{L_f})$
5.     $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
6.     $y_{k+1} = x_k + \frac{t_k - 1}{t_k + 1}(x_k - x_{k-1})$
```

5.4 Augmented Lagrangian Method (ALM)

The Augmented Lagrangian Method was proposed individually [12] and [6] and is explained in depth by [15]. It is an extension to the quadratic penalty function method proposed by [4]. In the penalty function method, we add a quadratic penalty term for each of the constraints in a constrained optimisation problem. That is for:

$$\min_x f(x) \text{ subj. to } c(x) = 0 \tag{12}$$

the formulation becomes,

$$\min_x f(x) + \frac{\mu}{2}\|c(x)\|^2 \tag{13}$$

where μ is the penalty parameter and is always positive. We can now progressively increase μ towards ∞ and try to find a minimizer for the objective function. Thus we can convert the constrained minimisation problem into an unconstrained one. However, an ill-conditioning for the Hessian in the formulation can lead to significantly poor results for the iterative methods. To reduce this possibility, we include a Lagrange multiplier to get the Augmented Lagrangian as:

$$\min_x f(x) + \frac{\mu}{2}\|c(x)\|^2 + \lambda c(x) \tag{14}$$

We look at both the primal and the dual formulations of this approach.

5.4.1 Primal ALM (PALM)

In our case, the problem boils down to:

$$L_\mu(x, e, \lambda) = \|x\|_1 + \|e\|_1 + \frac{\mu}{2}\|y - Dx - e\|^2 + \lambda(y - Dx - e) \tag{15}$$

For the primal problem, [3] showed that there exists a λ^* and μ^* such that

$$e_{k+1} = arg\min_e L_\mu(x_k, e, \lambda_k) \tag{16}$$

$$x_{k+1} = arg\min_x L_\mu(x, e_{k+1}, \lambda_k) \tag{17}$$

$$\lambda_{k+1} = \lambda_k + \mu(y - Dx_{k+1} - e_{k+1}) \tag{18}$$

Of the above equations, the one for e has a closed form solution. As for the update of x, it is a standard L_1 minimisation problem which we solve using the FISTA method explained above.

So the overall algorithm can be summarized as in [16]:

```
1. **Input:**$y \,\epsilon\, \mathbb{R}^m$, $D \,\epsilon\, \mathbb{R}^{mxn}$, $x_1 = 0, e_1 = y, \lambda_1 = 0$
2. **while** not converged($k$ = 1,2,...) **do**
3.     $e_{k+1} \leftarrow shrink(y - Dx_k + \frac{1}{\mu}\lambda_k, \frac{1}{\mu})$
4.     $t_1 \leftarrow 1, z_1 \leftarrow x_k, w_1 \leftarrow x_k$
5.     **while** not converged ($l$ = 1,2,...) **do**
6.         $w_{l+1} \leftarrow shrink(z_l + \frac{1}{l}D^T(y - Dz_1 - e_{k+1} + \frac{1}{\mu}\lambda_k), \frac{1}{\mu L})$
7.         $t_{l+1} \leftarrow \frac{1}{2}(1 + \sqrt{1 + 4t_l^2})$
8.         $z_{l+1} \leftarrow w_{l+1} + \frac{t_l - 1}{t_l + 1}(w_{l+1} - w_l)$
9.     **end while**
10.    $x_{k+1} \leftarrow w_l, \lambda_{k+1} \leftarrow \lambda_k + \mu(y - Dx_{k+1} - e_{k+1})$
11. **end while**
12. **Output:** $x^* \leftarrow x_k, e^* \leftarrow e_k$
```

While in the general case of Augmented Lagrangian methods, the value of μ is incremented after every iteration, we are holding it fixed to the initialisation value.

5.4.2 Dual ALM (DALM)

The dual Augmented Lagrangian method for efficient sparse reconstruction was proposed by [14]. It tries to solve the dual of the problem we have been tackling so far as:

$$\max_j y^T j \text{ subj. to } D^T j \,\epsilon\, \mathbb{B}_1^\infty \tag{19}$$

where $\mathbb{B}_1^\infty = \{x \,\epsilon\, \mathbb{R}^n : \|x\|_\infty \leq 1\}$

The associated Lagrangian function becomes:

$$\min_{j,z} -y^T j - \lambda^T(z - D^T j) + \frac{\mu}{2}\|z - D^T j\|^2 \text{ subj. to } z\epsilon\mathbb{B}_1^\infty \tag{20}$$

Here, again there is a simultaneous minimization w.r.t. j,λ and z. So we adopt an alternation strategy to get the following algorithm as in [16]:

```
1. **Input:** $y \,\epsilon\, \mathbb{R}^>, B = [A, I] \,\epsilon\, \mathbb{R}^{mx(n+m)}, w_1 = 0, j_1 = 0$
2. **while** not converged ($k$ = 1,2,...) **do**
3.     $z_{k+1} = \mathbb{P}_{\mathbb{B}_1^\infty}(B^T j_k + \frac{w_k}{\mu})$
```

```
4.     $j_{k+1} = (BB^T)^{-1}(Dz_{k+1} - (Bw_k - y)/\mu)$
5.     $w_{k+1} = w_k - \mu(z_{k+1} - D^T j_{k+1})$
6. **end while**
7. **Output:** $\lambda^* \leftarrow w_k[1:n], e^* \leftarrow w_k[n+1:n+m], j^* \leftarrow j_k$
```

5.5 Feature Sign

Feature sign [8] aims to solve the following l_1 regularized sparse coding problem

$$\min_x f(x) = \|y - Dx\|^2 + \lambda\|x\|_1 \tag{21}$$

The intuition behind this algorithm is that if we know the signs(positive, negative or zero) of x_i at the optimal value we can replace $\|x_i\|$ by either x_i if $x_i > 0$, $-x_i$ if $x_i < 0$ or 0 if $x_i = 0$. Considering only non zero coefficients this problem reduces to a standard quadratic optimization problem. This problem can them be solved analytically and efficiently. Feature sign algorithm tries to search for the signs of the coefficients. Given any such search results, it efficiently solves the resulting unconstrained quadratic problem. Further the algorithm systematically refines the guess if it turns out to be initially incorrect.

Following is an outline of the Feature Sign algorithm as proposed in [8]

1. **Initialize** $x := 0, \theta : 0$ and active set $:= \{\}$, $\theta_i \in \{-1, 0, 1\}$ denotes sign(x_i)
2. From zero coefficients of x, select $i = \text{argmax}_i \, |\frac{\partial\|y-Dx\|^2}{\partial x_i}|$
   Activate x_i (add i to active set) only if it locally improves the objective
   (a) If $|\frac{\partial\|y-Dx\|^2}{\partial x_i}| > \lambda$; set $\theta_i := -1$, active set $:= \{i\} \cup$active set
   (b) If $|\frac{\partial\|y-Dx\|^2}{\partial x_i}| < -\lambda$; set $\theta_i := 1$, active set $:= \{i\} \cup$active set
3. **Feature-Sign step:**
   Let $\hat{D}$ be a submatrix of D that contains only the columns corresponding to the active set.
   Let $\hat{x}$ and $\hat{\theta}$ be sub vectors of x and θ corresponding to the active set.
   Compute the analytical solution to the resulting quadratic problem
   $\hat{x}_{new} := (\hat{D}^T\hat{D})^{-1}(\hat{D}^T y - \lambda\hat{\theta}/2)$
   Perform a discrete line search on the closed line segment from $\hat{x}$ to $\hat{x}_{new}$:
   - Check the objective value at $\hat{x}_{new}$ and all points where any coefficient changes sign.
   - Update $\hat{x}$ (and all corresponding entries in x) to the point with the lowest objective value
   Remove zero coefficients of $\hat{x}$ from the active set and update $\theta :=$sign(x)
4. **Check the optimality conditions**:
   (a) Optimality condition for nonzero coefficients: $|\frac{\partial\|y-Dx\|^2}{\partial x_i}| + \lambda\text{sign}(x_i) = 0 \; \forall x_i \neq 0$
   If this condition is not satisfied, go to Step 3(without any new activation); else check condition (b).
   (b) Optimality condition for zero coefficients: : $|\frac{\partial\|y-Dx\|^2}{\partial x_i}| + \lambda\text{sign}(x_i) \leq 0 \; \forall x_i = 0$
   If this condition is not satisfied go to Step 3; otherwise return x as the solution.

5.6 Truncated Newton Interior Point Method (TNIPM, L1LS)

In [7], the authors formulate a Truncated Newton Interior Point method. The algorithm solves the l_1 regularised problem:

$$\min_x f(x) = \|y - Dx\|^2 + \lambda\|x\|_1 \tag{22}$$

This problem can be transformed to a convex quadratic program with linear inequality constraints,

$$\min \|y - Dx\|^2 + \sum_{i=1}^{K} \lambda u_i \tag{23}$$

$$\text{subject to: } -u_i \leq x_i \leq u_i, \forall i \in \{i, K\} \tag{24}$$

The aim of this algorithm is to solve this quadratic program using the Newton's method. Since solving the Newton system exactly is computationally expensive for large l_1 problems, TNIPM uses an iterative method, Preconditioned Conjugate Gradient (PCG) to approximately solve the Newton system and develops an interior point method to solve the l_1 problem. This method is referred as L1LS in this paper.

6 Experimental Setup

We have used the K-SVD approach suggested in the previous sections for Dictionary Learning, and have used above mentioned 7 solvers for benchmarking. The dictionary has been initialised with a DCT dictionary in each of the cases.

We break each noisy image into 4096 training vectors of size 64. Each vector is a 8x8 patch from the image.

8 iterations of K-SVD were performed. For each of the solvers, the following parameters have been varied:

1. Starting Image Noise Levels (σ) 10, 20, 50 %
2. Dictionary Size: 64, 128, 256

And the comparison is based on:

1. Execution Time
2. SNR of the de-noised image

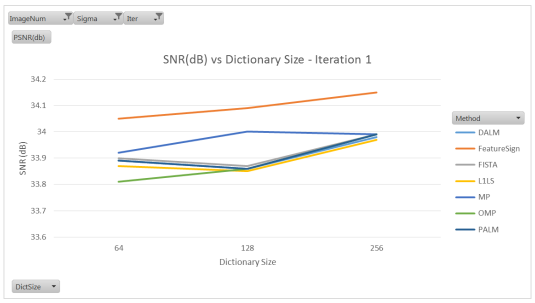
All other supplied parameters were fixed across approaches in order to maintain consistency across the experiment. Tuning them specifically to a particular method might have resulted in better results.

K-SVD toolbox [13] and l_1 Solvers from [16], [8], [5] and [7] were used for computation, and all the processes were run using matlab in nodesktop mode.

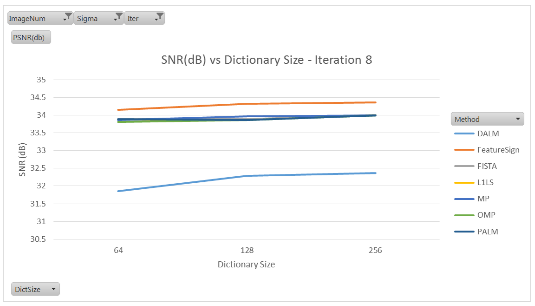
Pertinent values, images and dictionaries at each of the intermediate steps were recorded and are hosted at AML Project Results [1] under the respective results folders.

7 Results

Figure 1 shows the SNR of denoised image with different dictionary sizes. Figure 1a shows the SNR after the first iteration of K-SVD and Figure 1b after all iterations of K-SVD. The noisy image used for these results had noise level $\sigma = 10$. From these two figures we can deduce that Feature sign reaches its optimal solution after fewer iterations. We can also see DALM performs poorly for the task of denoising images.



(a) Iteration 1

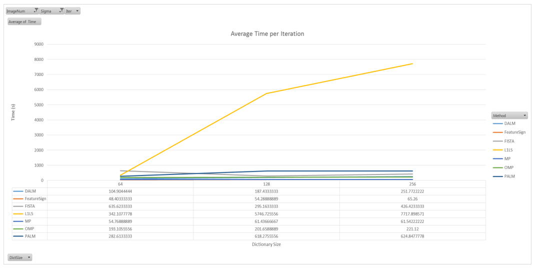


(b) Iteration 8

Figure 1: Variation with Dictionary Size

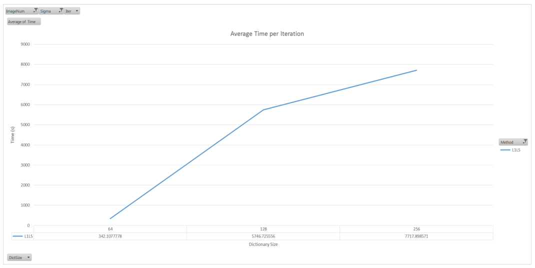
[1] https://github.com/dkdfirefly/aml

Figure 2 shows the average time per iteration for all the solvers over different dictionary sizes. From Figure 2a and 2b it is evident that L1LS takes longer time to complete. This is due to the computationally expensive Newton step in the algorithm. We remove L1LS from Figure 3 to compare average time of others. From Figure 3 we can see that MP and Feature sign are the fastest and indifferent to the dictionary size.



Dictionary Size	64	128	256
DALM	104.9044444	187.4333333	251.7722222
FeatureSign	48.40333333	54.28888889	65.26
FISTA	635.6233333	295.1633333	426.4233333
L1LS	342.1077778	5746.725556	7717.898571
MP	54.76888889	61.43666667	61.54222222
OMP	193.1055556	201.6588889	221.12
PALM	282.6133333	618.2755556	624.8477778

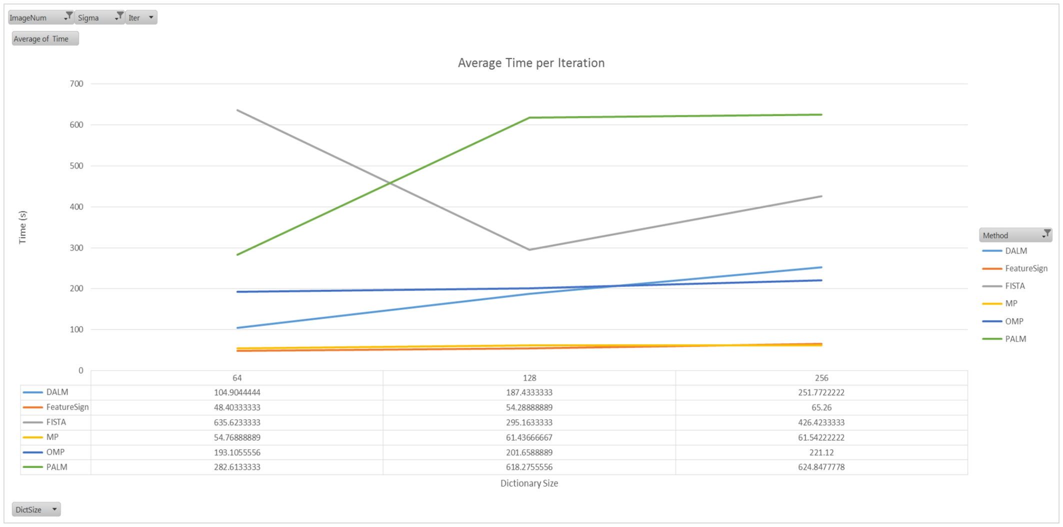
(a) Average Time per iteration



Dictionary Size	64	128	256
L1LS	342.1077778	5746.725556	7717.898571

(b) Average Time per iteration for L1LS

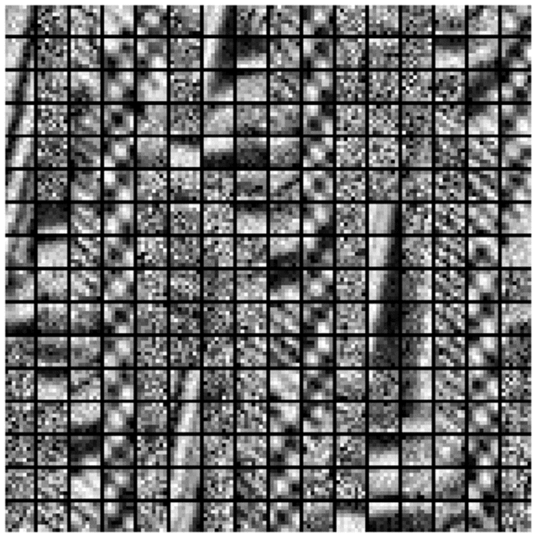
Figure 2: Avg Execution Time



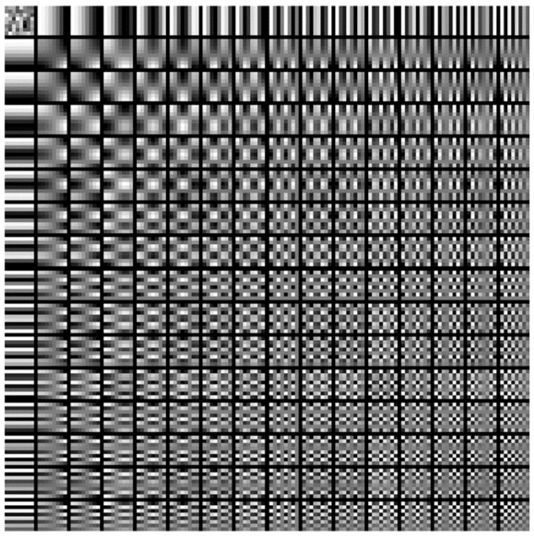
Dictionary Size	64	128	256
DALM	104.9044444	187.4333333	251.7722222
FeatureSign	48.40333333	54.28888889	65.26
FISTA	635.6233333	295.1633333	426.4233333
MP	54.76888889	61.43666667	61.54222222
OMP	193.1055556	201.6588889	221.12
PALM	282.6133333	618.2755556	624.8477778

Figure 3: Average Time per iteration for others

We have already seen that DALM performs poorly in denoising images. This is also evident from the dictionary learnt using DALM. Dictionaries learnt by DALM and PALM are shown in Figure 4. For most of the solvers the dictionary learnt is in close lines to that of PALM.



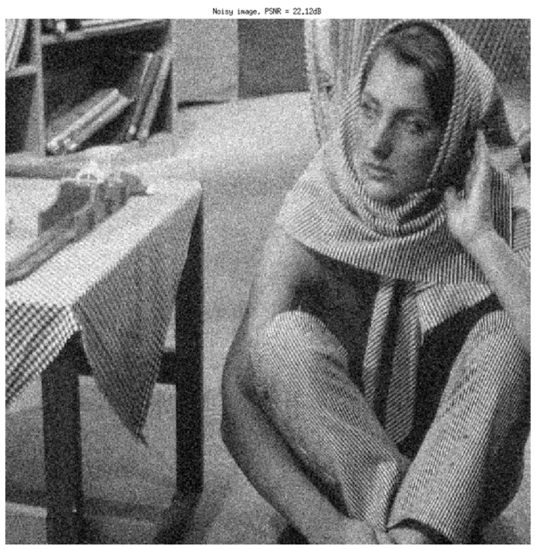
(a) Trained Dictionary for DALM
PSNR of resulting denoised image = 32.36dB



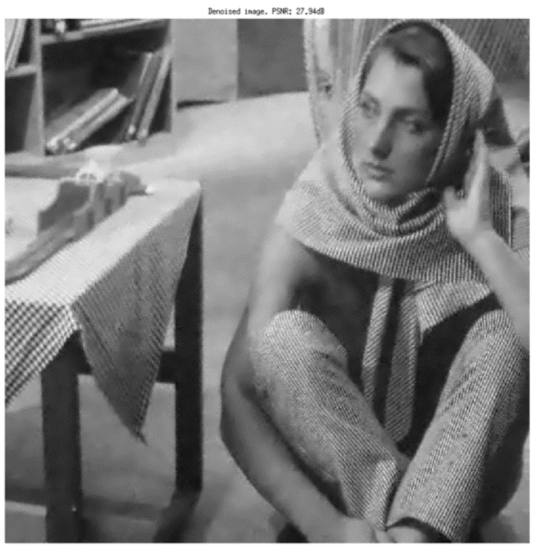
(b) Trained Dictionary for PALM
PSNR of resulting denoised image = 33.99dB

Figure 4: Trained Dictionary

Figures 5 and 6 provide a visual comparison of noisy image with the denoised image for DALM and Feature Sign case respectively.

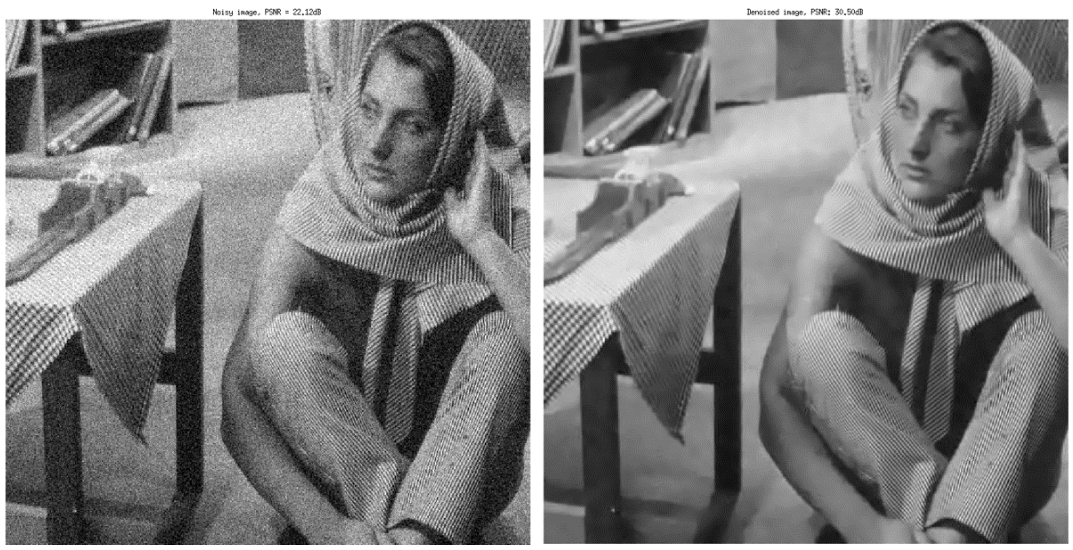


(a) Noisy Image
PSNR = 22.12dB



(b) Denoised Image for DALM
PSNR = 27.94dB

Figure 5: Images after Denoising - DALM
Sigma = 20



(a) Noisy Image
PSNR = 22.12dB

(b) Denoised Image for FeatureSign
PSNR = 30.50dB

Figure 6: Images after Denoising - Feature Sign
Sigma = 20

8 Conclusion

In this paper, we have studied the behaviour of various sparse coding algorithms when used in K-SVD for Dictionary Learning. We provide a comprehensive comparison of timing and denoising performance for different sparse coding techniques over varying dictionary sizes and noise levels. We see that image denoising performance improves with increasing dictionary size. Feature-sign gives best results and converges to optimal value in fewer iterations and faster than the other methods. We also found that DALM performs poorly in the denoising task. L1LS has a computationally expensive Newton step and was seen in the higher per iteration time. We also found that feature sign and MP are the fastest and their per iteration time is indifferent to dictionary size.

References

[1] Michal Aharon, Michael Elad, and Alfred Bruckstein. -svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on*, 54(11):4311–4322, 2006.

[2] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[3] Dimitri P Bertsekas. Nonlinear programming. 1999. *Athena Scientific*.

[4] Richard Courant et al. Variational methods for the solution of problems of equilibrium and vibrations. *Bull. Amer. Math. Soc*, 49(1):1–23, 1943.

[5] DL Donoho, I Drori, V Stodden, and Y Tsaig. Sparselab. software, 2005.

[6] Magnus R Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

[7] Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, and Stephen Boyd. An efficient method for compressed sensing. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 3, pages III–117. IEEE, 2007.

[8] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. *Advances in neural information processing systems*, 19:801, 2007.

[9] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 689–696. ACM, 2009.

[10] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12):3397–3415, 1993.

[11] Yagyensh Chandra Pati, Ramin Rezaiifar, and PS Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

[12] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964.

[13] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. *CS Technion*, page 40, 2008.

[14] Ryota Tomioka and Masashi Sugiyama. Dual-augmented lagrangian method for efficient sparse reconstruction. *Signal Processing Letters, IEEE*, 16(12):1067–1070, 2009.

[15] SJ Wright and J Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.

[16] Allen Y Yang, Zihan Zhou, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Fast l1-minimization algorithms for robust face recognition. *arXiv preprint arXiv:1007.3753*, 2010.